ISIT'2013 1569719283

Locally Repairable Codes with Multiple Repair Alternatives

Lluis Pamies-Juarez, Henk D.L. Hollmann and Frédérique Oggier
School of Physical and Mathematical Sciences
Nanyang Technological University
Singapore
Email: {lpjuarez,henk.hollmann,frederique}@ntu.edu.sg

***Abstract*—Distributed storage systems need to store data redundantly in order to provide some fault-tolerance and guarantee system reliability. Different coding techniques have been proposed to provide the required redundancy more efficiently than traditional replication schemes. However, compared to replication, coding techniques are less efficient for repairing lost redundancy, as they require retrieval of larger amounts of data from larger subsets of storage nodes. To mitigate these problems, several recent works have presented locally repairable codes designed to minimize the repair traffic and the number of nodes involved per repair. Unfortunately, existing methods often lead to codes where there is only one subset of nodes able to repair a piece of lost data, limiting the local repairability to the availability of the nodes in this subset.**

In this paper, we present a new family of locally repairable codes that allows different trade-offs between the number of contacted nodes per repair, and the number of different subsets of nodes that enable this repair. We show that slightly increasing the number of contacted nodes per repair allows to have repair alternatives, which in turn increases the probability of being able to perform efficient repairs.

Finally, we present pg-BLRC, an explicit construction of locally repairable codes with multiple repair alternatives, constructed from partial geometries, in particular from Generalized Quadrangles. We show how these codes can achieve practical lengths and high rates, while requiring a small number of nodes per repair, and providing multiple repair alternatives.

I. Introduction

In recent years, distributed storage systems as used in large data centers have started to incorporate coding techniques to redundantly store data across different storage nodes. For example, Facebook reported that it is archiving old data using a classic Reed-Solomon code implemented on top of the Hadoop Distributed File System (HDFS) [1], [2], and Microsoft uses a Pyramid Code as the main storage primitive of its Azure storage service [3]. The use of coding mechanisms in these distributed storage systems provides significantly higher fault-tolerance values and lower storage overheads than simpler replication schemes [4]. For example, in systems like HDFS or Azure, coding techniques allow to store data with a footprint of 1.3–1.5 times the size of the original object, which represents a footprint reduction of 50% as compared to the *de-facto* standard 3-replica scheme.

The main problem of using traditional coding techniques in distributed storage systems is that repairing lost encoded data requires the retrieval of large amounts of data from large subsets of nodes, which entails an important network traffic and lots of input/output (I/O) operations. In today's large distributed storage systems where node failures are the norm rather than the exception, minimizing the communication costs due to data repairs has therefore become an important problem. Regenerating Codes [5] were the first families of a new wave of codes especially designed to minimize repair costs in distributed storage systems. Regenerating Codes address the repair problem by describing an optimal trade-off between the storage overhead of the code and its repair communication costs. However, repair processes in Regenerating Codes require contacting a large subset of nodes, which complicates the design of the storage system and increases the number of required I/O operations.

A different line of new codes, called locally repairable codes (LRC), also addresses the repair problem, but focusing on reducing the number of nodes contacted during repair [6]–[11], while still guaranteeing a low repair traffic. However, the main problem with existing locally repairable codes is that although they reduce the size of the subset of contacted nodes, they suffer from the drawback that only a single subset of nodes enables the repair of a specific piece of redundant data. If a single node from this repair subset is not available, data cannot be repaired "locally", increasing the cost of the repair. Alternatively, some earlier codes like Pyramid codes [12], or Hierarchical codes [13], provide different subsets of nodes that enable the repair of each piece of redundant data. However, these different repair subsets do not have all the same size: during normal operations, lost data can be repaired using the smallest subset, however, when the number of unavailable nodes grows, repairs require the use of larger subsets, thus increasing the repair cost.

In order to maximize the reliability of storage systems it is therefore desirable to obtain codes where lost data can be repaired by contacting a small number of nodes r, where this number can be as small as $r = 2$. In addition, since the repair process might not be able to contact some of these r nodes during repair (e.g., due to temporary node unavailabilities, or even a correlated failure of multiple nodes), it is also desirable to have $a > 1$ different alternative r-subsets of nodes enabling the repair of any lost data. Unfortunately, although some locally repairable codes [8], [9] present constructions where $a > 1$, to the best of our knowledge there is no publication focusing on the analysis and design of codes that allow a *trade-off* between the values of a and r.

