A New Cooperative Repair Scheme With $k+1$ Helper Nodes for $(n, k)$ Hadamard MSR Codes With Small Sub-Packetization

Yajuan Liu, Han Cai, Member, IEEE, and Xiaohu Tang, Senior Member, IEEE

Abstract—The cooperative repair model is an available technology to deal with multiple node failures in distributed storage systems. Recently, explicit constructions of cooperative MSR codes were given by Ye (IEEE Transactions on Information Theory, 2020) with sub-packetization level $(d-k+h)(d-k+1)$. Specifically, the sub-packetization level is $(h+1)2^n$ when $d=k+1$. In this paper, we propose a new cooperative repair scheme by means of the inter-instance pairing and intra-instance pairing inherited from the perfect code which reduces the sub-packetization to $2^n$ when $(h+1)2^n$ and $(2\ell+1)2^n$ when $h+1=(2\ell+1)2^n$ for $\ell \geq 1$, $m \geq 0$ with $d=k+1$ helper nodes. That is to say, the sub-packetization is $h+1$ times or $2^n$ times less than Ye’s. It turned out to be the best result so far known.

Index Terms—Cooperative repair, MDS codes, perfect codes, sub-packetization, optimal repair bandwidth.

I. INTRODUCTION

WITH the wide deployment of large-scale distributed storage systems, such as Facebook’s coded Hadoop, Google Colossus, and Microsoft Azure, reliability is becoming one of the major concerns so that the redundancy is imperative. In Particular, the maximum distance separable (MDS) code, such as Reed-Solomon code [7], offers maximum reliability at the same redundancy level and thus is the most attractive. However, when a storage node fails, the conventional MDS codes adopt a naive recovery strategy which first reconstructs the original file and then repairs the failed node. It results in a large repair bandwidth, which is defined as the amount of data downloaded to repair failed nodes.

In [2], regenerating codes introduced by Dimakis et al. are shown to achieve the best tradeoff between the repair bandwidth and storage overhead. The minimum storage regenerating (MSR) code is one of the two most important regenerating codes, which can maintain the systematic MDS property and has the optimal repair bandwidth. The various MSR code constructions have been proposed, refer to [1] for details.

The aforementioned MSR codes only focus on a single node failure. Whereas, multiple node failures in large-scale distributed storage systems are the norm rather than the exception. In this scenario, to reduce the management cost, the repair mechanism is triggered only after that the total amount of the failed nodes reaches a given threshold. So far, there are two models for repairing multiple node failures. The first one is the centralized repair model, where all the failed nodes are recreated at a data center. The other is the cooperative repair model, where the new nodes respectively download data from the helper nodes and then communicate with each other to finish the repair process. It is proved in [11] that the cooperative repair model is stronger than the centralized repair model since the optimality of an MDS code under the former implies its optimality under the latter. Besides, the cooperative repair model is more suitable for distributed storage systems owing to the distributed architecture. Therefore, we concentrate on the cooperative repair model in this paper.

The $(n, k)$ MDS code $C$ consists of $k$ systematic nodes and $r=n-k$ parity nodes, each node storing $N$ symbols. It is a typical high rate storage code in distributed storage systems such that the data of any $k$ out of $n$ nodes suffice to reconstruct the whole source data. Specifically, $N$ is said to be the sub-packetization level of code $C$ in the literature. Assume that there are $h$ failed nodes. To repair them, one needs to connect $d \geq k$ helper nodes. Under the cooperative repair model, the repair process is composed of two phases [8]:

1) **Download phase**: Any failed node downloads $\beta_1$ symbols from each of $d$ helper nodes, respectively;

2) **Cooperative phase**: For any two failed nodes, each transfers $\beta_2$ symbols to the other.

Accordingly, the repair bandwidth is $\gamma = b(d\beta_1 + (h-1)\beta_2)$. In [3] and [8], the repair bandwidth of an MDS array code $C$ is proved to be lower bounded by

$$\gamma \geq \frac{b(d+h-1)N}{d-k+h}. \quad (1)$$

Especially, the optimal repair bandwidth $\frac{b(d+h-1)N}{d-k+h}$ in (1) is achieved only for $\beta_1 = \beta_2 = \frac{N}{d-k+h}$. In [11], Ye and Barg introduced the first explicit construction with optimal cooperative repair property for the sub-packetization level.
Follow-up works committed to reduce the sub-packetization of MDS codes with the optimal cooperative repair property. In [12], Zhang et al. proposed a code with the optimal access property while decreasing the sub-packetization to \((d-k+h)^2\). Remarkably, the recent work of Ye [10] dramatically lowers the sub-packetization to \((d-k+h)^{(h+1)}\). To reduce the repair bandwidth of an MDS code, one general method is to pile up its multiple codewords [6]. In the literature, this method is named as space sharing and these codewords are also referred to as instances. For example, the construction in [10] is an extension of the Hadamard MSR code in [5] by space sharing \(d-k+h\) instances. To attain the optimal cooperative repair property, the key technique thereby is the inter-instance pairing, which properly pairs the symbols across instances. However, it ignores some helpful symbol pairs inside the instance, i.e., the original Hadamard MSR code, called intra-instance pairing hereafter. Addressing this issue, in this paper we further present the intra-instance pairing method based on the Hamming code. When \(d=k+1\), with the help of the intra-instance pairing, we are able to obtain the optimal cooperative repair scheme for the Hadamard MSR code without or with less space sharing. Precisely, for the case \((h+1) \mid 2^n\), we utilize only the intra-instance pairing for the original Hadamard MSR code, whose sub-packetization is \(h+1\) times less than the one in [10]. Whereas, for the case \((h+1) \nmid 2^n\), i.e., \(h+1 = (2^\ell + 1)2^m\) for \(\ell \geq 1, m \geq 0\), we firstly extend the original Hadamard MSR code by space sharing its \(2\ell + 1\) instances and then take full advantage of both the inter-instance pairing and the intra-instance pairing. In this case, the sub-packetization is \(2^n\) times less than the one obtained in [10]. Table I illustrates the comparison.

The remainder of this paper is organized as follows. In Section II, some necessary preliminaries of Hadamard MSR codes and Hamming codes are reviewed. In Section III, a general repair principle is proposed. Then following the principle, the cooperative repair schemes are presented in Sections IV and V for \((h+1) \mid 2^n\) and \((h+1) \nmid 2^n\), respectively. Finally, the concluding remark is drawn in Section VI.

II. PRELIMINARIES

In this section, we briefly review Hadamard MSR codes and Hamming codes. For ease of reading, we firstly introduce some useful notation used throughout this paper.

