Toward Optimality in Both Repair and Update via Generic MDS Code Transformation

Hanxu Hou[†], Patrick P. C. Lee[§], and Yunghsiang S. Han[†]
[†] School of Electrical Engineering & Intelligentization, Dongguan University of Technology
[§] Department of Computer Science and Engineering, The Chinese University of Hong Kong

***Abstract*— An (n,k) maximum distance separable (MDS) code encodes $k\alpha$ data symbols into $n\alpha$ symbols that are stored in n nodes with α symbols each, such that the $k\alpha$ data symbols can be reconstructed from any k out of n nodes. MDS codes achieve optimal repair access if we can repair the lost symbols of any single node by accessing $\frac{\alpha}{d-k+1}$ symbols from each of d other surviving nodes, where $k+1 \le d \le n-1$. In this paper, we propose a generic transformation for any MDS code to achieve optimal repair access for a single-node repair among $d-k+1$ nodes, while the transformed MDS codes maintain the same update bandwidth (i.e., the total amount of symbols transferred for updating the symbols of affected nodes when some data symbols are updated) as that of the underlying MDS codes. By recursively applying our transformation for existing MDS codes with the minimum update bandwidth, we can obtain multi-layer transformed MDS codes that achieve both optimal repair access for any single-node repair among all n nodes and minimum update bandwidth.**

***Index Terms*—MDS codes, minimum update bandwidth, optimal repair access.**

I. Introduction

Maximum distance separable (MDS) codes are a class of erasure codes that are widely employed in distributed storage systems to provide data reliability. Reed-Solomon (RS) codes [1] are one well-known example of MDS codes. An (n,k) MDS code (where $k \le n$) encodes a data file of $k\alpha$ data symbols (where $\alpha \ge 1$) over the finite field $\mathbb{F}_q$ into $n\alpha$ coded symbols that are distributed across n storage nodes with α symbols each, such that the original $k\alpha$ data symbols can be reconstructed from any k out of n nodes (called the *MDS property*). The number of symbols stored in each node, α, is called the *sub-packetization level*. Dimakis *et al.* [2] show that we can repair the α lost symbols of any failed node by downloading at least

$$\beta = \frac{\alpha}{d-k+1}$$

symbols from each of d helper nodes (where $k+1 \le d \le n-1$). Also, the *repair bandwidth* (i.e., the total amount of symbols downloaded for repairing a single failed node of any (n,k) MDS code) is at least

$$\gamma = d\beta = \frac{d\alpha}{d-k+1}. \tag{1}$$

This work was partially supported by the National Natural Science Foundation of China (No. 61701115, 61671007), Research Grants Council of Hong Kong (GRF 14216316 and AoE/P-404/18), the Quality Engineering Project of 2018 DGUT under Grant 201802016, the Project of 2019 Guangdong Province Teaching Quality and Teaching Reform (Measurement and Control Technology Teaching Team No. 75).

We focus on a class of MDS codes called the *MDS array codes*, in which the code structure is arranged as an $\alpha \times n$ array. We define *repair access* as the total number of symbols accessed from d helper nodes when repairing a single failed node. An MDS array code achieves *optimal repair access* if the repair access equals the minimum repair bandwidth in Eq. (1). Some constructions of MDS array codes with the minimum repair bandwidth are given in [3]–[12].

In addition to the repair problem, another important issue in distributed storage systems is the *update problem*. Specifically, if some data symbols need to be updated to the new version, we need to send symbols to the nodes whose symbols need to be updated. When we update some data symbols, we need to send symbols to the corresponding nodes to update the symbols there. Here, we consider *vertical* MDS array codes, in which coded symbols are arranged in the rows of the array (i.e., the coded symbols span across all nodes), mainly because they achieve optimal update [13]. Table I shows a vertical MDS array code of $k=2$, $n=4$, and $\alpha=4$ [13]. The eight data symbols are $s_{i,j} \in \mathbb{F}_q$, where $i=1,2$ and $j=1,2,3,4$. First, we can decode the eight data symbols from any two nodes if $q>2$. For example, consider nodes 1 and 3. We can subtract $s_{1,3}$ each from $s_{1,2}+s_{1,3}+s_{1,4}$ and $s_{1,2}+2s_{1,3}+3s_{1,4}$ in node 1 to obtain $s_{1,2}+s_{1,4}$ and $s_{1,2}+3s_{1,4}$, respectively, and further solve for $s_{1,2}$ and $s_{1,4}$. The decoding of data symbols from any other two nodes is similar.

If we update the two data symbols of a node, we only need to send two updated data symbols to the node plus one coded symbol to each of the other three nodes to update the corresponding symbols. For example, if the two data symbols $s_{1,1}$ and $s_{2,1}$ are updated into $\bar{s}_{1,1}$ and $\bar{s}_{2,1}$, respectively, then we only need to send two symbols $\bar{s}_{1,1}, \bar{s}_{2,1}$ to node 1, and three coded symbols $\bar{s}_{1,1}-s_{1,1}$, $\bar{s}_{2,1}-s_{2,1}$ and $(\bar{s}_{1,1}-s_{1,1})+(\bar{s}_{2,1}-s_{2,1})$ to nodes 2, 3 and 4, respectively. The total amount of symbols transferred for the update is called the *update bandwidth* [13]. In the (n,k) vertical MDS array codes of size $n \times n$, each node stores k data symbols and $n-k$ coded symbols. If the k data symbols stored in one node should be updated, it is shown in [13] that the minimum update bandwidth of the (n,k) vertical MDS array codes is $n+k-1$. The update bandwidth of updating two data symbols $s_{1,j}$ and $s_{2,j}$ of the code in Table I is five, which is optimal. However, the repair access of the code in Table I is sub-optimal.

