ISIT'2013 1569714987

Exact-Repair Regenerating Codes Via Layered Erasure Correction and Block Designs

Chao Tian, Vaneet Aggarwal, Vinay A. Vaishampayan
AT&T Labs-Research, Shannon Laboratory, 180 Park Ave., Florham Park, NJ 07932
Email: {tian,vaneet,vinay}@research.att.com

***Abstract*—A new class of exact-repair regenerating codes is constructed by combining two layers of erasure correction codes together with combinatorial block designs. The proposed codes have the "uncoded repair" property where the nodes participating in the repair simply transfer part of the stored data directly, without performing any computation. The layered error correction structure results in a low-complexity decoding process. An analysis of our coding scheme is presented. This construction is able to achieve better performance than time-sharing between the minimum storage regenerating codes and the minimum repair-bandwidth regenerating codes.**

I. Introduction

Dimakis *et al.* proposed the framework of regenerating codes [1] to investigate the tradeoff between storage (*i.e.,* data storage) and repair data transfer (*i.e.,* repair bandwidth) in an erasure coded distributed storage system. It was shown that for the functional-repair case, the problem can be converted to an equivalent network multicast problem, and thus the celebrated network coding result [2] can be applied. By way of this equivalence, the optimal tradeoff between the storage and repair bandwidth was completely characterized for functional-repair regenerating codes. The two important extreme cases of the optimal tradeoff, where the data storage is minimized and the repair bandwidth is minimized, are referred to as minimum storage regenerating (MSR) codes and minimum bandwidth regenerating (MBR) codes, respectively. The functional-repair problem is thus well understood [1], [3], [4].

The functional-repair framework implies that the repair rule and the decoding rule in the storage system may evolve over time, which incurs additional system overhead. Furthermore, functional repair does not guarantee that data is stored in systematic form, which is an important practical requirement. In contrast, exact-repair regenerating codes do not have such disadvantages. The problem of exact-repair regenerating codes was investigated in [5]–[11], all of which address either the MBR case or the MSR case. Particularly, the optimal code constructions in [5] and [7] show that the more stringent exact-repair requirement does not incur any penalty for the MBR case; the constructions in [6]–[8] show that this is also true for the MSR case. These results may lead to the impression that enforcing exact-repair never incurs any penalty compared to functional repair. However, the result in [5] shows that this is not the case, and in fact a large portion of the optimal tradeoff achievable by functional-repair codes cannot be strictly achieved by exact-repair codes.

Codes achieving tradeoff other than the MBR or the MSR points may be more suitable for a system employing exact-repair regenerating codes. From a practical point of view, codes achieving other tradeoff points may have lower complexity than using the time-sharing approach, because the MSR point requires interference alignment, and it is impossible for linear codes to achieve the MSR point for some parameters without symbol extension [6]. As such, it is important to find such codes with competitive performance and low complexity. However, it is unknown whether there exist *exact-repair* codes that can achieve a storage-bandwidth tradeoff better than time-sharing between an MBR code and an MSR code.

In this work, we develop an exact-repair linear code construction based on the combination of two layers of erasure correction codes and combinatorial block designs. We show that our construction is able to achieve tradeoff points better than the time-sharing between an MBR code and an MSR code. The two erasure correction codes are not independent, and are jointly designed to satisfy certain full rank conditions to guarantee successful decoding. The code construction we propose has the property that the nodes participating in the repair do not need to perform any computation, but can simply transmit certain stored information for the new node to synthesize and recover the lost information. The uncoded repair approach is appealing in practice, since it reduces and almost completely eliminates the computational burden at the helper nodes. This property also holds in the constructions proposed in [5] and [12]. In fact our construction was partially inspired by and may be viewed as a generalization of these codes (see also [13] though the system model there is different).

The rest of the paper is organized as follows. Section II includes some preliminary material. An example code is described in Section III. Section IV is a brief tutorial on block designs. Section V and VI give an explicit code construction for $k = n-1$, and random generator matrix based construction for $k \leq n-1$, respectively. Section VII concludes the paper.