- Let \(F_q\) be a finite field with \(q\) elements, where \(q\) is a prime power.
- Denote \(a \in F_q^N\) as a vector of length \(N\) over \(F_q\).
- For two non-negative integers \(a\) and \(b\) with \(a < b\), define \([a, b]\) and \([a, b]\) as two sets \(\{a, a+1, \ldots, b-1\}\) and \(\{a, a+1, \ldots, b\}\), respectively.
- For any non-negative integer \(a \in [0, 2^n]\), let \(\rho(a_0, a_1, \ldots, a_{n-1})\) be its binary representation in vector form of length \(n\), i.e., \(a = \sum_{i=0}^{n-1} 2^i a_i, a_i \in \{0, 1\}\). For convenience, we use both of them alternatively with a slight abuse of notation. Since the length \(n\) is determined by the maximum value of \(a\), we always specify the range of \(a\) before using the binary vector representation.
- Let \(e_{n, i}\) be a binary vector of length \(n\) with only the \(i\)-th, \(i \in [0, n]\) component being non-zero.
- Let \(b\) and \(c\) be binary vectors of length \(n\), denote \(b \oplus c = (b_0 \oplus c_0, b_1 \oplus c_1, \ldots, b_{n-1} \oplus c_{n-1})\), where \(\oplus\) is the addition operation modulo 2.

A. Hadamard MSR Codes

Assume that the original data is of size \(M = kN\). An \((n, k)\) MDS code partitions the data into \(k\) parts and then encodes into \(n\) parts \(f = [f_0^T, f_1^T, \ldots, f_{n-1}^T]^T\) stored on \(n\) nodes, where \(f_i = (f_{i,0}, f_{i,1}, \ldots, f_{i,N-1})^T \in F_q^n, i \in [0, n]\) is a column vector and \(^T\) denotes the transpose operator.

Let \(N = 2^n\). The \((n, k)\) Hadamard MSR code is a class of MDS code defined by the following parity-check equations [5], [9]

\[
A_{t,0}f_0 + A_{t,1}f_1 + \cdots + A_{t,n-1}f_{n-1} = 0, \quad t \in [0, r),
\]

where \(A_{t,i}\) is an \(2^n \times 2^n\) nonsingular matrix over \(F_q\), called the parity matrix of node \(i \in [0, n]\) for the \(t\)-th parity-check equation. In matrix form, the structure of \((n, k)\) MSR codes based on the above parity-check equations can be rewritten as

\[
\begin{pmatrix}
A_{0,0} & A_{0,1} & \cdots & A_{0,n-1} \\
A_{1,0} & A_{1,1} & \cdots & A_{1,n-1} \\
\vdots & \vdots & \ddots & \vdots \\
A_{r-1,0} & A_{r-1,1} & \cdots & A_{r-1,n-1}
\end{pmatrix}
\begin{pmatrix}
f_0 \\
f_1 \\
\vdots \\
f_{n-1}
\end{pmatrix} = \mathbf{0}.
\]

block matrix \(A\)

Usually, \(A\) is designed as a block Vandermonde matrix, i.e.,

\[
A_{t,i} = A_i^t, \quad i \in [0, n], \quad t \in [0, r),
\]

where \(A_i, i \in [0, n]\) are \(2^n \times 2^n\) nonsingular matrices. In particular, we use the convention \(A_0^0 = I\).

In fact, Hadamard MSR codes can be characterized by the coding matrices below,

\[
A_i = I_{2^n-1} \otimes \text{blkdiag}(\lambda_{i,0}I_2, \lambda_{i,1}I_2), \quad i \in [0, n),
\]

where \(\otimes\) is the Kronecker product, and \(\lambda_{i,0}, \lambda_{i,1}\) are two distinct elements in \(F_q (q \geq 2n)\).

Accordingly, the \(t\)-th parity-check equation is

\[
\sum_{i=0}^{n-1} \lambda_{i,a}^t f_{i,a} = 0, \quad a \in [0, 2^n], \quad t \in [0, r),
\]

From (2), it is clear that the diagonal element in row \(a \in [0, 2^n]\)-th of \(A_i, i \in [0, n]\) satisfies

\[
\lambda_{i,a} = \lambda_{i,a_i}.
\]

Then, we have the following fact.

**Fact 1**: For any \(0 \leq i, j < n, \lambda_{i,a} \ominus e_{n,j} \neq \lambda_{i,a}\) if \(i = j\) and \(\lambda_{i,a} \ominus e_{n,a} = \lambda_{i,a}\) else.

Besides, the MDS property of Hadamard MSR codes further requires for \(i \neq j \in [0, n], a \neq b \in [0, 2^n]\)

\[
\lambda_{i,a} \neq \lambda_{j,b}.
\]
Example 1: For $N = 2^{14}$, let $\alpha$ be the primitive element of $\mathbb{F}_{29}$. The $(n, k = 2)$ Hadamard MSR code has the following coding matrices over $\mathbb{F}_{29}$

\[
\begin{align*}
A_0 &= \text{diag}(1, -1, 1, -1, 1, -1, \ldots), \\
A_1 &= \alpha \cdot \text{diag}(1, -1, 1, -1, 1, -1, \ldots), \\
A_2 &= \alpha^2 \cdot \text{diag}(1, 1, 1, -1, -1, -1, \ldots), \\
&\vdots \\
A_{13} &= \alpha^{13} \cdot \text{diag}(1, 1, 1, -1, -1, -1, \ldots, 1, -1, 1, -1, \ldots, 1)
\end{align*}
\]

according to (2) and (4).

Remark 1: It is known from [5] that the sub-packetization of an $(n, k)$ Hadamard MSR code is $N = s^n$ with $s = d - k + 1$. Since we focus on the case of $d = k + 1$ helper nodes, only $N = 2^n$ is considered in this paper.

### B. Hamming Codes

**Definition 1 (Perfect Codes [4]):** A $q$-ary $(n', k')$ code $C$ is said to be a perfect code if

\[q^{n'-k'} = \sum_{i=0}^{d'-1} \binom{n'}{i} (q-1)^i,\]

where $d'$ is the minimum Hamming distance of code $C$.

The sets of perfect codes are actually quite limited, where Hamming codes and Golay codes are the only nontrivial ones [4]. To be specific, given an integer $m \geq 2$, the Hamming code $C$ is a family of $(n' = 2^m - 1, k' = 2^m - 1 - m)$ binary codes with single-error-correcting ability, i.e., $d' = 3$.

Let $V_0 = \{e_0, e_1, \ldots, e_{2^n'-1}\}$, where $e_j, 0 \leq j < 2^k$ is the codewords of $C$. Then, based on $V_0$, we can define $n'$ sets

\[V_i = \{c_j \oplus e_{n',i-1} : c_j \in V_0, 0 \leq j < 2^k\}, \quad i \in [1, n'],\]

where $e_{n',i-1}$ is the binary vector of length $n'$ with only the $(i-1)$-th, $i \in [1, n']$ component being non-zero.

The following lemma is a direct consequence of the fact that the binary Hamming code is a perfect code for correcting a single error.

**Lemma 1:** The sets $V_0$ and $V_i, i \in [1, n']$ defined in (6) can cover all the $2^n'$ vectors of length $n'$, i.e.,

\[V_0 \cup V_1 \cup V_2 \cup \cdots \cup V_{n'} = [0, 2^n').\]
\[ f_{j,S}(v), v \in \left[0, \frac{n}{M}\right) \] from all helper nodes \( j \in \mathcal{H} \), node \( i_u \in \mathcal{E} \) can recover
\[ f_{i_u,S}(v) = f_{i,S}(v), \quad v \in \left[0, \frac{n}{M}\right) \] for itself, and
\[ f_{i_u,S}(v) = f_{i,S}(v), \quad v \in \left[0, \frac{n}{M}\right) \] for other failed nodes \( i \in \mathcal{E} \setminus \{i_u\} \).