We show via an example how our MDS code transformation (detailed in Section II) makes optimal repair access

TABLE I: Example of the code in [13] with $k = 2$ and $n = 4$.

Node 1	Node 2	Node 3	Node 4
$s_{1,1}$	$s_{1,2}$	$s_{1,3}$	$s_{1,4}$
$s_{2,1}$	$s_{2,2}$	$s_{2,3}$	$s_{2,4}$
$s_{1,2}+s_{1,3}+s_{1,4}$	$s_{1,1}+s_{2,3}+s_{2,4}$	$s_{2,1}+s_{2,2}+(s_{1,4}+s_{2,4})$	$(s_{1,1}+s_{2,1})+(s_{1,2}+s_{2,2})+(s_{1,3}+s_{2,3})$
$s_{1,2}+2s_{1,3}+3s_{1,4}$	$s_{1,1}+2s_{2,3}+3s_{2,4}$	$s_{2,1}+2s_{2,2}+3(s_{1,4}+s_{2,4})$	$(s_{1,1}+s_{2,1})+2(s_{1,2}+s_{2,2})+3(s_{1,3}+s_{2,3})$

TABLE II: The transformed code by applying the transformation with $e = -1$ for the first two nodes of the code in Table I. Note that the symbols in rows 5-8 (the symbols labeled with $\star$) in nodes 2-4 can repair node 1.

Row	Node 1	Node 2	Node 3	Node 4
1	$s^1_{1,1}+s^2_{1,1}$	$s^1_{1,2}$	$s^1_{1,3}$	$s^1_{1,4}$
2	$s^1_{2,1}+s^2_{2,1}$	$s^1_{2,2}$	$s^1_{2,3}$	$s^1_{2,4}$
3	$(s^1_{1,2}+s^1_{1,3}+s^1_{1,4})+$ $(s^2_{1,2}+s^2_{1,3}+s^2_{1,4})$	$s^1_{1,1}+s^1_{2,3}$ $+s^1_{2,4}$	$s^1_{2,1}+s^1_{2,2}+$ $(s^1_{1,4}+s^1_{2,4})$	$(s^1_{1,1}+s^1_{2,1})+(s^1_{1,2}+s^1_{2,2})$ $+(s^1_{1,3}+s^1_{2,3})$
4	$(s^1_{1,2}+2s^1_{1,3}+3s^1_{1,4})+$ $(s^2_{1,2}+2s^2_{1,3}+3s^2_{1,4})$	$s^1_{1,1}+2s^1_{2,3}$ $+3s^1_{2,4}$	$s^1_{2,1}+2s^1_{2,2}+$ $3(s^1_{1,4}+s^1_{2,4})$	$(s^1_{1,1}+s^1_{2,1})+2(s^1_{1,2}+s^1_{2,2})$ $+3(s^1_{1,3}+s^1_{2,3})$
5	$s^2_{1,2}$	$\star s^1_{1,1}-s^2_{1,1}$	$\star s^2_{1,3}$	$\star s^2_{1,4}$
6	$s^2_{2,2}$	$\star s^1_{2,1}-s^2_{2,1}$	$\star s^2_{2,3}$	$\star s^2_{2,4}$
7	$s^2_{1,1}+s^2_{2,3}$ $+s^2_{2,4}$	$\star(s^1_{1,2}+s^1_{1,3}+s^1_{1,4})-$ $(s^2_{1,2}+s^2_{1,3}+s^2_{1,4})$	$\star s^2_{2,1}+s^2_{2,2}+$ $(s^2_{1,4}+s^2_{2,4})$	$\star(s^2_{1,1}+s^2_{2,1})+(s^2_{1,2}+s^2_{2,2})$ $+(s^2_{1,3}+s^2_{2,3})$
8	$s^2_{1,1}+2s^2_{2,3}$ $+3s^2_{2,4}$	$\star(s^1_{1,2}+2s^1_{1,3}+3s^1_{1,4})-$ $(s^2_{1,2}+2s^2_{1,3}+3s^2_{1,4})$	$\star s^2_{2,1}+2s^2_{2,2}+$ $3(s^2_{1,4}+s^2_{2,4})$	$\star(s^2_{1,1}+s^2_{2,1})+2(s^2_{1,2}+s^2_{2,2})$ $+3(s^2_{1,3}+s^2_{2,3})$

viable. Table II shows the case when we apply our code transformation to the first two nodes of the code in Table I. In Table II, the 16 data symbols are $s^{\ell}_{i,j}$ with $\ell = 1, 2$, $i = 1, 2$ and $j = 1, 2, 3, 4$. We first claim that the repair access of each of the first two nodes is optimal. For example, we can repair node 1 by accessing the symbols in rows 5-8 (i.e., the symbols labeled with $\star$ in Table II) from each of the other three nodes. Using the downloaded eight symbols from node 3 and node 4, we can obtain two data symbols $s^2_{2,1}$ and $s^2_{2,2}$ by first subtracting $s^2_{1,4}$ and $s^2_{2,4}$ each from $s^2_{2,1}+s^2_{2,2}+(s^2_{1,4}+s^2_{2,4})$ and $s^2_{2,1}+2s^2_{2,2}+3(s^2_{1,4}+s^2_{2,4})$, followed by solving for $s^2_{2,1}$ and $s^2_{2,2}$ from $s^2_{2,1}+s^2_{2,2}$ and $s^2_{2,1}+2s^2_{2,2}$. Similarly, we can obtain $s^2_{1,1}$ and $s^2_{1,2}$ by subtracting the data symbols $s^2_{2,1}$, $s^2_{2,2}$, $s^2_{1,3}$ and $s^2_{2,3}$ from two coded symbols downloaded from node 4 and solving a 2×2 linear system. Then, we can compute the following four symbols