Contributions:

In this paper, we present a new framework to facilitate the analysis and design of locally repairable codes with different trade-offs between their repair locality and their number of repair alternatives per failure. This framework allows us to define the "local repair tolerance" as a new metric that measures

the maximum number of nodes that can be unavailable in the system without compromising the ability to locally repair all the stored data. Furthermore, we present pg-BLRC codes, an explicit construction of locally repairable codes with multiple repair alternatives, constructed from partial geometries. We provide an upper and lower bound for the rate of such codes, which is attained by the class of Generalized Quadrangles. Design of efficient pg-BLRC codes involves a trade-off among three of its desirable features: (i) small repair locality, (ii) large number of repair alternatives, and (iii) low storage footprint. Throughout numerical evaluations we identify that optimizing individually each of these properties leads to a bad performance of the other two. Requiring multiple repair alternatives thus introduces a new point of view in the study of the optimal trade-offs between the rate and the repair locality of storage codes.

II. Linear Codes for Distributed Storage

Let q be a prime power. A q-ary linear code C of length n and rank k is a k-dimensional linear subspace of the vector space $\mathbb{F}_q^n$, where $\mathbb{F}_q$ is the finite field with q elements and $n \geq k$. As a linear subspace, the code C can be defined by a $k \times n$ full-rank generator matrix G as

$$C = \left\{ \mathbf{o}G : \mathbf{o} \in \mathbb{F}_q^k \right\},$$

where the vectors $\mathbf{c} \in C$ are called the *codewords* of C. The code C can alternatively be defined using an $(n-k) \times n$ full-rank parity check matrix H such that $GH^\top = 0$, as

$$C = \left\{ \mathbf{c} \in \mathbb{F}_q^n : H\mathbf{c}^\top = 0 \right\}.$$

In distributed storage systems, a data object of k symbols is represented as a vector $\mathbf{o} \in \mathbb{F}_q^k$, containing $k\lceil \log_2 q \rceil$ bits. To redundantly store this object across different storage nodes, the system first obtains the codeword $\mathbf{c} = \mathbf{o}G \in \mathbb{F}_q^n$, and then it stores the n symbols of $\mathbf{c}$ on n different storage nodes. Since $n \geq k$, each stored object requires a disk capacity larger than its original size. The rate $R = k/n$ of the code represents the proportion of storage capacity that is used to store non-redundant symbols. The closer R is to one the more storage-efficient the code is. The efficiency of codes is also measured in terms of the storage footprint, n/k, which is the capacity used to store each data object compared to its original size.

After storing the codeword $\mathbf{c}$ the system can reconstruct the original object $\mathbf{o}$ by gathering some k symbols out of the n stored ones. For that, let $\mathcal{I}$, $|\mathcal{I}| = k$, be a set containing the indexes of these k symbols. Then the object $\mathbf{o}$ can be reconstructed by solving the system $\mathbf{o} = \mathbf{c}_\mathcal{I} G_\mathcal{I}^{-1}$, where $\mathbf{c}_\mathcal{I}$ is the vector composed of the elements of $\mathbf{c}$ that are indexed by the members of $\mathcal{I}$, and similarly, $G_\mathcal{I}$ is the submatrix of G composed of the columns of G that are also indexed by the members of $\mathcal{I}$. It is important to note that the previous system can only be solved if the matrix $G_\mathcal{I}$ is invertible. When this matrix is invertible for any k-subset $\mathcal{I}$ we say that the code is a maximum distance separable code (or MDS code).

III. Data Repairability of Linear Codes

As pointed out in the introduction, providing efficient mechanisms to repair lost encoded data is an important problem in distributed storage systems. In this section we introduce an analytic framework to evaluate the repairability properties of different linear codes, focusing on codes providing local repairs.