**Proof:** It follows from (3) that for any \( v \in \left[0, \frac{n}{M}\right) \) and \( t \in [0, r] \),
\[
\lambda^t_{i_u,S}(v) f_{i_u,S}(v) + \sum_{j \in \mathcal{H}} \lambda^t_{i_j,S}(v) f_{i_j,S}(v) + \sum_{z \in \mathcal{U}} \lambda^t_{i_z,S}(v) f_{i_z,S}(v) = 0 \quad (10)
\]
and
\[
\lambda^t_{i_u,S}(v) f_{i_u,S}(v) + \sum_{i \in \mathcal{E} \setminus \{i_u\}} \lambda^t_{i,S}(v) f_{i,S}(v) + \sum_{j \in \mathcal{H}} \lambda^t_{i_j,S}(v) f_{i_j,S}(v) + \sum_{z \in \mathcal{U}} \lambda^t_{i_z,S}(v) f_{i_z,S}(v) = 0 \quad (11)
\]
Substituting (8) and (9) to the summation of (10) and (11), we get
\[
\lambda^t_{i_u,S}(v) f_{i_u,S}(v) + \lambda^t_{i_u,S}(v) f_{i_u,S}(v) + \sum_{j \in \mathcal{H}} \lambda^t_{i_j,S}(v) f_{i_j,S}(v) + \sum_{z \in \mathcal{U}} \lambda^t_{i_z,S}(v) f_{i_z,S}(v) = 0 \quad (12)
\]
For a fixed \( v \in \left[0, \frac{n}{M}\right) \), since we have downloaded \( f_{j,S}(v) + f_{j,S}(v) \) in (7) from all helper nodes \( j \in \mathcal{H} \), the data on the right-hand side of (12) is known. Then, there are \( r \) unknowns \( f_{i_u,S}(v) = f_{i_u,S}(v) \) for nodes 1 and 2.

Following (5) and (8), we can know the entries in the second row of matrix \( A \) satisfy
\[
\lambda_{i_u,S}(v) \neq \lambda_{i_0,S}(v) \neq \cdots \neq \lambda_{i_{h-1},S}(v) \\
\neq \lambda_{i_0,S}(v) \neq \cdots \neq \lambda_{i_{h-2},S}(v).
\]
This means the matrix \( A \) is a Vandermonde matrix, which is invertible. Thus, with \( v \) ranging over \( \left[0, \frac{n}{M}\right) \), node \( i_u \) gets the desired data as claimed.

**Example 3:** Assume that nodes 0, 1, 2 fail in Example 1, and helper nodes and unconnected nodes are [3, 5] and [6, 13], respectively. Take symbols 0 and 1 of node 0 as an example. For \( t \in [0, 11] \), from (3) we know
\[
f_{0,0} + \alpha^t f_{1,0} + \alpha^{2t} f_{2,0} + \sum_{j=3}^{13} \alpha^{jt} f_{j,0} = 0
\]
and
\[
(-1)^t f_{0,1} + \alpha^t f_{1,1} + \alpha^{2t} f_{2,1} + \sum_{j=3}^{13} \alpha^{jt} f_{j,1} = 0,
\]
which gives
\[
f_{0,0} + (-1)^t f_{0,1} + \alpha^t (f_{1,0} + f_{1,1}) + \alpha^{2t} (f_{2,0} + f_{2,1}) \\
+ \sum_{j=6}^{13} \alpha^{jt} (f_{j,0} + f_{j,1}) = 0, \quad (13)
\]
Node 0 downloads \( f_{0,0} + f_{j,1} \) from helper nodes \( j \in [3, 5] \), so the data on the right-hand side of equation (13) is known. We can get the reordered coefficient matrix on the left-hand side
\[
A = \begin{pmatrix}
1 & 1 & 1 & 1 & \cdots & 1 \\
-1 & 1 & \alpha & \alpha^2 & \cdots & \alpha^{13} \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
-1 & 1 & \alpha^{11} & \alpha^{22} & \cdots & \alpha^{10} \\
\end{pmatrix}
\]
Then, the equation (13) has a unique solution because of \( \text{Rank}(A) = 12 \). Hence, node 0 can recover symbols \( f_{0,0} \) and \( f_{0,1} \) for itself and \( f_{1,0} + f_{1,1}, f_{2,0} + f_{2,1} \) for nodes 1 and 2.

Noting that the above repair principle was first proposed in [10], we generalize it here. In [10], \( h + 1 \) instances are generated and then the groups are paired across instances. In our optimal cooperative repair scheme, we firstly form a partition of \( [0, M] \) by \( h + 1 \) sets \( V_0, \cdots, V_h \), i.e.,
\[ P1 \cup \cdots \cup P_h = [0, M] \] and \( V_i \cap V_j = \emptyset \) for any \( 0 \leq i \neq j \leq h \).

Next, for convenience, each failed node chooses two sets \( V_i, V_j \) and pairs each element \( g \in V_i \) with a unique \( g' \in V_j \) such that the conditions (8) and (9) are satisfied. In next two sections, we respectively realize the scheme by means of only Hamming codes for \( h + 1 \) | \( 2^h \), or the combination of Hamming codes and the space sharing technique for \( h + 1 \) | \( 2^k \).

**IV. AN OPTIMAL COOPERATIVE REPAIR SCHEME FOR HADAMARD MSR CODES WITH \( (h+1)2^n \)**

In this section, we propose an optimal cooperative repair scheme for the \( (n, k) \) Hadamard MSR code \( C \) with subpacketization \( N = 2^n \) when the number of failed nodes
satisfies $h \geq 2$ and $(h + 1)2^n$. In this case, we have $h = 2^m - 1$ for a positive integer $2 \leq m \leq \log_2(n - k + 1)$. Let $\mathcal{E} = \{i_0, \ldots, i_{h-1}\}$ and $\mathcal{H} = \{j_0, \ldots, j_k\} \subseteq [0,n]\big\backslash \mathcal{E}$ be the sets of $h$ failed nodes and $k+1$ helper nodes, respectively. Unlike [10], we do not need include any more instances for the original Hadamard MSR code. To acquire P1, we obtain the $h + 1$ sets $V_i$ in (6), $0 \leq i \leq h$, generated by an $(h = 2^m - 1, 2^m - m - 1)$ Hamming code. Next, we pair $g \in V_0$ with a unique $g' \in V_i$ for any given $1 \leq i \leq h$. Note that they are the intra-instance pairing inherited from the original Hadamard MSR code. As a result, we greatly reduce the sub-packetization from $(h + 1)2^n$ in [10] to $2^n$ in this paper.

A. Grouping

Following the repair principle in the last section, we first divide the symbols at each node of a Hadamard MSR code into $M = 2^n$ groups, each group having $2^n-h$ symbols. For a given $g = (g_0, \ldots, g_{h-1}) \in [0,2^h)$, define an ordered indices set as

$$S_g = \{a : a_i = g_u, a = (a_0, \ldots, a_{n-1}) \in [0,2^n), u \in [0,h]\}$$

$$= \{S_g(0), \ldots, S_g(2^n-h-1)\},$$

where

$$(14)$$

$$S_g(0) < S_g(1) < \ldots < S_g(2^n-h-1).$$

Consequently, we have $f_i = (f_i, S_g(v))$ ranging over $[0,2^{n-h}]$, $i \in [0,n)$.