$$s^2_{1,1}+s^2_{2,3}+s^2_{2,4}, s^2_{1,1}+2s^2_{2,3}+3s^2_{2,4},$$
$$s^2_{1,2}+s^2_{1,3}+s^2_{1,4}, s^2_{1,2}+2s^2_{1,3}+3s^2_{1,4}.$$

Together with the downloaded four symbols from node 2, we can recover the first four symbols in node 1. We can repair node 2 by accessing the symbols in rows 1-4 from each of nodes 1, 3 and 4.

We next claim that we only need to send 10 symbols (the minimum update bandwidth) to the nodes to update the corresponding symbols if four data symbols $s^1_{1,j}, s^1_{2,j}, s^2_{1,j}, s^2_{2,j}$ are updated, where $j = 1, 2, 3, 4$. For example, suppose that the data symbols $s^1_{1,1}, s^1_{2,1}, s^2_{1,1}, s^2_{2,1}$ are updated into $\bar{s}^1_{1,1}, \bar{s}^1_{2,1}, \bar{s}^2_{1,1}, \bar{s}^2_{2,1}$. We only need to send three symbols $\bar{s}^1_{1,1} - s^1_{1,1}, \bar{s}^2_{1,1} - s^2_{1,1}, \bar{s}^1_{2,1} + \bar{s}^2_{2,1}$ to node 1, three coded symbols $\bar{s}^1_{1,1} - s^1_{1,1}, \bar{s}^2_{1,1} - s^2_{1,1}, \bar{s}^1_{2,1} - \bar{s}^2_{2,1}$ to node 2, two coded symbols $\bar{s}^1_{2,1} - s^1_{2,1}, \bar{s}^2_{2,1} - s^2_{2,1}$ to node 3, and two coded symbols $\bar{s}^1_{1,1} + \bar{s}^1_{2,1} - (s^1_{1,1} + s^1_{2,1}), \bar{s}^2_{1,1} + \bar{s}^2_{2,1} - (s^2_{1,1} + s^2_{2,1})$ to node 4.

Note that the code in Table II is MDS, i.e., we can retrieve all 16 data symbols from any two nodes. For example, from nodes 3 and 4, we can subtract $s^1_{1,4}, s^1_{2,4}$ each from $s^1_{2,1} + s^1_{2,2} + s^1_{1,4} + s^1_{2,4}$ and $s^1_{2,1} + 2s^1_{2,2} + 3(s^1_{1,4} + s^1_{2,4})$ in node 3 to obtain $s^1_{2,1} + s^1_{2,2}$ and $s^1_{2,1} + 2s^1_{2,2}$, respectively, and further solve for $s^1_{2,1}$ and $s^1_{2,2}$. By subtracting $s^1_{1,3}, s^1_{2,3}$ each from $(s^1_{1,1} + s^1_{2,1}) + (s^1_{1,2} + s^1_{2,2}) + (s^1_{1,3} + s^1_{2,3})$ and $(s^1_{1,1} + s^1_{2,1}) + 2(s^1_{1,2} + s^1_{2,2}) + 3(s^1_{1,3} + s^1_{2,3})$ in node 4, we obtain $(s^1_{1,1} + s^1_{2,1}) + (s^1_{1,2} + s^1_{2,2})$ and $(s^1_{1,1} + s^1_{2,1}) + 2(s^1_{1,2} + s^1_{2,2})$. Then, we can solve $s^1_{1,1} + s^1_{2,1}$ and $s^1_{1,2} + s^1_{2,2}$. Recall that $s^1_{2,1}$ and $s^1_{2,2}$ are known, we can further obtain $s^1_{1,1}$ and $s^1_{1,2}$. Similarly, we can compute $s^2_{1,1}, s^2_{1,2}, s^2_{2,1}, s^2_{2,2}$ from the eight symbols in rows 5-8 in nodes 3 and 4.

In this paper, we design a new generic transformation for any $m \times n$ MDS array code to obtain a transformed MDS array code of size $m(d-k+1) \times n$ with three properties: (i) it achieves optimal repair access for a single-node repair among $d-k+1$ nodes; (ii) it maintains the same update bandwidth as the underlying $m \times n$ MDS array code; and (iii) it is an MDS code. By recursively applying the generic transformation for the $n \times n$ vertical MDS array code in [13] $\lceil \frac{n}{d-k+1} \rceil$ times, we can obtain an $n \cdot (d-k+1)^{\lceil \frac{n}{d-k+1} \rceil} \times n$ multi-layer transformed vertical MDS array code, in which each node contains $k \cdot (d-k+1)^{\lceil \frac{n}{d-k+1} \rceil}$ data symbols and $(n-k) \cdot (d-k+1)^{\lceil \frac{n}{d-k+1} \rceil}$ coded symbols, such that the transformed code achieves not only optimal repair access for any single-node repair, but also minimum update bandwidth when $k \cdot (d-k+1)^{\lceil \frac{n}{d-k+1} \rceil}$ data symbols are updated.