II. Preliminaries

An (n, k, d) exact-repair regenerating code is a storage system with a total of n storage nodes (disks)[1], where any k of them can be used to reconstruct the complete data, and furthermore to repair a lost disk, the new disk may access data from any d of the remaining $n-1$ disks. Let the total

[1]From here on, we shall use "node" and "disk" interchangably.

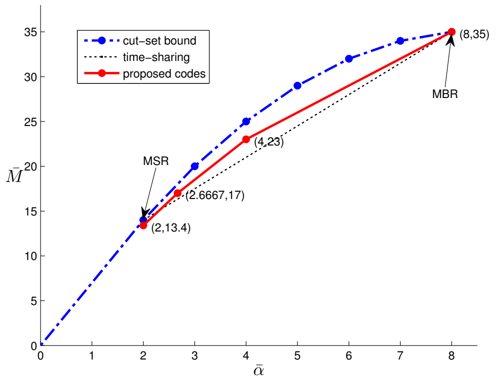

Fig. 1. Cut-set bound, time-sharing line and the performance of the proposed codes for $(n, k, d) = (9, 7, 8)$.

amount of raw data stored be M units and let each storage site store α units of data, which implies that the redundancy of the system is $n\alpha - M$. To repair (regenerate) a disk failure, each contributing disk transmits β units of data to the new node, which results in a total of $d\beta$ units of data transfer for repair. It is clear that the quantities α and β scale linearly with M, because a code can simply be concatenated. For this reason we shall normalize the other two quantities using β

$$\bar{\alpha} \triangleq \frac{\alpha}{\beta}, \quad \bar{M} \triangleq \frac{M}{\beta}, \tag{1}$$

and use them as the measure of performance from here on. In this work we mainly focus on the case when $d = n - 1$, though the construction can be generalized to other cases.

Since exact-repair is a more stringent requirement than functional-repair, the functional-repair optimal tradeoff found using cut-set analysis [1] provides an outer bound for exact-repair regenerating codes as follows:

$$\sum_{i=0}^{k-1} \min(\bar{\alpha}, (d - i)) \geq \bar{M}. \tag{2}$$

One extreme case of this outer bound is when the storage is minimized, *i.e.*, the minimum storage regenerating (MSR) point, which is

$$\bar{\alpha} = (d - k + 1), \quad \bar{M} = k(d - k + 1). \tag{3}$$

The other extreme case is when the repair bandwidth is minimized, *i.e.*, the minimum bandwidth regenerating (MBR) point, which is

$$\bar{\alpha} = d, \quad \bar{M} = \frac{k(2d - k + 1)}{2}. \tag{4}$$

Both of these extreme points are achievable [5]–[8] for the exact-repair case. The functional repair outer bound is however not tight in general [5]. The cut-set bound is illustrated as the dotted blue line in Fig. 1 for $(n, k, d) = (9, 7, 8)$; note that the bound is piece-wise linear.

III. An Example $(9, 7, 8)$ Code

To illustrate the basic code components, we shall give an example construction of a $(9, 7, 8)$ exact-repair regenerating

TABLE I
An Example $(9, 7, 8)$ Code.

Disk #	1	2	3	4	5	6	7	8	9
	P_3	X_1	Y_1	P_1	X_2	Y_2	P_2	X_3	Y_3
	X_4	X_7	Y_5	Y_4	P_5	Y_6	P_4	P_6	Y_7
	X_5	X_8	X_{11}	X_6	Y_8	P_9	P_7	P_8	P_{10}
	X_9	Y_9	X_{12}	X_{10}	Y_{10}	Y_{12}	Y_{11}	P_{11}	P_{12}

code with $M = 23$, $\alpha = 4$ and $\beta = 1$. The addition and multiplication operations in the encoding and decoding are in the finite field $\mathbb{F}(3)$, where $\mathbb{F}(q)$ denotes the finite field of cardinality q. We use I_n to denote the set $\{1, 2, \ldots, n\}$.