B. Pairing

Define $h + 1$ sets $V_0$ and $V_i, i \in [1,h]$ by applying an $(h = 2^m - 1, 2^m - m - 1)$ Hamming code to (6) in place of $n' = h$ and $k' = 2^m - m - 1$. For $g = (g_0, \ldots, g_{h-1}) \in V_0$, pair it with the index $g' = g \oplus e_{h,u}$ in $V_{u+1}$ to repair failed node $i_u \in \mathcal{E}$.

Lemma 2: Given a failed node $i_u, u \in [0,h)$, for any $g, g' \in [0,2^h)$, conditions (8) and (9) are satisfied if $g' = g \oplus e_{h,u}$.

Proof: For any $v \in [0,2^{n-h})$, let $a = S_g(v) \in [0,2^n), b = S_{g'}(v) \in [0,2^n)$. From (14) and $g' = g \oplus e_{h,u}$, we know

$$a_i = a_{i_u} + 1$$

$$a_i = b_i, \text{ for all } i \in \mathcal{E}\backslash\{i_u\}.$$  

Moreover, according to (15), as the $v$-th element in the ordered set $S_g$ and $S_{g'}$, respectively, the remaining $n-h$ coordinates of $a$ and $b$ should be exactly the same, i.e.,

$$a_i = b_i, \text{ for all } i \in [0,n)\backslash\mathcal{E}. $$

That is, $b = a \oplus e_{n,i_u}$. Then the conclusion follows from Fact 1, which completes the proof.

When $(h + 1)2^n$, the cooperative repair scheme of $h$ failed nodes is given as follows.

C. Download Phase

For any $g \in V_0$ and $g' = g \oplus e_{h,u}$, the failed node $i_u \in \mathcal{E}$ downloads $f_j, S_{g'(v)} + f_j, S_{g''(v)}$ for $v \in [0,2^{n-h}]$ defined in (7) from helper nodes $j \in \mathcal{H}$ to recover

$$(16)$$

$$\{f_i, S_g(v), f_i, S_{g\oplus e_{h,u}}(v) : g \in V_0, v \in [0,2^{n-h})\}$$

for itself and

$$(17)$$

$$\{f_i, S_g(v) + f_i, S_{g\oplus e_{h,u}}(v) : g \in V_0, v \in [0,2^{n-h})\}$$

for other failed nodes $i \in \mathcal{E}\backslash\{i_u\}$ according to Lemma 2 and Theorem 1.

D. Cooperative Phase

Other failed node $i_u, u \in [0,h]\backslash\{u\}$ transfers $\{f_i, S_g(v) + f_i, S_{g\oplus e_{h,u}}(v) : g \in V_0, v \in [0,2^{n-h})\}$ recovered in (17) at download phase to node $i_u$.

Node $i_u$ utilizes its own data $\{f_i, S_g(v), g \in V_0, v \in [0,2^{n-h})\}$ recovered in (16) to solve

$$(18)$$

$$\{f_i, S_g(u), v \in [0,2^{n-h}), g \in V_0, v \in [0,2^{n-h}), u \in [0,h]\backslash\{u\}\}$$

Combing (16) and (18), node $i_u$ obtains

$$\{f_i, S_g(u), v \in [0,2^{n-h}), g \in V_0, u \in [0,h), g \in V_0, v \in [0,2^{n-h})\}. $$

From (6), the set $g \oplus e_{h,i}$ enumerates $V_1, \ldots, V_h$ with $i$ ranging over $[0,h)$. According to Lemma 1,

$$V_0 \cup V_1 \cup V_2 \cup \ldots \cup V_h = [0,2^h).$$

Thus, as $v$ ranging over $[0,2^n-h)$, by (14) we have

$$\bigcup_{g \in V_0, v \in [0,2^{n-h}), i \in [0,h]} S_g(v), S_{g\oplus e_{h,i}}(v) = [0,2^n).$$

(20)

Resulting, associated with (20), equation (19) indicates that all data $\{f_{i_{u,0}}, \ldots, f_{i_{u,2^n-1}}\}$ of the failed node $i_u \in \mathcal{E}$ has been recovered.

During the download phase, each failed node downloads $2^{h-m}, 2^{n-h} = 2^n-m = N/(h+1)$ symbols from each of the $k+1$ helper nodes, i.e., the repair bandwidth at this phase is

$$\gamma_1 = h(k+1) \cdot \frac{N}{h+1}.$$

Next, during the cooperative phase, each failed node accesses $2^{h-m}, 2^{n-h} = 2^n-m = N/(h+1)$ symbols from other $h-1$ failed nodes, i.e., the repair bandwidth at this phase is

$$\gamma_2 = h(h-1) \cdot \frac{N}{h+1}. $$

Totally, the repair bandwidth is

$$\gamma = \gamma_1 + \gamma_2 \cdot \frac{h(k+h) \times N}{h+1},$$
which attains the optimal repair bandwidth according to (1).

This concludes the property of our repair scheme as well as the proof of the following Theorem 2.

**Theorem 2:** When \((h+1)2^n\), any \(h \geq 2\) failed nodes of an \((n, k)\) Hadamard MSR code \(C\) with sub-packetization \(N = 2^n\) can be optimally cooperative repaired with \(d = k + 1\) helper nodes.

**Remark 2:** For \((h + 1)2^n\), the optimal cooperative repair scheme in [10] requires the sub-packetization \((h + 1)N\). In contrast, the sub-packetization in Theorem 2 is \(h + 1 = 2^n\) times less.

**Example 4:** Continued with Example 3. Firstly, divide \(N = 2^{14}\) symbols into \(M = 8\) groups as follows, each group having \(2^{11}\) symbols,

\[
S_0 = \{a : a_0 = 0, a_1 = 0, a_2 = 0, a \in [0, 2^{14}]\},
S_1 = \{a : a_0 = 1, a_1 = 0, a_2 = 0, a \in [0, 2^{14}]\},
S_7 = \{a : a_0 = 1, a_1 = 1, a_2 = 1, a \in [0, 2^{14}]\}.
\]

The binary representation of these indices of groups can be regarded as the codewords of a \((3, 1)\)-Hamming code, so they can be partitioned into \(V_0, V_1, V_2\) and \(V_3\) as given in Example 2. Then, we pair the symbols of groups in \(V_0\) with \(V_1, V_0\) with \(V_2\), and \(V_0\) with \(V_3\) to repair nodes 0, 1 and 2, respectively. Note that all of them are intra-instance pairings.

The repair processes of failed nodes \([0, 1, 2]\) are explained in Figure 1. Tables III and IV illustrate the downloaded and recovered data at download phase and cooperative phase, respectively. The pairing \(\{V_0, V_j\}, j \in [1, 3]\) means that all the groups \(g \in V_0\) are paired with \(g' = g \oplus e_{3,j-1} \in V_j\), one by one, to repair node \(j - 1\).