Let the code $C^\perp$, the dual of C, be the code generated by the parity check matrix H. Then, by definition all codewords $\mathbf{v} \in C^\perp$ are parity check vectors of C, which means that $\mathbf{v}\mathbf{c}^\top = 0$ for any $\mathbf{c} \in C$. This parity check property can be used to construct repair mechanisms able to repair lost symbols in the codewords of C. For that, let $\mathbf{v} \in C^\perp$ be a parity check vector with a nonzero ith symbol, that is $\mathbf{v}(i) \neq 0$. Then, repairing the ith symbol of $\mathbf{c}$, $\mathbf{c}(i)$, consists of solving the equation $\mathbf{v}\mathbf{c}^\top = 0$, or equivalently, solving

$$\mathbf{v}(1)\mathbf{c}(1) + \mathbf{v}(2)\mathbf{c}(2) + \cdots + \mathbf{v}(n)\mathbf{c}(n) = 0.$$

This equation has then as many unknowns as number of nonzero symbols in $\mathbf{v}$, or $\mathrm{w}(\mathbf{v})$ unknowns, where w is the Hamming weight function. Then, repairing a missing symbol of $\mathbf{c}$ requires to retrieve $\mathrm{w}(\mathbf{v}) - 1$ other symbols and solve the equation for $\mathbf{c}(i)$. However, retrieving $\mathrm{w}(\mathbf{v}) - 1$ symbols over a communication network might entail a significant overhead in terms of network traffic. Consequently, to minimize this traffic it is important to design codes C guaranteeing that for every $i \in [n] = \{1, 2, \ldots, n\}$ there exists at least one vector $\mathbf{v} \in C^\perp$ with $\mathbf{v}(i) \neq 0$, and small Hamming weight $\mathrm{w}(\mathbf{v})$.

To analyze the repair efficiency of codes we can enumerate all the possible ways in which the ith symbol of a codeword $\mathbf{c} \in C$ can be repaired. To that end, we define $\Omega(i)$ as the set containing all the parity check vectors repairing this symbol:

$$\Omega(i) = \left\{ \mathbf{v} \in C^\perp : \mathbf{v}(i) \neq 0 \right\}.$$

Then, for each $i \in [n]$, we can evaluate the cost of repairing the ith symbol by analyzing the Hamming weight of all vectors $\mathbf{v} \in \Omega(i)$. The *repair degree* is a metric that describes the number of symbols that need to be retrieved per repair:

Definition 1 (Repair Degree)**.** *We define the repair degree for the ith codeword symbol as $r(i) = \min\{\mathrm{w}(\mathbf{v}) - 1 : \mathbf{v} \in \Omega(i)\}$, and the overall repair degree r of a linear code is its maximum repair degree: $r = \max\{r(i)\}_{i=1}^n$.*

In classic MDS codes such as Reed-Solomon codes [14], or in Regenerating Codes [5], we have that the repair degree is at least equal to the rank of the code, $r \geq k$. For Reed-Solomon codes it means that repairing a single failure requires to transfer an amount of information equal to the size of the original object, or $k\lceil \log_2 q \rceil$ bits. For Regenerating Codes, although they still have to contact at least k nodes per repair, the overall amount of data transferred per repair can be slightly reduced below the size of the original object. However, instead of aiming at reducing the repair traffic, in this paper we are interested in codes able to improve the repair performance by reducing the number of nodes that need to be contacted per repair below k, which in turn leads to reduce the network traffic per repair as well. We will refer to those codes with an overall repair degree much smaller than its rank ($r \ll k$) as *locally repairable codes*, or LRC.

IV. Codes with Multiple Repair Alternatives

Even though locally repairable codes significantly reduce the number of nodes that need to be contacted during repairs, it is often also desirable to guarantee the existence of multiple subsets of this kind. This is especially important in storage

systems where some of the nodes to be contacted might be unavailable, either because they are temporarily busy, or because there was a correlated failure affecting several nodes. In this section we measure the multiple repair alternatives of LRC codes and their ability to perform local repairs in the presence of node unavailabilities.

Let $\Omega_r(i)$ be the subset of $\Omega(i)$ containing the vectors that allow to repair $\mathbf{c}(i)$, $\mathbf{c} \in C$, with a repair degree at most r, i.e., $\Omega_r(i) = \{\mathbf{v} \in \Omega(i) : \mathrm{w}(\mathbf{v}) \leq r+1\}$. Each of these vectors represents then a possible alternative to repair the ith symbol of any codeword $\mathbf{c} \in C$.