Let the information be given as a length-23 vector, where the i-th entry is denoted as $d_i \in \mathbb{F}(q)$. The components of this code are described below.

Encoding:

1) Generate a parity symbol $d_{24} = \sum_{j=1}^{12} d_{2j-1} + \sum_{j=1}^{11} 2d_{2j}$;
2) Pair up (d_{2j-1}, d_{2j}), and rename it as (X_j, Y_j) where $j = 1, 2, \ldots, 12$;
3) Generate a new parity symbol $P_j = X_j + Y_j$, and (X_j, Y_j, P_j) will be referred to as the jth parity group, where $j = 1, 2, \ldots, 12$;
4) Place the symbols as given in Table. I.

Repair:

Let us suppose the first disk fails. To recover, for example, symbol X_5, first obtain Y_5 and P_5 from disk-3 and disk-5, respectively, and then compute $X_5 = P_5 - Y_5$. Clearly, other symbols on the disk can also be repaired following a similar procedure. This procedure also applies to other disk failures. It can be checked that for any disk failure, each remaining disk sends a single symbol during the repair.

Reconstruction:

For data reconstruction, several different cases need to be considered. Before going into the details of these cases, consider a scenario where disk 1 and disk 2 are not accessible. Notice that although $(X_1, X_4, X_5, X_7, X_8, X_9, Y_9, P_3)$ are not accessible directly, $(X_1, X_4, X_5, X_7, X_8, P_3)$ can be recovered using the symbols on other disks, as discussed in the repair procedure above. As a consequence, only symbols in the 9-th parity group are affected, and can not be completely recovered, but even in this parity group, P_9 is still accessible on disk-6. The reconstruction cases can be classified according to which parity group is affected and which symbol within this parity group is still accessible.

1) The j-th parity group, $j \in I_{11}$, is effected, but X_j or Y_j is still accessible within it. An example case is when disk-3 and disk-4 are not accessible. Note $X_1 = d_1$ is still available in this case, and $d_1 + 2d_2$ can be computed using $d_{24} = Y_{12} = \sum_{j=1}^{12} d_{2j-1} + \sum_{j=1}^{11} 2d_{2j}$ after eliminating (d_{2j-1}, d_{2j}) pairs for $j = 2, 3, \ldots, 11$ and d_{23}, from which (d_1, d_2) can be solved. The information vector can be obtained by rearranging the symbols.
2) The j-th parity group, $j \in I_{11}$, is affected, but the parity symbol P_j is still accessible within it. An example case is when disk-2 and disk-3 are not accessible. In this case (d_{2j-1}, d_{2j}) pairs for $j = 2, 3, \ldots, 12$ can be recovered.

In addition, $P_1 = d_1 + d_2$ and $d_1 + 2d_2$ are available, from which (d_1, d_2) can be solved.

3) Parity group 12 is affected, but X_{12} is still accessible. This case is trivial since all d_j, $j = 1, 2, \ldots, 23$ have been directly recovered.
4) Parity group 12 is affected, but Y_{12} is still accessible. In this case $(X_j, Y_j) = (d_{2j-1}, d_{2j})$ for $j = 1, 2, \ldots, 11$ can be recovered, and thus only d_{23} needs to be recovered. But we have $Y_{12} = d_{24} = \sum_{i=1}^{12} d_{2j-1} + \sum_{1}^{11} 2d_{2j}$, from which d_{23} can now be obtained.
5) Parity group 12 is affected, but P_{12} is still accessible. Again (d_{2j-1}, d_{2j}) pairs for $j = 1, 2, \ldots, 11$ can be recovered. Additionally we still have $P_{12} = d_{24} + d_{23} = \sum_{j=1}^{11} d_{2j-1} + \sum_{j=1}^{11} 2d_{2j} + 2d_{23}$, from which d_{23} can be obtained.