There exist some similar transformations [9], [11], [14] for MDS codes to achieve optimal repair access for a subset of n nodes. However, the update bandwidth increases in the existing transformed codes in [9], [11], [14], while the update bandwidth of our transformed codes is kept the same as that of the underlying MDS codes.

II. Generic Transformation for MDS Array Codes

A. Transformation Design

For $j = 1, 2, \ldots, n$, let the m symbols stored in node j be

$$\mathbf{v}_j = \begin{bmatrix} v_{1,j} & v_{2,j} & \cdots & v_{m,j} \end{bmatrix}^T,$$

such that we can reconstruct all km data symbols from any k out of n vectors $\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_n$. In the following, we present the generic transformation for MDS array codes to enable optimal repair access for a single-node repair among $d-k+1$ nodes, where $k+1 \le d \le n-1$. Without loss of generality, we assume that the first $d-k+1$ nodes of the transformed codes are the selected nodes, for which we enable optimal repair access.

First, we define the multiplication of a scalar e and a vector

$$\mathbf{v} = \begin{bmatrix} v_1 & v_2 & \ldots & v_m \end{bmatrix}$$

by

$$e\mathbf{v} = \begin{bmatrix} ev_1 & ev_2 & \ldots & ev_m \end{bmatrix}$$

and the addition of two vectors

$$\mathbf{v}^1 = \begin{bmatrix} v_1^1 & v_2^1 & \ldots & v_m^1 \end{bmatrix}$$

and

$$\mathbf{v}^2 = \begin{bmatrix} v_1^2 & v_2^2 & \ldots & v_m^2 \end{bmatrix}$$

by

$$\mathbf{v}^1 + \mathbf{v}^2 = \begin{bmatrix} v_1^1 + v_1^2 & v_2^1 + v_2^2 & \ldots & v_m^1 + v_m^2 \end{bmatrix}.$$

Let $t = d-k+1$. We create t instances

$$\begin{aligned} &\mathbf{v}_1^1, \mathbf{v}_2^1, \cdots, \mathbf{v}_n^1, \\ &\mathbf{v}_1^2, \mathbf{v}_2^2, \cdots, \mathbf{v}_n^2, \\ &\cdots, \\ &\mathbf{v}_1^t, \mathbf{v}_2^t, \cdots, \mathbf{v}_n^t, \end{aligned}$$

of the MDS array codes, such that any k out of n vectors $\mathbf{v}_1^\ell, \mathbf{v}_2^\ell, \cdots, \mathbf{v}_n^\ell$ can reconstruct km data symbols for $\ell = 1, 2, \ldots, t$. For $j = 1, 2, \ldots, t$, the t vectors (tm symbols) stored in node j are obtained by the following three steps.

1) We perform the cyclic-right-shift of $i-1$ positions of the i-row ($i = 1, 2, \ldots, t$) of the following $t \times t$ matrix

$$\mathbf{V}_{t\times t}^1 = \begin{bmatrix} \mathbf{v}_1^1 & \mathbf{v}_2^1 & \cdots & \mathbf{v}_t^1 \\ \mathbf{v}_1^2 & \mathbf{v}_2^2 & \cdots & \mathbf{v}_t^2 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{v}_1^t & \mathbf{v}_2^t & \cdots & \mathbf{v}_t^t \end{bmatrix}$$

to obtain the matrix

$$\mathbf{V}_{t\times t}^2 = \begin{bmatrix} \mathbf{v}_1^1 & \mathbf{v}_2^1 & \cdots & \mathbf{v}_t^1 \\ \mathbf{v}_t^2 & \mathbf{v}_1^2 & \cdots & \mathbf{v}_{t-1}^2 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{v}_2^t & \mathbf{v}_3^t & \cdots & \mathbf{v}_1^t \end{bmatrix}$$

2) For $i, j \in \{1, 2, \ldots, t\}$, the entry in row i and column j of the matrix $\mathbf{V}_{t\times t}^2$ is $\mathbf{v}^i_{((j-i) \bmod t)+1}$. When $i + j < t+1$, we replace the entry in row i and column j of the matrix $\mathbf{V}_{t\times t}^2$ by $\mathbf{v}^i_{((j-i) \bmod t)+1} + \mathbf{v}^{t+1-j}_{((j-i) \bmod t)+1}$. When $i+j > t+1$, we replace the entry in row i and column j of the matrix $\mathbf{V}_{t\times t}^2$ by $e\mathbf{v}^i_{((j-i) \bmod t)+1} + \mathbf{v}^{t+1-j}_{((j-i) \bmod t)+1}$. The resulting matrix $\mathbf{V}_{t\times t}^3$ is

$$\begin{bmatrix} \mathbf{v}_1^1 + \mathbf{v}_1^t & \mathbf{v}_2^1 + \mathbf{v}_2^{t-1} & \cdots & \mathbf{v}_t^1 \\ \mathbf{v}_t^2 + \mathbf{v}_t^t & \mathbf{v}_1^2 + \mathbf{v}_1^{t-1} & \cdots & \mathbf{v}_{t-1}^1 + e\mathbf{v}_{t-1}^2 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{v}_3^{t-1} + \mathbf{v}_3^t & \mathbf{v}_4^{t-1} & \cdots & \mathbf{v}_2^1 + e\mathbf{v}_2^{t-1} \\ \mathbf{v}_2^t & \mathbf{v}_3^{t-1} + e\mathbf{v}_3^t & \cdots & \mathbf{v}_1^1 + e\mathbf{v}_1^t \end{bmatrix}. \quad (2)$$

3) The t vectors stored in node j are the t entries in column j of the matrix $\mathbf{V}_{t\times t}^3$ for $j = 1, 2, \ldots, t$.