As a comparison, for repairing \(h = 3\) failed nodes of the \((14, 2)\) Hadamard MSR code in Example 3 with \(d = 3\) helper nodes, Ye’s scheme in [10] needs space sharing \(4\) instances, which leads to a sub-packetization of \(4 \times 2^{14}\), whereas the new scheme can repair the original \((14, 2)\) Hadamard MSR code with sub-packetization of \(2^{14}\) owing to the intra-instance pairing.

**V. AN OPTIMAL COOPERATIVE REPAIR SCHEME FOR HADAMARD MSR CODES WITH \((h + 1) \mid 2^n\)**

In this section, we propose an optimal cooperative repair scheme for the \((n, k)\) Hadamard MSR code \(C\) with sub-p!-packetization \(N = (2\ell + 1)2^n\), \(\ell \geq 1\) when the number of failed nodes satisfies \(h \geq 2\) and \((h + 1) \mid 2^n\). Set \(h = (2\ell + 1)2^{m-1} - 1 = n - k\) for some integers \(\ell \geq 1\), \(m \geq 0\) and let \(h' = 2^m - 1\). Let \(E = \{i_0, \ldots, i_{h-1}\}\) and \(H = \{j_0, \ldots, j_k\} \subseteq [0, n] \setminus E\) be the sets of \(h\) failed nodes and \(k + 1\) helper nodes, respectively.

As mentioned at the repair principle in Section III, \(N\) is divided by \(M\). Moreover, by \(P_1\), \(h + 1\) should be a factor of \(M\). That is, we have \((h + 1) \mid N\). Then, for \((h + 1) \mid 2^n\), we can not get such \(h + 1\) sets \(V_i, i \in [0, h]\) in \(P_1\) for the original Hadamard MSR code with \(N = 2^n\). Thus, we have to extend the Hadamard MSR code by space sharing \(2\ell + 1\) instances and divide the \(N = (2\ell + 1)2^n\) symbols into \(M = (2\ell + 1)2^h\) groups. Then, we can form a partition \((i, V_j), i \in [0, 2\ell]\) of \([0, M]\) satisfying \(P_1\), based on the sets \(V_j, 0 \leq j \leq h', \) generated by an \((h' + 1)2^{m-1} - 1, 2^m - m - 1\)
where i

Hamming code. Next, we pair (0, V_j) with some (i, V_j'), 0 ≤ j, j' ≤ h', which are either the intra-instance pairing when i = 0 or the inter-instance pairing when i ≠ 0. By means of both the intra-instance pairing and inter-instance pairing, we are able to greatly reduce the sub-packetization from (h+1)2^n in [10] to (2^l + 1)2^n in this paper.

A. An (n, k) Hadamard MSR Code C With Sub-Packetization N = (2^l + 1)2^n

In the original (n, k) Hadamard MSR code C with sub-packetization 2^n, each node i ∈ [0, n) stores a column vector f_i = (f_i,0, f_i,1, ..., f_i,2^n-1) of length 2^n. Generate 2^l + 1 instances of C, whose column vectors are denoted by f_i(0), ..., f_i(2^l), 0 ≤ i < n. We obtain the desired (n, k) Hadamard MSR code C with sub-packetization N = (2^l + 1)2^n. By convenience, still denote the code by C and write the column vector of length (2^l + 1)2^n stored at node i as f_i = ((f_i(0))^T, ..., (f_i(2^l))^T)^T, i ∈ [0, n).

1) Grouping: Recall from Section IV that the 2^n symbols of f_i(w), w ∈ [0, 2^l] are divided into 2^h' groups, where h' = 2^m - 1, each group having 2^n-h' symbols. That means the data of each nodes is divided into (2^l + 1)2^h' groups. Similar to (14), for g ∈ [0, 2^h'), define S_g

| Failed nodes | 0     | 1     | 2     |
|--------------|-------|-------|-------|
| Pairing      | (V_0, V_1) | (V_0, V_2) | (V_0, V_3) |
| Download     | f_i, S_0(v) + f_i, S_2(v) | f_i, S_0(v) + f_i, S_2(v) | f_i, S_0(v) + f_i, S_2(v) |
| Repair       | f_i, S_0(v) + f_i, S_2(v) | f_i, S_0(v) + f_i, S_2(v) | f_i, S_0(v) + f_i, S_2(v) |

Then, the parity-check equation (3) of C with sub-packetization N = (2^l + 1)2^n can be rewritten as

\[
\sum_{i=0}^{n-1} \lambda_i s_{g}(v) f_{i, S_{i}(v)} = 0, \quad w \in [0, 2^l], \ g \in [0, 2^{h'}], \ v \in [0, 2^{n-h'}], \ t \in [0, r].
\] (23)

B. Pairing

In this subsection, we fix the failed node i_u, u = (2^l + 1)u_1 + u_2 ∈ [0, h), u_1 ∈ [0, h'), u_2 ∈ [0, 2^l]. As the data of each node is divided into (2^l + 1)2^h' groups, we employ indices (w, g), w ∈ [0, 2^l], g ∈ [0, 2^h') to denote these groups, and denote the v-th symbol by S_{w,g}(v), where v ∈ [0, 2^{n-h'}]. Naturally, the set S_g in (21) is defined for w = 0, i.e. S_0,g = S_g. Directly, for any w ∈ [0, 2^l], we define ordered set

S_{w,g} = S_g, \quad g \in [0, 2^{h'})

with a default order

S_{w,g}(v) \triangleq S_g(v), \quad g \in [0, 2^{h'}), \ v \in [0, 2^{n-h'}].

(24)

Similarly, we need to pair these groups.

1) Pairing: In order to make (8) and (9) valid, we have to reorder some groups. For the sake of simplification, the v-th symbol is still denoted as S_{w,g}(v). Here we highlight that the reorder is related with the index of node i_u.

- When 0 ≤ u_1 < h', u_2 = 0, the same as the pairing method in section IV, pair the group (0, g) ∈ {0} × V_0 with group (0, g ⊕ e_{w_0} u_1) ∈ {0} × V_{n+1}. In this case, we use the default order of the latter, i.e.

S_{0,g} \circ e_{w_0,u_1}(v) \triangleq S_g \circ e_{w_0,u_1}(v).

(25)
• When $0 \leq u_1 < h'$, $0 < u_2 \leq 2\ell$, pair the group $(0, g) \in \{0\} \times V_{u_1}$ with group $(u_2, g) \in \{u_2\} \times V_{u_1}$, and reorder the elements of $S_{u_2,g}$ such that

$$S_{u_2,g}(v) \triangleq S_g(v) \oplus e_{n_i,u}, \quad g \in V_{u_1}.$$  

(26)

According to (21), the symbols of $S_g$ is determined by components with indices $i_{(2\ell+1)u_1}$, $u_1 \in [0,h')$, i.e. $u_2 = 0$. In this case, $u_2 \neq 0$, we have $\{S_g(v) \oplus e_{n_i,u} : v \in \{0,2^{n-h'}\}\} = S_{g'}, g \in V_{u_1}$, which is only a reordering of $S_{u_2,g}$ without adding or losing any elements.