Definition 2 (Repair Alternativity)**.** *The repair alternativity of the ith codeword symbol is the number of distinct subsets of nodes with at most r nodes, which contain enough information to repair the ith symbol. The repair alternativity of i is then $a(i) = |\Omega_r(i)|$, and the code's overall repair alternativity is $a = \min\{a(i)\}_{i=1}^{n}$.*

Non-locally repairable MDS codes such as Reed-Solomon codes and Regenerating Codes have a large repair alternativity of $a = \binom{n}{r}$, which guarantees that all stored symbols can be repaired even when a large portion of storage nodes is unavailable. In fact, all symbols can be repaired as long as the stored information is available (that is, if k symbols are available). On the other hand, most of the existing locally repairable codes [6], [7] have a local repair alternativity of $a = 1$. In this case, if any of the r nodes involved in the repair is temporary unavailable, the code cannot use the local repair mechanisms, requiring then more expensive repair solutions. To the best of our knowledge SRC [8], [9] are the only LRC codes with a repair alternativity larger than one, $a > 1$. Unfortunately, in SRC codes the value of a depends on the code construction, which does not allow to obtain codes with arbitrary a values. Moreover, obtaining SRC codes with large a leads to unpractical codes with low rate R.

In this paper, we focus on the design of locally repairable codes with arbitrary repair alternativity and practical rates. Having codes with multiple repair alternatives increases the probability to be able to locally repair lost data when some nodes are unavailable. To maximize the local repair probability, it is therefore important to guarantee that the number of common nodes in different repair alternatives for a given symbol is as small as possible. For example, in order to maximize the number of repair alternativities of the ith symbol, $a(i)$, we have to minimize $|\mathrm{supp}(\mathbf{v}) \cap \mathrm{supp}(\mathbf{u})|$, for any distinct pair of vectors $\mathbf{v}, \mathbf{u} \in \Omega_r(i)$, where $\mathrm{supp}(\mathbf{v})$ is the support of $\mathbf{v}$, which is the set containing the indices of the non-zero positions of $\mathbf{v}$. Using this concept, we can formally define the local repair tolerance of a LRC code as follows:

Definition 3. *The local repair tolerance of the ith symbol, $\delta(i)$, is the size of the smallest set of coordinates different than i that intersects with the support of all codewords in $\Omega_r(i)$:*

$$\delta(i) = \min\{|\mathcal{I}| : \mathcal{I} \subset [n] \setminus \{i\},\ \mathcal{I} \cap \mathrm{supp}(\mathbf{v}) \neq \emptyset,\ \forall\ \mathbf{v} \in \Omega_r(i)\}.$$

The overall local repair tolerance of the code is $\delta = \min\{\delta(i)\}_{i=1}^{n}$.

This means that the ith symbol can be locally repaired when at most $\delta(i)$ nodes are unavailable, and any symbol is locally repairable when at most δ nodes are unavailable. In the design of locally repairable codes we will then aim at maximizing the value of δ. However, some code designs might have unbalanced constructions where $\delta(i) \ll \delta(j)$ for different symbols $i, j \in [n]$, where different symbols have different probability to be repaired locally. To avoid working with this unbalanced code constructions, in this paper we only focus on balanced locally repairable codes:

Definition 4 (Balanced Codes)**.** *When $\delta(i) = \delta(j)$ for all $i, j \in [n]$, $i \neq j$, we say that the code is balanced.*

We will refer to these balanced locally repairable codes as BLRC codes, and we will use the notation (n, k, r, a, δ)-code to refer to a code of length n and rank k, where each symbol of the codeword can be repaired from at most r other symbols, having at least a different sets of symbols that guarantee such a repair, and being able to locally repair each symbol if there are at most δ unavailable symbols. In the next section we will present a simple code construction of BLRC code with arbitrary a and r values.

V. Balanced Locally Repairable Codes from Partial Geometries

We present a way to generate explicit BLRC constructions from partial geometries and analytically evaluate the new codes in terms of repair tolerance δ and code rate R. We first provide a brief description of partial geometries.

A. Partial Geometries

A partial geometry $pg(s, t, \alpha)$ is an incidence structure between a set of points $\mathcal{P}$ and a set of lines $\mathcal{B}$ such that:

1) Each point $P \in \mathcal{P}$ is incident with $t+1$ lines ($t \geq 1$).
2) Each line $B \in \mathcal{B}$ is incident with $s+1$ points ($s \geq 1$).
3) Any two lines have at most one point in common.
4) If a point P and a line B are not incident, there are exactly α ($\alpha \geq 1$) pairs $(Q, M) \in \mathcal{P} \times \mathcal{B}$, such that P is incident with M and Q is incident with B.