Let us compare this code with the time-sharing code using an MBR code and an MSR code. For $(n, k, d) = (9, 7, 8)$, the MSR point is $(\bar{\alpha}, \bar{M}) = (2, 14)$ and the MBR point is $(\bar{\alpha}, \bar{M}) = (8, 35)$. Our construction achieves $(\bar{\alpha}, \bar{M}) = (4, 23)$, while the time sharing performance between the MBR point and the MSR point at $\bar{\alpha} = 4$ gives $\bar{M} = 21$, thus the example construction indeed achieves an improvement on $\bar{M}$ while keeping $\bar{\alpha}$ the same.

It is evident that the placement of the coded symbols is an important design choice. The placement given in the above example is derived from a particular combinatorial block design known as a Steiner triple system [14]. We shall give a brief introduction of block designs in the next section.

IV. Block Designs

Block designs have been considered in combinatorial mathematics with applications in experimental design, finite geometry, software testing, and cryptography.

One important class of block designs is the t-designs. The class of t-designs with parameter (λ, t, r, n) is denoted as $S_\lambda(t, r, n)$; a valid t-design in $S_\lambda(t, r, n)$ is a pair $(X, \mathcal{B})$ where X is n-element set and $\mathcal{B}$ is a collection of r-element subsets of X with the property that every element in X appears in exactly γ blocks and every t-element subset of X is contained in exactly λ blocks. Without loss of generality, one can always use $X = I_n$, and we shall use this convention from here on.

The most extensively researched class of block designs is perhaps Steiner systems, which is the case when $\lambda = 1$ and $t \geq 2$. In this case, the subscript λ is usually omitted and we directly write it as $S(t, r, n)$. The simplest design in this class is when $t = 2$ and $r = 3$, which is the particularly well understood Steiner triple systems $S(2, 3, n)$. It is known that there exists a Steiner triple system $S(2, 3, n)$ if and only if $n = 0$, or n modulo 6 is 1 or 3; see, *e.g.*, [14]. It follows that the smallest positive integer which gives us a non-trivial Steiner system is $n = 7$ and the next is 9, and an example is given in Table II using which the example code given in the previous section is constructed. Another well-known special class of t-designs is Balanced Incomplete Block Designs (BIBDs), which is a special case of t-designs for the case $t = 2$. It is clear that Steiner systems $S(2, 3, n)$ are also BIBDs.

TABLE II
Example Steiner triple systems $S(2, 3, 7)$.

$(I_9, \mathcal{B}) \in S(2, 3, 9)$	$\{(2,3,4), (5,6,7), (1,8,9), (1,4,7), (1,3,5), (4,6,8), (2,7,9), (2,5,8), (1,2,6), (4,5,9), (3,7,8), (3,6,9)\}$

For a given (λ, t, r) triple, a t-design may not exist for an arbitrary n, however, for any (t, r, n), a trivial t-design always exists with $\lambda^*(t, r, n) \triangleq \binom{n-t}{r-t}$, as given in the following proposition.

***Proposition** 1:* For any (t, r, n) where $t \leq r \leq n$, a complete block design is a block design where the blocks are all the r-element subsets of I_n (blocks are not repeated). In this design every element in I_n appears in exactly $\binom{n-1}{r-1}$ blocks and every t-element subset of X is contained in exactly $\lambda^*(t, r, n)$ blocks.

We may still refer to such a complete block design for the case of $t = 2$ as a BIBD, although it is in fact a complete block design instead of an incomplete one. The following proposition [14] is useful.

***Proposition** 2:* If $(I_n, \mathcal{B})$ is an $S_\lambda(t, r, n)$ design and S is any s-element subset of I_n, with $0 \leq s \leq t$, then the number of blocks containing S is

$$|\{B \in \mathcal{B} : S \subseteq B\}| = \lambda \binom{n-s}{t-s} \binom{r-s}{t-s}^{-1}. \tag{5}$$

The following corollary follows by setting $s = 0$ in Theorem 2.