• When $u_1 = h', 0 \leq u_2 < 2\ell$, pair the group $(0, g) \in \{0\} \times V_{h'}$ with group $(u_2+1, g) \in \{u_2+1\} \times V_{h'}$, and reorder the elements of $S_{u_2+1,g}$ such that

$$S_{u_2+1,g}(v) \triangleq S_g(v) \oplus e_{n_i,u}, \quad g \in V_{h'}.$$  

(27)

Similarly, the equation (27) is only a reordering of $S_{u_2+1,g}$ since $u_2+1 \neq 0$ and $\{S_g(v) \oplus e_{n_i,u} : v \in \{0,2^{n-h'}\}\} = S_{g'}, g \in V_{h'}$ by (21).

Note that $V_1, i \in [0,h')$ is the same as those sets defined in section III such that $V_0 \cup V_1 \cup V_2 \cup \cdots \cup V_{h'} = \{0,2^{h'}\}$

(28)

due to Lemma 1.

Lemma 3: Given a failed node $i_u, u \in [0,h)$, for any $w \in [0,2\ell], g \in [0,2^{h'}]$, the pairing method above makes (8) and (9) hold.

Proof: During the pairing process above, suppose that the group $(w, g)$ is paired with group $(w', g')$. For any $v \in [0,2^{n-h'})$, let $a = S_w(v), \quad b = S_{w'}(g')$.

When $0 \leq u_1 < h', u_2 = 0$, and $u = (2\ell + 1)u_1$, we have $w' = w = 0$ and $g' = g + e_{n_i,u}$. According to (21) and (22), we have $a = a + e_{n_i,u}$ by the same argument used in Lemma 2. Thus, equations (8) and (9) follow from Fact 1.

Next, we prove the other two cases, i.e., $0 \leq u_1 < h', 0 < u_2 \leq 2\ell$ or $u_1 = h', 0 \leq u_2 < 2\ell$. Note that for the set $S_{w=0,g}$ we use the default order, i.e., $\{S_g(v) : v \in \{0,2^{n-h'}\}\}$, then, based on (26) and (27), $b = a \oplus e_{n_{i_u}}$ still holds for both of the remaining two cases.

Thus, we conclude that conditions (8) and (9) hold by Fact 1, which completes the proof.

C. A Repair Scheme

When $(h+1) \uparrow 2^n$, the cooperative repair process of $h$ failed nodes is as follows.

1) Download Phase: For failed node $i_u, u = (2\ell + 1)u_1 + u_2 \in E$, $u_1 \in [0,h'), u_2 \in [0,2\ell]$ or $u_1 = h', u_2 \in [0,2\ell]$, the downloaded phase is considered by three different cases. Based on Lemma 3 and Theorem 1,

• Case 1. When $0 \leq u_1 < h', u_2 = 0$, node $i_u$ downloads

$$\{f^{(0)}(i_{(2\ell+1)u_1}, v) + f^{(0)}(i_{u_1}, v) : v \in \{0,2^{n-h'}\}\}$$  

(29)

from helper nodes $j \in \mathcal{H}$ to recover

$$\{f^{(0)}(i_{(2\ell+1)u_1}, v) + f^{(0)}(i_{u_1}, v) : v \in \{0,2^{n-h'}\}\}$$  

for itself and

$$\{f^{(0)}(i_{(2\ell+1)u_1}, v) + f^{(0)}(i_{u_1}, v) : v \in \{0,2^{n-h'}\}\}$$  

(30)

for other failed nodes $i \in \mathcal{E}\setminus\{i_u\}$.

• Case 2. When $0 \leq u_1 < h', 0 < u_2 \leq 2\ell$, node $i_u$ downloads

$$\{f^{(0)}(i_{(2\ell+1)u_1}, v) + f^{(0)}(i_{u_1}, v) : v \in \{0,2^{n-h'}\}\}$$  

(31)

from helper nodes $j \in \mathcal{H}$ to recover

$$\{f^{(0)}(i_{(2\ell+1)u_1}, v) + f^{(0)}(i_{u_1}, v) : v \in \{0,2^{n-h'}\}\}$$  

for itself and

$$\{f^{(0)}(i_{(2\ell+1)u_1}, v) + f^{(0)}(i_{u_1}, v) : v \in \{0,2^{n-h'}\}\}$$  

(32)

for other failed nodes $i \in \mathcal{E}\setminus\{i_u\}$.

• Case 3. When $u_1 = h', 0 \leq u_2 < 2\ell$, node $i_u$ downloads

$$\{f^{(0)}(i_{(2\ell+1)u_1}, v) + f^{(0)}(i_{u_1}, v) : v \in \{0,2^{n-h'}\}\}$$  

(33)

from helper nodes $j \in \mathcal{H}$ to recover

$$\{f^{(0)}(i_{(2\ell+1)u_1}, v) + f^{(0)}(i_{u_1}, v) : v \in \{0,2^{n-h'}\}\}$$  

for itself and

$$\{f^{(0)}(i_{(2\ell+1)u_1}, v) + f^{(0)}(i_{u_1}, v) : v \in \{0,2^{n-h'}\}\}$$  

(34)

for other failed nodes $i \in \mathcal{E}\setminus\{i_u\}$.

2) Cooperative Phase: Other failed nodes $i_u, \bar{u} = (2\ell + 1)\bar{u}_1 + \bar{u}_2 \in \mathcal{E}\setminus\{i_u\}, \bar{u}_1 \in [0,h'), \bar{u}_2 \in [0,2\ell]$ or $\bar{u}_1 = h', \bar{u}_2 \in [0,2\ell]$ transfers

$$\{f^{(0)}(i_{(2\ell+1)u_1}, v) + f^{(0)}(i_{u_1}, v) : v \in \{0,2^{n-h'}\}\},$$  

(35)

$$\{f^{(0)}(i_{(2\ell+1)u_1}, v) + f^{(0)}(i_{u_1}, v) : v \in \{0,2^{n-h'}\}\},$$  

(36)

$$\{f^{(0)}(i_{(2\ell+1)u_1}, v) + f^{(0)}(i_{u_1}, v) : v \in \{0,2^{n-h'}\}\},$$  

(37)

by repaired by (30), (32) or (34) to node $i_u$.

• Case 1. When $0 \leq u_1 < h', u_2 = 0$, the failed node $i_u$ utilizes its own data $\{f^{(0)}(i_{(2\ell+1)u_1}, v) : v \in \{0,2^{n-h'}\}\}$ to solve

$$\mathcal{F}_1 = \{f^{(0)}(i_{(2\ell+1)u_1}, v) : v \in \{0,2^{n-h'}\},$$  

$$\bar{u}_1 \in [0,h') \setminus \{u_1\}\},$$  

$$\mathcal{F}_2 = \{f^{(0)}(i_{u_1}, v) : v \in \{0,2^{n-h'}\},$$  

$$\bar{u}_1 \in [0,h'), \bar{u}_2 \in [1,2\ell]\},$$  

$$\mathcal{F}_3 = \{f^{(0)}(i_{(2\ell+1)u_1}, v) : v \in \{0,2^{n-h'}\},$$  

$$\bar{u}_2 \in [0,2\ell]\},$$  

from the data in (35), (36) and (37), respectively.