It follows that $1 \leq \alpha \leq \min\{t+1, s+1\}$, and by definition of partial geometries the cardinalities of the point and line sets must satisfy:

$$|\mathcal{P}| = \frac{(s+1)(st+\alpha)}{\alpha} \text{ and } |\mathcal{B}| = \frac{(t+1)(st+\alpha)}{\alpha}.$$

The dual of a partial geometry, which is the incidence structure that arises from interchanging the set of points $\mathcal{P}$ with the set of lines $\mathcal{B}$, is also a partial geometry with parameters $pg(s' = t, t' = s, \alpha)$. Finally, according to the values of the parameters s, t and α, partial geometries can be divided into four classes:

1) When $\alpha = s+1$, or dually $\alpha = t+1$, the partial geometry is a Steiner 2-design.
2) When $\alpha = s$, or dually $\alpha = t$, the partial geometry is called a *net* or a *transversal design*.
3) When $\alpha = 1$, the partial geometry is called a *generalized quadrangle*.
4) For $1 < \alpha < \min\{s, t\}$ the partial geometry is *proper*.

As we will show in Section V-C, generalized quadrangles are of special interest to design optimal codes in terms of rate.

B. Codes from Partial Geometries

Partial geometries and other incidence structures have been widely studied for the construction of LPDC codes [15], [16]. The incidence matrix of partial geometries can be used as a simple mechanism to obtain sparse parity-check matrices with low rank over $\mathbb{F}_2$, which makes them particularly suitable to construct high rate LPDC codes with efficient iterative decoders. The similarity of the requirements of these LPDC codes with those of BLRC codes makes incidence matrices of partial geometries a promising source of BLRC designs. We will use some of the results of Johnson and Weller [16] to evaluate the local repairability of such codes. Although in the previous sections we considered generic q-ary codes, in this section we limit our code designs to binary codes, i.e., $q = 2$. As we will show, this limitation does not affect the rate of the obtained code.

Let us define the incidence matrix of a partial geometry $pg(s, t, \alpha)$ as a $|\mathcal{B}| \times |\mathcal{P}|$ matrix $N = (n_{ij})$, where $n_{ij} = 1$ or 0 according as whether the ith line is incident with the jth point or not. Constructing a linear code C from a partial geometry consists then of building an $m \times n$ parity check matrix H containing m linear independent rows of N, where $m = \operatorname{rank}_2(N)$. Then we can obtain the generator matrix G of the code C by solving the equation $GH^\top = 0$. If H can be expressed as $H = [I_{n-k}, Q]$, the generator matrix is then defined as $G = [-Q^\top | I_k]$. Note that this requires that Q cannot contain all zero rows. And since the generator matrix G contains an identity matrix, the code C is systematic. We will denote a code C constructed from a partial geometry $pg(s, t, \alpha)$, as a pg-BLRC code.

Besides the formal definition of pg-BLRC codes, we can also state the following lemma regarding the repair degree and the repair alternativity of such codes.

Lemma 1. *The repair degree of a pg-BLRC code C and its repair alternativity satisfies $r \leq s$, and $a \geq t + 1$.*

Proof: From the properties of partial geometries we have that every point $P \in \mathcal{P}$ is incident with $t + 1$ lines, and each of these lines is at the same time incident with $s + 1$ points. It means that for each $i \in [n]$ there are $t + 1$ rows of the incidence matrix N, namely $\mathbf{v}_0 \dots \mathbf{v}_t$, such that $\mathbf{v}_j(i) = 1$ and $\mathrm{w}(\mathbf{v}_j) = s + 1$ for all $j \in [n]$. Then, by definition of $\Omega_r(i)$ we have that $\mathbf{v}_j \in \Omega_r(i)$, for all $j = 0, \dots, t$, and hence $|\Omega_r(i)| \geq t+1$. Then, from Definition 2 we get that $a \geq t+1$, and similarly, from Definition 1 we get that $r(i) \leq s$ for all $i \in [n]$ and thus $r \leq s$. ∎

For the rest of the paper we will call (r, a) pg-BLRC codes those codes constructed from partial geometries $pg(s, t, \alpha)$, where $s > 1$. When $s = 1$ any lost data can be repaired by contacting a single node in the system, which corresponds to a simple data replication scheme. The condition $s > 1$ allows us to exclude replication schemes from the definition of our BLRC codes.