***Corollary** 1:* If $(I_n, \mathcal{B})$ is a $S_\lambda(t, r, n)$ design, then the total number of blocks in $\mathcal{B}$ is

$$N_\lambda(t, r, n) \triangleq |\mathcal{B}| = \lambda \binom{n}{t} \binom{r}{t}^{-1}. \tag{6}$$

For the case of Steiner systems, we shall write the number of blocks as $N(t, r, n)$, omitting the subscript λ. When the parameters are clear from the context, we may also write $N_\lambda(t, r, n)$ as N^*.

There are various known constructions, existence results, and non-existence results for Steiner systems, BIBDs and t-designs in the literature; interested readers are referred to [14] for more details.

V. A Construction Based on $S(2, r, n)$

Given a block design $(I_n, \mathcal{B}) \in S(2, r, n)$ we construct an exact-repair regenerating code with parameters $(n, k, d) = (n, n-2, n-1)$ and

$$\alpha = \frac{n-1}{r-1}, \quad \beta = 1, \quad M = \frac{n(n-1)}{r} - 1. \tag{7}$$

Note that these parameters are all integers for a valid Steiner system, moreover, $n(n-1)$ is a multiple of $r(r-1)$, which can be seen using Proposition 2 and its corollary. The alphabet for this code can be chosen to be any finite field $\mathbb{F}(q)$ with a field size $q \geq r$, and the addition and multiplication operations in the coding process are performed in this field.

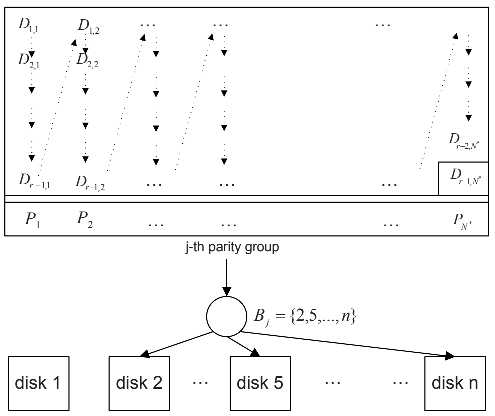

Fig. 2. Code structure based on $S(2, r, n)$.

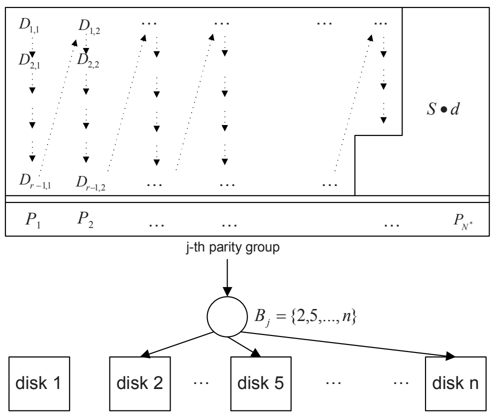

Fig. 3. Code structure based on $S_\lambda(2, r, n)$.

Let the M information symbols in $\mathbb{F}(q)$ be given in a $(r-1) \times N^*$ matrix except the bottom-right entry D_{r-1,N^*}, which is left blank. The code has several components (see Fig. 2):

Encoding:

1) Choose $(r-1)$ distinct non-zero elements $\phi_1, \phi_2, \ldots, \phi_{r-1}$ in $\mathbb{F}(q)$, which satisfy $\phi_i + 1 \neq 0$ for $i = 1, 2, \ldots, r-2$. Generate a parity symbol and assign it to D_{r-1,N^*} as
$$D_{r-1,N^*} = \sum_{i=1}^{r-2} \phi_i \sum_{j=1}^{N^*} D_{i,j} + \phi_{r-1} \sum_{j=1}^{N^*-1} D_{r-1,j}. \quad (8)$$
2) For each column $j = 1, 2, \ldots, N^*$, generate new parity symbols as
$$D_{r,j} \triangleq P_j = \sum_{i=1}^{r-1} D_{i,j}. \quad (9)$$
The collection $(D_{1,j}, D_{2,j}, \ldots, D_{r-1,j}, P_j)$ will be referred to as the j-th parity group;
3) For each block $B_j = \{b_{j,1}, b_{j,2}, \ldots, b_{j,r}\} \in \mathcal{B}$, $j = 1, 2, \ldots, N^*$, distribute the symbols in the j-th parity group onto disks $b_{j,1}, b_{j,2}, \ldots, b_{j,r}$, one symbol onto each disk.

The repair and reconstruction of this construction are straightforward generalizations of the example given in Section III, and thus we omit the details here. There are essentially two MDS codes: the long code in the construction is an $(M+1, M)$ systematic code whose parity symbol is specified by (8), and the short code is a $(r, r-1)$ systematic code whose parity symbol is specified by (9). It can be verified easily that for many cases the performance of this code is better than time-sharing performance.

VI. A Construction Based on BIBDs $S_\lambda(2, r, n)$

In this section, we generalize the construction previously described in Sec. V to the setting of exact-repair regenerating codes for any positive integer n, $d = n-1$ and any $k \le n-1$, based on BIBDs $S_\lambda(2, r, n)$.

First fix a BIBD $(I_n, \mathcal{B}) \in S_\lambda(2, r, n)$, and denote $N_\lambda(2, r, n)$ as N^*. Define the quantity
$$T(A) = \sum_{B \in \mathcal{B}: |B \cap A| \ge 2} (|B \cap A| - 1), \quad (10)$$
where $A \subset I_n$ and $|A| = n-k$, then further define
$$T = \max_{A: A \subseteq I_n,\, |A| = n-k} T(A). \quad (11)$$
The relevance of this quantity will become clear shortly. When $n - k = 2$, the definition of BIBDs gives $T = \lambda$. The construction given in the previous section belongs to this case with $T = \lambda = 1$.

The code we construct has the following parameters
$$\alpha = \frac{\lambda(n-1)}{r-1}, \quad \beta = \lambda, \quad M = \frac{\lambda n(n-1)}{r} - T. \quad (12)$$
Note that although α is always an integer, $(n-1)$ is not necessarily a multiple of $r-1$ here, unlike the previous construction. This implies that $\bar{\alpha}$ may not be an integer.

Let the M information symbols in $\mathbb{F}(q)$ be given in a vector $\boldsymbol{d}$, and let them fill the first M entries in a $(r-1) \times N^*$ matrix D following the column-wise order, *i.e.*, the first column (top-down), and the second column, etc.; the rest of the T-entries of the matrix are left blank. The code requires a matrix S of size $T \times M$, whose entries are also in $\mathbb{F}(q)$. The matrix S is used to generate the parity symbols for the long erasure code, and we shall specify the condition for S shortly.

Encoding: The encoding procedure is similar to the procedure given in the previous section, with the only difference being that we first compute the multiplication $S \cdot \boldsymbol{d}$ and then fill the rest of D matrix using the resultant T parity symbols in a column-wise manner; see Fig. 3.

Repair: The repair is similar to the repair procedure given in the Section III. Note that each remaining disk contributes exactly λ symbols, which is implied by the definition of $S_\lambda(2, r, n)$.

Reconstruction: Let $(n-k)$ disks in the set A be inaccessible. For each parity group $j = 1, 2, \ldots, N^*$, construct a length-r vector $\boldsymbol{z_i}$ as follows

- If $B_j \cap A \le 1$: collect, and if necessary, compute using (9), the symbols $D_{i,j}$, $i = 1, 2, \ldots, r-1$; let

$\boldsymbol{z_j} = (D_{1,j}, D_{2,j}, \ldots, D_{r-1,j}, 0)^t$;

- If $B_j \cap A \geq 2$: collect the available symbols in this parity group, denoted as $(D_{i_1,j}, D_{i_2,j}, \ldots, D_{i_l,j})$, assign $D_{i_1,j}, D_{i_2,j}, \ldots, D_{i_l,j}$ to the $i_1, i_2, \ldots, i_l$ positions of vector $\boldsymbol{z_j}$, and let the rest of $\boldsymbol{z_j}$ be zeros.