For $j = t+1, t+2, \ldots, n$, node j stores t vectors $\mathbf{v}_j^1, \mathbf{v}_j^2, \ldots, \mathbf{v}_j^t$. The obtained codes are our transformed codes.

Table II shows the transformed code by directly applying the transformation with $e = -1$ for the example in Table I.

B. Properties

We first present the following lemma before showing the properties of the transformed codes.

Lemma 1. *If $e \neq 0$ and $e \neq 1$, then we can obtain*

1) *$\mathbf{v}_\ell^i$ and $\mathbf{v}_\ell^j$ from $\mathbf{v}_\ell^i + \mathbf{v}_\ell^j$ and $\mathbf{v}_\ell^i + e\mathbf{v}_\ell^j$;*
2) *$\mathbf{v}_\ell^i + \mathbf{v}_\ell^j$ from $\mathbf{v}_\ell^j$ and $\mathbf{v}_\ell^i + e\mathbf{v}_\ell^j$;*
3) *$\mathbf{v}_\ell^i + e\mathbf{v}_\ell^j$ from $\mathbf{v}_\ell^j$ and $\mathbf{v}_\ell^i + \mathbf{v}_\ell^j$;*
4) *$\mathbf{v}_\ell^j$ from $e\mathbf{v}_\ell^j$.*

Proof. Consider the first statement. From $\mathbf{v}_\ell^i + \mathbf{v}_\ell^j$ and $\mathbf{v}_\ell^i + e\mathbf{v}_\ell^j$, we can obtain

$$\begin{aligned} &v_{h,\ell}^i + v_{h,\ell}^j \text{ for } 1 \le h \le m, \\ &v_{h,\ell}^i + ev_{h,\ell}^j \text{ for } 1 \le h \le m. \end{aligned}$$

We can compute $v_{h,\ell}^i$ and $v_{h,\ell}^j$ by

$$\begin{aligned} v_{h,\ell}^i &= \frac{e(v_{h,\ell}^i + v_{h,\ell}^j) - (v_{h,\ell}^i + ev_{h,\ell}^j)}{e-1}, \\ v_{h,\ell}^j &= \frac{(v_{h,\ell}^i + v_{h,\ell}^j) - (v_{h,\ell}^i + ev_{h,\ell}^j)}{1-e}. \end{aligned}$$

The last three statements can be proven similarly. □

The next two theorems show that the transformed codes are MDS codes and have optimal repair access for a single-node repair of the first t nodes.

Theorem 2. *If $e \neq 0$ and $e \neq 1$, then the transformed codes satisfy the MDS property.*

Proof. The transformed codes satisfy the MDS property if any k out of n nodes can reconstruct all the data symbols.

Suppose that the indices of the chosen k nodes are $j_1, j_2, \ldots, j_k$, where $1 \le j_1 < \ldots < j_k \le n$. If $j_1 \ge t+1$, then the kt vectors stored in the chosen k nodes are $\mathbf{v}_{j_1}^\ell, \mathbf{v}_{j_2}^\ell, \cdots, \mathbf{v}_{j_k}^\ell$ with $\ell = 1, 2, \ldots, t$. We can thus compute all the data symbols since the underlying codes are MDS codes.

In the following, we consider the case that $1 \le j_1 < j_2 < \cdots < j_h < t$ and $j_{h+1} \ge t+1$ with $1 \le h \le$

t. The entry in row $t+1-j_1$ and column j_2 of the matrix $\mathbf{V}^3_{t\times t}$ is $\mathbf{v}^{t+1-j_2}_{((j_2+j_1-1) \bmod t)+1} + e\mathbf{v}^{t+1-j_1}_{((j_2+j_1-1) \bmod t)+1}$; while the entry in row $t+1-j_2$ and column j_1 is $\mathbf{v}^{t+1-j_2}_{((j_1+j_2-1) \bmod t)+1} + \mathbf{v}^{t+1-j_1}_{((j_1+j_2-1) \bmod t)+1}$. By Lemma 1, we can obtain $\mathbf{v}^{t+1-j_2}_{((j_1+j_2-1) \bmod t)+1}$ and $\mathbf{v}^{t+1-j_1}_{((j_1+j_2-1) \bmod t)+1}$ from the above two entries. Similarly, we can obtain $\mathbf{v}^{t+1-j_\ell}_{((j_1+j_\ell-1) \bmod t)+1}$ and $\mathbf{v}^{t+1-j_1}_{((j_1+j_\ell-1) \bmod t)+1}$ from the entries in row $t+1-j_1$ and column j_ℓ, and row $t+1-j_\ell$ and column j_1, for $\ell = 2, 3, \ldots, h$. Recall that the entry in row $t+1-j_1$ and column j_1 is $\mathbf{v}^{t+1-j_1}_{((2j_1-1) \bmod t)+1}$. We obtain

$$\{\mathbf{v}^{t+1-j_1}_{((2j_1-1) \bmod t)+1}, \mathbf{v}^{t+1-j_1}_{((j_1+j_2-1) \bmod t)+1}, \ldots, \mathbf{v}^{t+1-j_1}_{((j_1+j_h-1) \bmod t)+1}\},$$

together with the entries in row $t+1-j_1$ and columns $j_{h+1}, j_{h+2}, \ldots, j_k$, we can compute all the data symbols in row $t+1-j_1$ according to the MDS property of the underlying codes.