Lemma 2. *A pg-BLRC code with $s > 1$ has a per-symbol repair tolerance bounded by $\delta(i) \geq t + 1$, for all $i \in [n]$, and an overall repair tolerance bounded by $\delta \geq t + 1$.*

Proof: Let $\mathcal{N}(i) \subseteq \Omega_r(i)$ be the set containing all the rows $\mathbf{v}$ of the incidence matrix N satisfying that $\mathbf{v}(i) = 1$, for all $i \in [n]$. On the one hand, let us first assume that $\mathcal{N}(i) = \Omega_r(i)$. In this case, from property 3) of partial geometries we have that for any two $\mathbf{v}_1, \mathbf{v}_2 \in \mathcal{N}(i)$, $\mathbf{v}_1 \neq \mathbf{v}_2$, the support intersection $\operatorname{supp}(\mathbf{v}_1) \cap \operatorname{supp}(\mathbf{v}_2)$ is either the empty set or the set $\{i\}$. Then, since $|\mathcal{N}(i)| = t + 1$, the minimum subset $\mathcal{I} \subseteq [n] \setminus \{i\}$ that intersects the support of all vectors in $\mathcal{N}(i)$ satisfies $|\mathcal{I}| = t + 1$, and then from Definition 3 it follows that $\delta(i) = a(i) = t+1$ and $\delta = a = t+1$. On the other hand, if we assume $\mathcal{N}(i) \subset \Omega_r(i)$, then the extra vectors $\mathbf{w} \in \Omega_r(i) \setminus \mathcal{N}(i)$ might only contribute to increase the repair tolerance of the symbol i, and hence $\delta(i) \geq t + 1$. ∎

From the last two lemmas we know that an (r, a) pg-BLRC code C constructed from a $pg(s, t, \alpha)$ partial geometry guarantees that $r \leq s$, $a \geq t + 1$ and $\delta \geq t + 1$. Not achieving these three bounds with equality implies that there exists some codewords in the dual code $C^\perp$ with a Hamming weight smaller than $s + 1$. Due to the difficulty of finding such a type of codewords, we will refer to an (r, a) pg-BLRC code as code with a *designed* repair degree r, and a *designed* repair alternativity a. This gives to the designer of the storage system the guarantee to be able to repair *all* missing symbols by contacting r other nodes, having a alternative r-subsets that enable this repair (the lines of the geometry), and the guarantee that the system can repair any failure when at most $\delta = a$ nodes are temporary unavailable.

C. Rate Bounds of pg-BLRC Codes

In the previous sections we have presented the construction of pg-BLRC codes and the properties that allow to measure the repair performance of these codes. However, besides offering efficient repair mechanisms, codes used in distributed storage systems also need to guarantee low storage overheads (or equivalently high code rates). In this section we provide an upper and lower bound for the rate R of (r, a) pg-BLRC codes.

Theorem 1. *The rate R of an (r, a) pg-BLRC code C is lower bounded by*

$$R \geq \frac{r^2}{(a + r - 1)(r + 1)},$$

and when $r + a - 1$ is even, the rate is upper bounded by

$$R \leq \frac{a(r^2 - r + 1) - (r - 1)^2}{(a + r - 1)(r(a - 1) + 1)}.$$

Proof: The rate of an (r,a) pg-BLRC code is

$$R = \frac{k}{n} = \frac{n - \operatorname{rank}_2(N)}{n}, \tag{1}$$

where N is the incidence matrix of the partial geometry. From Johnson et al. [16] we have that the $\operatorname{rank}_2(N)$ of such codes is upper bounded by $\operatorname{rank}_2(H) \leq \vartheta + 1$, and when $s + t + 1 - \alpha$ is even, then is is lower bounded by $\vartheta \leq \operatorname{rank}_2(H)$, where

$$\vartheta = \frac{st(s + 1)(t + 1)}{\alpha(t + s + 1 - \alpha)}.$$

Lower Bound: The minimum possible rank is achieved when $\operatorname{rank}_2(N) = \vartheta + 1$. Substituting in (1) $\operatorname{rank}_2(N)$ by $\vartheta + 1$, and n by $|\mathcal{P}|$ we get:

$$R \geq \frac{-s\alpha + s(s - 1)}{-(s + 1)\alpha + s(s + 2) + t(s + 1) + 1} =: \frac{Q(\alpha)}{S(\alpha)}.$$