Finally let $\bar{\boldsymbol{d}}_A = [\boldsymbol{z_1}^t, \boldsymbol{z_2}^t, \ldots, \boldsymbol{z_{N^*}}^t]^t$, *i.e.*, concatenate the vectors $\boldsymbol{z_j}$'s. The entries of $\bar{\boldsymbol{d}}_A$ are linear combinations of $\boldsymbol{d}$. Our claim is that by properly choosing S, the vector $\boldsymbol{d}$ can be reconstructed from $\bar{\boldsymbol{d}}_A$ for any possible set A.

This construction is illustrated in Fig. 3, from which the difference and similarity from the construction given in the previous section is straightforward. For the case $r = 2$, the proposed construction is precisely the repair-by-transfer construction in [5]. In this case, the parity symbol P_j is a simple repetition, and the $S_\lambda(2, 2, n)$ design is when $\lambda = \lambda^* = 1$ in the trivial complete block design.

Next we show that a matrix S with the desired properties indeed exists. Note that as long as the transfer matrix between $\bar{\boldsymbol{d}}_A$ and $\boldsymbol{d}$ has rank M, the information vector $\boldsymbol{d}$ can be correctly reconstructed. To identify this matrix, first construct a template matrix R of size $r \times (r-1)$ as $R = [I, \mathbf{1}^t]^t$, where I is the identity matrix, and $\mathbf{1}$ is the all one vector of length $r-1$. For each $j = 1, 2, \ldots, N^*$, construct matrix R_j of size $r \times (r-1)$ as follows,

- If $B_j \cap A \leq 1$, let R_j be R with the last row set to all zeros;
- If $B_j \cap A = r - l \geq 2$, then let the corresponding symbols in the i-th parity group stored on disks $B_j \backslash A$ be $D_{i_1}, D_{i_2}, \ldots, D_{i_l}$. Keep the rows $i_1, i_2, \ldots, i_l$ in R, and assign the other rows as all zeros, and let the resultant matrix be R_j.

Finally form a matrix Q_A of size $(rN^*) \times (r-1)N^*$ using matrix R_j's as the diagonal. Clearly, we have

$$Q_A \cdot G \cdot \boldsymbol{d} = \bar{\boldsymbol{d}}_A, \tag{13}$$

where $G = [I, S^t]^t$. Thus as long as $Q_A \cdot G$ has rank M for each set $A \subset I_n$ such that $|A| = n-k$, the information vector $\boldsymbol{d}$ can be correctly decoded no matter which $n-k$ disks are inaccessible. We have the following proposition.

Proposition 3: Among the q^{TM} distinct assignments of S, at most a fraction of $q^{-1}\binom{n}{k}TM$ may induce a matrix $Q_A \cdot G$ with rank less than M for some $A \subset I_n$ such that $|A| = n-k$.

The proof relies on the well-known Schwarz-Zippel lemma, and we omit the details here due to space constraint. As a consequence of this proposition, when $q > \binom{n}{k}TM$, there exists at least one valid choice of matrix S.

A particularly important class of codes using the proposed method is when $S_\lambda(2, r, n)$ is in fact the complete block design, *i.e.*, $\lambda = \lambda^*(2, r, n)$. For this case, it can be shown

$$T = T^* \triangleq \sum_{i=2}^{\min(n-k,r)} (i-1)\binom{n-k}{i}\binom{k}{r-i}. \tag{14}$$

Using this block design, we have the following theorem.