By the same argument, we can reconstruct all the data symbols in rows $t+1-j_2, \ldots, t+1-j_h$. Once the vectors in rows $t+1-j_2, \ldots, t+1-j_h$ are known, we can reconstruct all the other data symbols similarly. $\square$

Theorem 3. *The transformed codes have the optimal repair access for a single-node repair of the first t nodes.*

Proof. For $j = 1, 2, \ldots, t$, we show that we can recover node j by accessing k vectors $\mathbf{v}^{t+1-j}_{h_1}, \mathbf{v}^{t+1-j}_{h_2}, \ldots, \mathbf{v}^{t+1-j}_{h_k}$ with $t+1 \le h_1 < \cdots < h_k \le n$ and $t-1$ entries in row $t+1-j$ of the matrix $\mathbf{V}^3_{t\times t}$ except the failed node j.

By accessing $\mathbf{v}^{t+1-j}_{h_1}, \mathbf{v}^{t+1-j}_{h_2}, \ldots, \mathbf{v}^{t+1-j}_{h_k}$, we can compute $\mathbf{v}^{t+1-j}_{1}, \mathbf{v}^{t+1-j}_{2}, \ldots, \mathbf{v}^{t+1-j}_{t}$ as the underlying codes are MDS codes. With the obtained $\mathbf{v}^{t+1-j}_{((2j-1) \bmod t)+1}$ and the accessed $t-1$ entries in row $t+1-j$ of the matrix $\mathbf{V}^3_{t\times t}$ except the failed node j, we can compute all t vectors in node j by Lemma 1. Thus, we can repair t vectors in node j by accessing $k+t-1 = d$ vectors and the repair access is optimal by Eq. (1). $\square$

The repair method of node 1 of the transformed code in Table II (see Section I) is due to the proof of Theorem 3 and the repair access is optimal due to Theorem 3. The next theorem shows that the transformed codes maintain the same update bandwidth as that of the underlying MDS codes.

Theorem 4. *In the (n,k) underlying MDS codes, suppose that we want to update the data symbol $s_{i,j}$ with $1 \le j \le n$ and $1 \le i \le k$, and we need to send the coded symbol c_h to node u_h to update some symbols in vector $\mathbf{v}_h$ for $h = 1, 2, \ldots, \eta$ with $1 \le \eta \le n$ and $1 \le u_1 < \cdots < u_\eta \le n$. In the transformed codes, if we want to update t data symbols $s^1_{i,j}, s^2_{i,j}, \ldots, s^t_{i,j}$, we need to send t coded symbols $c^1_h, c^2_h, \ldots, c^t_h$ to node u_h to update some symbols in vectors $\mathbf{v}^1_h, \mathbf{v}^2_h, \ldots, \mathbf{v}^t_h$ when $u_h > t$, and one coded symbol to each of the first t nodes when $u_h \le t$.*

Proof. By assumption, the coded symbol c^ℓ_h is needed to update some symbols in $\mathbf{v}^\ell_{u_h}$, where $\ell = 1, 2, \ldots, t$ and $h = 1, 2, \ldots, \eta$. If $u_h > t$, we can update the symbols in $\mathbf{v}^\ell_{u_h}$ stored in node u_h with the coded symbols $c^1_h, c^2_h, \ldots, c^t_h$.

Consider the case of $u_h \le t$. For $j \le t$, the t vectors stored in node j are

$$\begin{aligned}
&\mathbf{v}^1_{((j-1) \bmod t)+1} + \mathbf{v}^{t+1-j}_{((j-1) \bmod t)+1},\\
&\mathbf{v}^2_{((j-2) \bmod t)+1} + \mathbf{v}^{t+1-j}_{((j-2) \bmod t)+1}, \ldots,\\
&\mathbf{v}^{t-j}_{((2j) \bmod t)+1} + \mathbf{v}^{t+1-j}_{((2j) \bmod t)+1},\\
&\mathbf{v}^{t+1-j}_{((2j-1) \bmod t)+1},\\
&\mathbf{v}^{t+1-j}_{((2j-2) \bmod t)+1} + e\mathbf{v}^{t+2-j}_{((2j-2) \bmod t)+1},\\
&\mathbf{v}^{t+1-j}_{((2j-3) \bmod t)+1} + e\mathbf{v}^{t+3-j}_{((2j-3) \bmod t)+1}, \ldots,\\
&\mathbf{v}^{t+1-j}_{((j) \bmod t)+1} + e\mathbf{v}^{t}_{((j) \bmod t)+1}.
\end{aligned}$$

For $i_1 \ne i_2 \in \{1, 2, \ldots, t\}$, we have

$$\{((j-i_1) \bmod t) + 1 \ne ((j-i_2) \bmod t) + 1\}$$

and

$$\{((j-1) \bmod t)+1, ((j-2) \bmod t)+1, \ldots, ((j-t) \bmod t)+1\} = \{1, 2, \ldots, t\}.$$

Thus, we only need to send one coded symbol to each of the first t nodes to update some symbols. Specifically, we need to send one of the following symbols

$$\begin{aligned}
&c^{j+t+1-u_h}_{u_h} + c^{t+1-j}_{u_h},\\
&c^{j+1-u_h}_{u_h} + c^{t+1-j}_{u_h},\\
&c^{j+1-u_h}_{u_h},\\
&c^{t+1-j}_{u_h} + ec^{t+j+1-u_h}_{u_h},\\
&c^{t+1-j}_{u_h} + ec^{j+1-u_h}_{u_h},
\end{aligned}$$

to node j to update some symbols in the vector with subscript being u_h. $\square$

By Theorem 4, if the update bandwidth of updating one data symbol of the underlying MDS codes is η, then the update bandwidth of updating t data symbols of the transformed codes is $t\eta$. Thus, the normalized update bandwidth (i.e., the ratio of update bandwidth to the number of updated data symbols) of the transformed codes is the same as that of the underlying codes. If the underlying codes have optimal update bandwidth, then the transformed codes also have optimal update bandwidth.