Firstly, by (28) noting that

$$\{g, g \oplus e_{h',u_1}, g \oplus e_{h',u_1} : v \in V_0, \bar{u}_1 \in [0,h') \setminus \{u_1\}\} = \cup_{i \in \mathcal{E}} V_i = \{0,2^{h'}\},$$  

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
TABLE V
PAIRING METHOD FOR h = 11 FAILED NODES

| Node u | 0 | 1 | 2 | 3 | 4 | 5 |
|--------|---|---|---|---|---|---|
| Pairing | {0} x V₀, {0} x V₁ | {0} x V₀, {1} x V₀ | {0} x V₀, {2} x V₀ | {0} x V₁, {1} x V₁ | {0} x V₁, {2} x V₁ |

| Node u | 6 | 7 | 8 | 9 | 10 |
|--------|---|---|---|---|---|
| Pairing | {0} x V₀, {0} x V₅ | {0} x V₀, {1} x V₂ | {0} x V₂, {2} x V₂ | {0} x V₂, {1} x V₃ | {0} x V₅, {2} x V₃ |

TABLE VI
REORDER OF THE SYMBOLS OF NODE i ∈ [0, 10) \ {0, 3, 6}, WHERE S₀ ⊕ eₙ,u = {S₀(v) ⊕ eₙ,u : v ∈ [0, 2¹¹]}

| Node u | 1 | 2 | 4 | 5 | 7 |
|--------|---|---|---|---|---|
| Reorder | S₁,₀ = S₂ ⊕ e₁₄,₁ S₂,₀ = S₂ ⊕ e₁₄,₂ S₁,₁ = S₁ ⊕ e₁₄,₁ S₁,₂ = S₁ ⊕ e₁₄,₅ S₂,₁ = S₂ ⊕ e₁₄,₅ | S₂,₂ = S₂ ⊕ e₁₄,₅ S₁,₅ = S₅ ⊕ e₁₄,₇ S₂,₂ = S₂ ⊕ e₁₄,₈ S₂,₅ = S₅ ⊕ e₁₄,₈ |

| Node u | 9 | 10 |
|--------|---|---|
| Reorder | S₁,₄ = S₅ ⊕ e₁₄,₉ S₂,₄ = S₄ ⊕ e₁₄,₁₀ S₂,₅ = S₅ ⊕ e₁₄,₈ |

thus,

F_1 ∪ \{ f_{i_u,S_{0,g}(v)}^{(0)}(v, g) : g ∈ V₀, v ∈ [0, 2ⁿ⁻h') \}

contains all symbols of f_{i_u}^{(0)}.

Secondly, for a fixed 0 ≤ u₂ ≤ 2ℓ, we can recover

\{ f_{i_u,(u₂,g)}^{(0)}(v, g) : g ∈ V_u, v ∈ [0, 2ⁿ⁻h') \}

from F₂ with u₁ ranging over [h', h']. Combining it with F₃, we obtain all symbols of f_{i_u}^{(0)}, u₂ ∈ [1, 2ℓ] by (28).

The preceding two steps have recovered all symbols of f_{i_u}^{(0)}, w ∈ [0, 2ℓ].

- Cases 2 and 3. When 0 ≤ u₁ < h', 0 ≤ u₂ ≤ 2ℓ or u₁ = h', 0 ≤ u₂ < 2ℓ or if u₁ > 0, the failed node i_u utilizes its own data \{ f_{i_u,S_{0,g}(v)}^{(0)}(v, g) : g ∈ V_u, v ∈ [0, 2ⁿ⁻h') \} to solve

\{ f_{i_u,S_{0,g}(v)}^{(0)}(v, g) : g ∈ V_u, v ∈ [0, 2ⁿ⁻h') \}

from the data in (35) transferred by failed nodes i_u, u = (2ℓ + 1)u₁ in Case 1.

During the downloading phase, by (29), (31) and (33), each failed node downloads 2ⁿ⁻m · 2ⁿ⁻h' = 2ⁿ⁻m = N/(h + 1) symbols from each of the k + 1 helper nodes, i.e., the repair bandwidth at this phase is

γ₁ = h(h + 1) · N/h + 1.

Next, during the cooperative phase, by (35), (36) and (37), each failed node accesses 2ⁿ⁻m · 2ⁿ⁻h' = 2ⁿ⁻m = N/(h + 1) symbols from other h - 1 failed nodes, i.e., the repair bandwidth at this phase is

γ₂ = h(h + 1) · N/h + 1.

Totally, the repair bandwidth is

γ = γ₁ + γ₂ = h(h + 1) · N/h + 1,

which attains the optimal repair bandwidth according to (1).

This concludes the property of our repair scheme as well as the proof of the following Theorem 3.

Theorem 3: When (h + 1) | 2ⁿ where h + 1 = (2ℓ + 1)2ᵐ, m ≥ 0, ℓ ≥ 1, any h ≥ 2 failed nodes of an (n, k) Hadamard MSR code C with sub-packetization N = (2ℓ + 1)2ᵐ can be optimally cooperative repaired with \(d = k + 1\) helper nodes.

Remark 3: For (h + 1) | 2ⁿ, the optimal cooperative repair scheme in [10] requires the sub-packetization \(h + 1)N = (2ℓ + 1)2ⁿ⁺⁺⁺⁺. In contrast, the sub-packetization in Theorem 3 is 2²ᵐ times less.

Example 3: Based on the (n = 14, k = 2) Hadamard MSR code with sub-packetization N = 2¹⁴ in Example 1, we obtain a new (n = 14, k = 2) Hadamard MSR code C with sub-packetization N = 3 × 2¹⁴ by generating 3 instances. At this point, m = 2, ℓ = 1, h' = 3, the number of failed nodes is h = 11. Assume that the failed nodes are the first 11 nodes, i.e., \{i₀, i₁, ⋯, i₁₀\} = \{0, 1, ⋯, 10\}, and the remaining 3 nodes are helper nodes.

Firstly, divide the 2¹⁴ symbols of the base code into 8 groups based on (21), each group having 2¹¹ symbols. Then 3 instances are divided into M = 3 × 8 groups. For any failed node i_u ∈ E, the pairing method is shown at Table V, where \{i\} x V_j is the group (i, g), g ∈ V_j. From Table V, we can see that the pairing method of nodes 0, 3 and 6 is implemented with the intra-instance pairing, i.e., inside the instance 0, while the pairing method of other nodes is by means of the inter-instance pairing across the instances.

In Table V, to repair nodes 1, 2, 4, 5, 7 and 8, we reorder the symbols of (w, g) by (26) and nodes 9 and 10 by (27). The reorder processes for these nodes are shown in Table VI.

When u₂ = 0, the data of nodes 0, 3 and 6 can be paired and recovered following the previous section. We will only illustrate the repair process of the failed nodes when u₂ > 0.

Take symbols S₀(1) = 0 and S₁(1) = 2 of node 1 as an example, applying conditions (8), (9) and equation (23), we have

\[ λ₁,₁₀f₁,₀^{(0)} + λ₁,₁₂f₁,₂^{(1)} + \sum_{i ≠ 1} (λ₁,₁₀f₁,₀^{(0)} + λ₁,₁₂f₁,₂^{(1)}) = 0, \]

\[ t ∈ [0, 11]. \]  (38)

From (4),

\[ λ₁,₁₀ ≠ λ₁,₁₂, λ₁,₀ = λ₁,₁₂, i ≠ 1. \]

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
There are 12 unknowns $f_{i,0}^{(0)}, f_{i,2}^{(1)}, f_{i,1}^{(0)} + f_{i,2}^{(1)}, i \in \{0\} \cup [2, 10]$ in equation (38). With $t$ enumerating all elements in $[0, 11]$, the equation has a unique solution, so the desired data could be solved.