Theorem 1: For any $n > k$, and $d = n-1$, there exists an (n, k, d) exact-repair regenerating codes in any finite field size larger than $\binom{n}{k}T^*M$ such that

$$\alpha = \frac{\lambda^*(n-1)}{r-1}, \quad \beta = \lambda^*, \quad M = \frac{\lambda^* n(n-1)}{r} - T^*. \tag{15}$$

One can straightforwardly compare it with the time-sharing performance using an MSR code and an MBR code. We omit the details of such comparison due to space constraint.

VII. Conclusion

A new construction for (n, k, d) exact-repair regenerating codes is proposed by combining two layers of error correction codes together with combinatorial block designs. The resultant codes have the desirable "uncoded repair" property where the nodes participating in the repair simply send certain stored data without performing any computation. The proposed code is able to achieve performance better than the time-sharing between an MSR code and an MBR code for some parameters. Though here we focus on the case $d = n-1$, the construction can also be generalized to the case of $d < n-1$, which is not included here due to space constraint.

References

[1] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. Wainwright and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Information Theory*, vol. 56, no. 9, pp. 4539-4551, Sep. 2010.

[2] R. Ahlswede, Ning Cai, S.-Y.R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Information Theory*, vol. 46, no. 4, pp. 1204-1216, Jul. 2000.

[3] Y. Wu, "Existence and construction of capacity-achieving network codes for distributed storage," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 2, pp. 277-288, Feb. 2010.

[4] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A survey on network codes for distributed storage," *Proceedings of the IEEE*, vol. 99, no. 3, pp. 476-489, Mar. 2011.

[5] N. B. Shah, K. V. Rashmi, P. V. Kumar and K. Ramchandran, "Distributed storage codes with repair-by-transfer and non-achievability of interior points on the storage-bandwidth tradeoff," *IEEE Transactions on Information Theory*, vol. 58, no. 3, pp. 1837-1852, Mar. 2012.

[6] N. B. Shah, K. V. Rashmi, P. V. Kumar and K. Ramchandran, "Interference alignment in regenerating codes for distributed storage: necessity and code constructions," *IEEE Transactions on Information Theory*, vol. 58, no. 4, pp. 2134-2158, Apr. 2012.

[7] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction," *IEEE Transactions on Information Theory*, vol. 57, no. 8, pp. 5227-5239, Aug. 2011.

[8] V. Cadambe, S. Jafar, H. Maleki, K. Ramchandran and C. Suh, "Asymptotic interference alignment for optimal repair of MDS codes in distributed storage," *IEEE Trans on Information Theory*, vol. 59, no. 5, pp. 2974-2987 , May 2013.

[9] D. S. Papailiopoulos, A. G. Dimakis, and V. Cadambe, "Repair optimal erasure codes through Hadamard designs," arXiv:1106.1634.

[10] I. Tamo, Z. Wang, and J. Bruck, "MDS array codes with optimal rebuilding," in Proceedings *2011 IEEE International Symposium on Information Theory*, St. Petersberg, Russia, Aug. 2011, pp. 1240-1244.

[11] V. R. Cadambe, C. Huang, S. A. Jafar, and J. Li, "Optimal repair of MDS codes in distributed storage via subspace interference alignment," arXiv:1106.1250.

[12] D. S. Papailiopoulos, J. Luo, A. G. Dimakis, C. Huang, and J. Li, "Simple regenerating codes: network coding for cloud storage," in Proc. *2012 IEEE INFOCOM*, Orlando FL, Mar. 2012.

[13] S. El Rouayheb and K. Ramchandran, "Fractional repetition codes for repair in distributed storage systems," in Proceedings *48th Annual Allerton Conference on Communication, Control and Computation*, Monticello, Sep. 2010.

[14] C. J. Colbourn and J. H. Dinitz, *Handbook of Combinatorial Designs, Second Edition (Discrete Mathematics and Its Applications)*, Chapman and Hall/CRC, Nov. 2006.