Recall that the update bandwidth of the code in Table I is five. The update bandwidth of the transformed code in Table II is 10 when the four data symbols $s^1_{1,j}, s^1_{2,j}, s^2_{1,j}, s^2_{2,j}$ need to be updated by Theorem 4, where $j = 1, 2, 3, 4$. For example, when $j = 1$, i.e., the data symbols $s^1_{1,1}, s^1_{2,1}, s^2_{1,1}, s^2_{2,1}$ are updated into $\bar{s}^1_{1,1}, \bar{s}^1_{2,1}, \bar{s}^2_{1,1}, \bar{s}^2_{2,1}$. We can send three symbols $\bar{s}^1_{1,1} + \bar{s}^2_{1,1}, \bar{s}^1_{2,1} + \bar{s}^2_{2,1}, \bar{s}^2_{1,1} - s^2_{1,1}$ to node 1 to update $s^1_{1,1} + s^2_{1,1}, s^1_{2,1} + s^2_{2,1}, s^2_{1,1} + s^2_{2,3} + s^2_{2,4}, s^2_{1,1} + 2s^2_{2,3} + 3s^2_{2,4}$, three symbols $\bar{s}^1_{1,1} - s^1_{1,1}, \bar{s}^1_{1,1} - \bar{s}^2_{1,1}, \bar{s}^1_{2,1} - \bar{s}^2_{2,1}$ to node 2 to update $s^1_{1,1} + s^1_{2,3} + s^1_{2,4}, s^1_{1,1} + 2s^1_{2,3} + 3s^1_{2,4}, s^1_{1,1} - s^2_{1,1}, s^1_{2,1} - s^2_{2,1}$, two symbols $\bar{s}^1_{2,1} - s^1_{2,1}, \bar{s}^2_{2,1} - s^2_{2,1}$ to node 3 to update

$$\begin{aligned}
&s^1_{2,1} + s^1_{2,2} + (s^1_{1,4} + s^1_{2,4}), s^1_{2,1} + 2s^1_{2,2} + 3(s^1_{1,4} + s^1_{2,4}),\\
&s^2_{2,1} + s^2_{2,2} + (s^2_{1,4} + s^2_{2,4}), s^2_{2,1} + 2s^2_{2,2} + 3(s^2_{1,4} + s^2_{2,4}),
\end{aligned}$$

TABLE III: Example of the transformed codes with $k=3$, $n=6$ and $t=3$.

Node 1	Node 2	Node 3	Node 4	Node 5	Node 6
$\mathbf{v}_1^1+\mathbf{v}_1^3$	$\mathbf{v}_2^1+\mathbf{v}_2^2$	$\mathbf{v}_3^1$	$\mathbf{v}_4^1$	$\mathbf{v}_5^1$	$\mathbf{v}_6^1$
$\mathbf{v}_3^2+\mathbf{v}_3^3$	$\mathbf{v}_1^2$	$\mathbf{v}_2^1+e\mathbf{v}_2^2$	$\mathbf{v}_4^2$	$\mathbf{v}_5^2$	$\mathbf{v}_6^2$
$\mathbf{v}_2^3$	$\mathbf{v}_3^2+e\mathbf{v}_3^3$	$\mathbf{v}_1^1+e\mathbf{v}_1^3$	$\mathbf{v}_4^3$	$\mathbf{v}_5^3$	$\mathbf{v}_6^3$

and two symbols $\bar{s}_{1,1}^1-s_{1,1}^1+\bar{s}_{2,1}^1-s_{2,1}^1$, $\bar{s}_{1,1}^2-s_{1,1}^2+\bar{s}_{2,1}^2-s_{2,1}^2$ to node 4 to update

$$\begin{aligned}
&(s_{1,1}^1+s_{2,1}^1)+(s_{1,2}^1+s_{2,2}^1)+(s_{1,3}^1+s_{2,3}^1),\\
&(s_{1,1}^1+s_{2,1}^1)+2(s_{1,2}^1+s_{2,2}^1)+3(s_{1,3}^1+s_{2,3}^1),\\
&(s_{1,1}^2+s_{2,1}^2)+(s_{1,2}^2+s_{2,2}^2)+(s_{1,3}^2+s_{2,3}^2),\\
&(s_{1,1}^2+s_{2,1}^2)+2(s_{1,2}^2+s_{2,2}^2)+3(s_{1,3}^2+s_{2,3}^2).
\end{aligned}$$

Table III shows an example with $k=3$, $n=6$ and $t=3$. Suppose that we want to update one symbol c_1 in vector $\mathbf{v}_1$ in the underlying code. For the transformed code, we only need to send one symbol $c_1^1+c_1^3$ to node 1 to update the symbol in vector $\mathbf{v}_1^1+\mathbf{v}_1^3$, one symbol c_1^2 to node 2 to update the symbol in vector $\mathbf{v}_1^2$ and one symbol $c_1^1+ec_1^3$ to node 3 to update the symbol in vector $\mathbf{v}_1^1+e\mathbf{v}_1^3$.