As a comparison, for repairing $h = 11$ failed nodes of the $(14, 2)$ Hadamard MSR code in Example 1 with $d = 3$ helper nodes, Ye’s scheme in [10] needs space sharing 12 instances, which results in the sub-packetization of $12 \times 2^{14}$, whereas we need space sharing only 3 instances, which leads to sub-packetization of $3 \times 2^{14}$, owing to the usage of both the intra-instance pairing and inter-instance pairing.

VI. Conclusion

In this paper, a new cooperative repair scheme was proposed for the $(n, k)$ Hadamard MSR code with $h$ node failures, which downloads the data from $d = k + 1$ helper nodes. Particularly, the cooperative repair scheme is feasible for sub-packetization $2^n$ when $(h + 1)2^n$ and $(2^h + 1)2^m$ when $h + 1 = (2^h + 1)2^m$ for $m \geq 0$ and $h \geq 1$. In contrast to the known best result in [10], the sub-packetization is greatly decreased.

Unfortunately, our scheme only works for $d = k + 1$ helper nodes to recover the failed data, and cannot be generalized to $k + 2 \leq d \leq n - h$. Let us fix $d = k + 2$ as an illustrated example. In [10], the failed node $i \in [0, h)$ chooses instances $0, 1, i, i + 2$ and then pairs the corresponding symbols. However, it seems no way to find out the elements $g_1, g_2, g_3$ and $g_4$ in the same instance of Hadamard MSR codes such that two pairs indexed by $(g_1, g_2, g_3)$ and $(g_1, g_2, g_4)$ simultaneously satisfy properties similar to (8) and (9). That is, no desired intra-instance pairing method exists. Therefore, our next task is to find a new cooperative repair scheme for any $d$ helper nodes with small sub-packetization.

ACKNOWLEDGMENT

The authors would like to thank the Associate Editor, Prof. Xiaode Zhang, and the anonymous reviewers, whose comments and suggestions improved the presentation of this paper.

REFERENCES

[1] S. B. Balaji, M. N. Krishnan, M. Vajha, V. Ramkumar, B. Sasidharan, and P. V. Kumar, “Erasure coding for distributed storage: An overview,” Sci. China Inf. Sci., vol. 61, no. 10, pp. 1-45, Oct. 2018.
[2] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, “Network coding for distributed storage systems,” IEEE Trans. Inf. Theory, vol. 56, no. 9, pp. 4539-4551, Sep. 2010.
[3] Y. Hu, Y. Xu, X. Wang, C. Zhan, and P. Li, “Cooperative recovery of distributed storage systems from multiple losses with network coding,” IEEE J. Sel. Areas Commun., vol. 28, no. 2, pp. 268-276, Feb. 2010.
[4] T. K. Moon, Error Correction Coding: Mathematical Methods and Algorithms. Hoboken, NJ, USA: Wiley, 2005.
[5] D. S. Papailiopoulos, A. G. Dimakis, and V. R. Cadambe, “Repair optimal erasure codes through Hadamard designs,” in Proc. 49th Annu. Allerton Conf. Commun., Control, Comput. (Allerton), Sep. 2011, pp. 1382-1389.
[6] K. Rashmi, N. B. Shah, and K. Ramchandran, “A piggybacking design framework for read-and download-efficient distributed storage codes,” IEEE Trans. Inf. Theory, vol. 63, no. 9, pp. 5802-5820, Sep. 2017.
[7] I. S. Reed and G. Solomon, “Polynomial codes over certain finite fields,” J. Soc. Ind. Appl. Math., vol. 8, no. 2, pp. 300-304, Jun. 1960.
[8] K. W. Shum and Y. Hu, “Cooperative regenerating codes,” IEEE Trans. Inf. Theory, vol. 59, no. 11, pp. 7229-7259, Nov. 2013.
[9] X. Tang, B. Yang, J. Li, and H. D. L. Hollmann, “A new repair strategy for the Hadamard minimum storage regenerating codes for distributed storage systems,” IEEE Trans. Inf. Theory, vol. 61, no. 10, pp. 5271-5279, Oct. 2015.
[10] M. Ye, “New constructions of cooperative MSR Codes: Reducing node size to $\exp(O(n))$,” IEEE Trans. Inf. Theory, vol. 66, no. 12, pp. 7457-7464, Dec. 2020.
[11] M. Ye and A. Barg, “Cooperative repair: Constructions of optimal MDS codes for all admissible parameters,” IEEE Trans. Inf. Theory, vol. 65, no. 3, pp. 1639-1656, Mar. 2019.
[12] Y. Zhang, Z. Zhang, and L. Wang, “Explicit constructions of optimal-access MSCR codes for all parameters,” IEEE Commun. Lett., vol. 24, no. 5, pp. 941-945, May 2020.

Yajuan Liu received the B.S. degree in mathematics and applied mathematics from Northwest Normal University, Lanzhou, China, in 2018. She is currently pursuing the Ph.D. degree in information security with Southwest Jiaotong University. Her research interests include coding for the distributed storage and DNA storage.

Han Cai (Member, IEEE) received the B.S. and M.S. degrees in mathematics from Hubei University, Wuhan, China, in 2009 and 2013, respectively, and the Ph.D. degree from the Department of Communication Engineering, Southwest Jiaotong University, Chengdu, China, in 2017. From October 2015 to October 2017, he was a Visiting Ph.D. Student with the Faculty of Engineering, Information and Systems, University of Tsukuba, Japan. From 2018 to 2021, he was a Post-Doctoral Fellow with the School of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Israel. In 2021, he joined Southwest Jiaotong University, where he currently holds a tenure-track position. His research interests include coding theory and sequence design.

Xiaohu Tang (Senior Member, IEEE) received the B.S. degree in applied mathematics from Northwest Polytechnic University, Xi’an, China, in 1992, the M.S. degree in applied mathematics from Sichuan University, Chengdu, China, in 1995, and the Ph.D. degree in electronic engineering from Southwest Jiaotong University, Chengdu, in 2001. From 2003 to 2004, he was a Research Associate with the Department of Electrical and Electronic Engineering, The Hong Kong University of Science and Technology. From 2007 to 2008, he was a Visiting Professor with the University of Ulm, Germany. Since 2001, he has been with the School of Information Science and Technology, Southwest Jiaotong University, where he is currently a Professor. His research interests include coding theory, network security, distributed storage, and information processing for big data. Dr. Tang was a recipient of the National Excellent Doctoral Dissertation Award, China, in 2003; the Humboldt Research Fellowship, Germany, in 2007; and the Outstanding Young Scientist Award by NSFC, China, in 2013. He served on a number of technical program committees for conferences. He served as an Associate Editor for several journals, including IEEE TRANSACTIONS ON INFORMATION THEORY and IEEE TRANSACTIONS ON FUNDAMENTALS.