Since both numerator and denominator have negative slopes and $Q(\alpha) < S(\alpha)$ for all $1 \leq \alpha \leq \min\{s + 1, t + 1\}$, $s \geq 1$ and $t \geq 1$, then the maximum possible value is achieved when

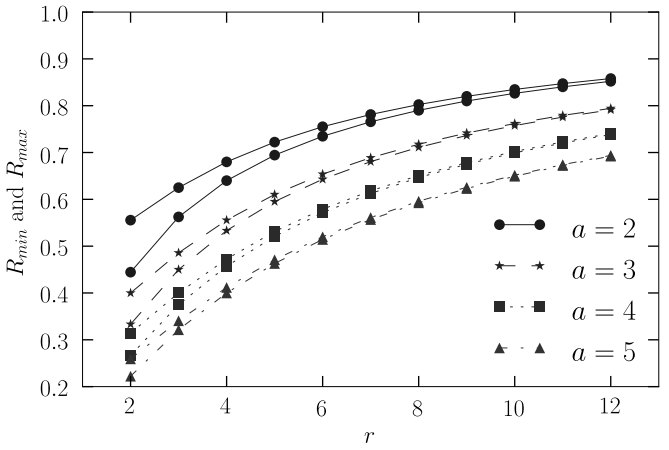

Fig. 1: *Upper bound R_{max} and lower bound R_{min} for the rate R achieved by pg-BLRC codes with repair degree r and repair alternativity a.*

$\alpha = 1$. The lower bound is obtained by evaluating $Q(1)/S(1)$ and substituting $s := r$ and $t := a - 1$.

Upper Bound: Similarly, the maximum possible rank is achieved when $\mathrm{rank}_2(N) = \vartheta$. Substituting in (1) $\mathrm{rank}_2(N)$ by ϑ, and n by $|\mathcal{P}|$ we get:

$$R \leq \frac{\alpha^2 + \alpha(st - s - t - 1) - ts^2}{\alpha^2 + \alpha(st - s - t - 1) - ts^2 - t^s - ts} = \\ = 1 + \frac{t^2 + ts}{\alpha^2 + \alpha(st - s - t - 1) - ts^2 - t^s - ts}.$$

Note that we can write the denominator as a polynomial function $Q(\alpha) = \alpha^2 + \alpha(st - s - t - 1) - ts^2 - t^2 - ts$, which has minimum value at $\alpha_{\min} = -\frac{1}{2}(st - s - t - 1)$, which satisfies $\alpha_{\min} < 1$. Then, since the valid values of α are $\alpha \in [\min\{s+1, t+1\}]$ and the function $Q(\alpha)$ is monotonically increasing in this interval, the maximum rate R is obtained when $\alpha = 1$. The upper bound is obtained by evaluating $R \leq 1 + (t^2 + ts)/Q(1)$ and substituting $s := r$ and $t := a - 1$. ∎

Remark 1. *The maximum rate of an (r, a) pg-BLRC code is achieved when the partial geometry $pg(s, t, \alpha)$ is a generalized quadrangle ($\alpha = 1$).*

In Figure 1 we use the bounds from Theorem 1 to depict the minimum and maximum[1] possible theoretical rates of an (r, a) *pg*-BLRC code with $\alpha = 1$ for different combinations of r and a values. In general it is interesting to see how the rate decreases when we either (i) decrease the repair degree r, or (ii) increase the repair alternativity a. It means that the two main objectives to achieve efficient repair mechanisms (small r values and large a values) entail an increase in the storage overhead of the system (low rate R), posing an optimization trade-off for storage system designers.

D. Explicit pg-BLRC Code Constructions

Note that there is no known construction of generalized quadrangles for all possible s and t values, and until now, only a few generalized quadrangles are known [17]. They are those with $(s, t) \in \{(2, 2), (2, 4), (3, 3), (3, 9), (3, 5), (4, 4), (4, 6), (4, 8), (4, 16)\}$, and those with $(s, t) \in \{(1, z), (q - 1, q + 1), (q, q), (q, q^2), (q^2, q^3)\}$, for integers z and prime powers q, and their dual constructions. If we evaluate all the possible (s, t) pairs and filter out the cases where $R \leq 1/3$ or $n > 100$, which are the rates and length interesting from a practical point of view, then we have possible BLRC codes for the following pairs of parameters: $(r, a) \in \{(r, 2), (2, 3), (4, 3), (3, 4), (5, 4), (4, 5)\}$.

[1]Limited to the cases where $r + a - 1 \equiv 1 \mod 2$.

VI. Conclusions

In this paper, we presented a new approach to design locally repairable codes. Instead of only focusing on codes achieving minimum repair locality and maximum rate, we analyze how to increase the diversity of this repair locality, by providing more than one local repair alternative for data blocks that need repair. We present an explicit construction of locally repairable codes that provide different trade-offs between repair locality and number of repair alternatives per failure. We also provide an upper and lower bound on the attainable rate of such codes.

Acknowledgments

The authors would like to thank C. Bracken for his valuable comments. The research of L. Pamies-Juarez, H.D.L. Hollmann and F. Oggier is supported by the Singapore National Research Foundation under Research Grant NRF-CRP2-2007-03.

References

[1] A. Thusoo, Z. Shao, S. Anthony, D. Borthakur, N. Jain, J. Sen Sarma, R. Murthy, and H. Liu, "Data warehousing and analytics infrastructure at facebook," in *Proceedings of the 2010 ACM SIGMOD Intl. Conference on Management of data*, ser. SIGMOD '10, 2010.

[2] L. X. Bin Fan, Wittawat Tantisiriroj and G. Gibson, "Diskreduce: Replication as a prelude to erasure coding in data-intensive scalable computing," Carnegie Mellon University, Parallel Data Laboratory, Tech. Rep. Technical Report CMU-PDL-11-112, 2011.

[3] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin, "Erasure coding in windows azure storage," in *Proceedings of the USENIX Annual Technical Conference (ATC)*, 2012.

[4] R. Rodrigues and B. Liskov, "High availability in dhts: Erasure coding vs. replication," in *Proceedings of the 4th Intl. Workshop on Peer-To-Peer Systems (IPTPS)*, 2005.

[5] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Transactions on Information Theory*, vol. 56, no. 9, 2010.

[6] D. Papailiopoulos and A. Dimakis, "Locally repairable codes," in *Proceedings of the IEEE Intl. Symposium on Information Theory (ISIT)*, 2012.

[7] G. Kamath, N. Prakash, V. Lalitha, and P. Kumar, "Codes with local regeneration," *arXiv preprint arXiv:1211.1932*, 2012.

[8] F. Oggier and A. Datta, "Self-repairing homomorphic codes for distributed storage systems," in *The 30th IEEE Intl. Conference on Computer Communications (INFOCOM)*, 2011.

[9] ——, "Self-repairing codes for distributed storage - a projective geometric construction," in *Information Theory Workshop (ITW)*, 2011.

[10] D. S. Papailiopoulos and A. G. Dimakis, "Storage codes with optimal repair locality," in *Proceedings of the IEEE Intl. Symposium on Information Theory (ISIT)*, 2012.

[11] A. Rawat and S. Vishwanath, "On locality in distributed storage systems," in *Information Theory Workshop (ITW)*, 2012.

[12] C. Huang, M. Chen, and J. Li, "Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems," in *Intl. Symposium on Network Computing and Applications (NCA)*, 2007.

[13] A. Duminuco and E. W. Biersack, "Hierarchical codes: How to make erasure codes attractive for peer-to-peer storage systems," in *Proceedings of the 8th Intl. Conference on Peer-to-Peer Computing (P2P)*, 2008.

[14] I. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[15] X. Li, C. Zhang, and J. Shen, "Regular ldpc codes from semipartial geometries," *Acta Applicandae Mathematicae*, vol. 102, no. 1, pp. 25–35, 2008.

[16] S. Johnson and S. Weller, "Codes for iterative decoding from partial geometries," *Communications, IEEE Transactions on*, vol. 52, no. 2, pp. 236–243, 2004.

[17] C. Colbourn and J. Dinitz, *Handbook of combinatorial designs*. Chapman & Hall/CRC, 2006, vol. 42.