The code in Table III have optimal repair access for single-node repair among the first three nodes. For example, we can repair node 1 by accessing five vectors

$$\mathbf{v}_3^2+e\mathbf{v}_3^3, \mathbf{v}_1^1+e\mathbf{v}_1^3, \mathbf{v}_4^3, \mathbf{v}_5^3, \mathbf{v}_6^3$$

from the other five nodes and the repair access is optimal. Specifically, we can first compute $\mathbf{v}_1^3$, $\mathbf{v}_2^3$ and $\mathbf{v}_3^3$ from the downloaded three vectors $\mathbf{v}_4^3, \mathbf{v}_5^3, \mathbf{v}_6^3$. Then, we can recover two vectors $\mathbf{v}_1^1+\mathbf{v}_1^3$ and $\mathbf{v}_3^2+\mathbf{v}_3^3$ by

$$\begin{aligned}
\mathbf{v}_1^1+\mathbf{v}_1^3 =&(1-e)\mathbf{v}_1^3+(\mathbf{v}_1^1+e\mathbf{v}_1^3),\\
\mathbf{v}_3^2+\mathbf{v}_3^3 =&(1-e)\mathbf{v}_3^3+(\mathbf{v}_3^2+e\mathbf{v}_3^3).
\end{aligned}$$

The repair method of node 2 and node 3 is similar.

III. MDS Array Codes with Optimal Repair Access and Optimal Update Bandwidth

In this section, we propose the construction of MDS array codes that have optimal repair access for any single-node repair and optimal update bandwidth by applying the transformation in Section II multiple times for the MDS array codes in [13].

We first provide a brief overview of the MDS codes in [13]. The MDS code given in [13] is an (n,k) vertical MDS array code of size $n\times n$, where node j stores k data symbols $s_{1,j}, s_{2,j}, \ldots, s_{k,j}$ and $n-k$ coded symbols $c_{1,j}, c_{2,j}, \ldots, c_{n-k,j}$ with $j=1,2,\ldots,n$. If the k data symbols stored in any one node should be updated, it is shown in [13] that we need to send at least $n+k-1$ symbols to update all the related symbols (including the updated k data symbols and some related coded symbols). Table I shows an example of $n=4$ and $k=2$.

Recall that there are n nodes in MDS array codes in [13]. We divide n nodes into $\lceil\frac{n}{d-k+1}\rceil$ *partitions*, each of which contains $d-k+1$ nodes. For $i=1,2,\ldots,\lceil\frac{n}{d-k+1}\rceil$, partition i contains $d-k+1$ nodes that are from node $(i-1)(d-k+1)+1$ to node $i(d-k+1) \bmod n$.

By applying the transformation in Section II for the first partition (nodes 1 to $d-k+1$) of the codes in [13], we can obtain the transformed codes of size $n(d-k+1)\times n$, denoted by $\mathcal{C}_1(n,k,d)$, that satisfy MDS property according to Theorem 2, have optimal repair access for a single-node repair among the first $d-k+1$ nodes according to Theorem 3 and have optimal update bandwidth according to Theorem 4 when the $k(d-k+1)$ data symbols are updated. Specifically, the update bandwidth of $\mathcal{C}_1(n,k,d)$ is $(d-k+1)(n+k-1)$ if $k(d-k+1)$ data symbols $s_{1,j}^\ell, s_{2,j}^\ell, \ldots, s_{k,j}^\ell$ with $\ell=1,2,\ldots,d-k+1$ are updated, where $j=1,2,\ldots,n$. By applying the transformation for the second partition (nodes between $d-k+2$ and $2(d-k+1)$) of the codes $\mathcal{C}_1(n,k,d)$, we obtain the codes $\mathcal{C}_2(n,k,d)$ with each node storing $n\cdot(d-k+1)^2$ symbols. The codes $\mathcal{C}_2(n,k,d)$ are MDS codes according to Theorem 2, have optimal repair access for a single-node repair among the first $2(d-k+1)$ nodes according to Theorem 3. As the update bandwidth of the underlying codes $\mathcal{C}_1(n,k,d)$ is $(d-k+1)(n+k-1)$ when $k(d-k+1)$ data symbols are updated, the update bandwidth of $\mathcal{C}_2(n,k,d)$ is $(n+k-1)(d-k+1)^2$ when the $k(d-k+1)^2$ data symbols are updated according to Theorem 4 and the update bandwidth of $\mathcal{C}_2(n,k,d)$ is optimal.

Similarly, by recursively applying the transformation for partition $i+1$ of $\mathcal{C}_i(n,k,d)$ for $i=2,3,\ldots,\lceil\frac{n}{d-k+1}\rceil-1$, we obtain the codes $\mathcal{C}_{\lceil\frac{n}{d-k+1}\rceil}(n,k,d)$ with each node storing $n\cdot(d-k+1)^{\lceil\frac{n}{d-k+1}\rceil}$ symbols. The codes $\mathcal{C}_{\lceil\frac{n}{d-k+1}\rceil}(n,k,d)$ are MDS codes according to Theorem 2, have optimal repair access for any single-node repair according to Theorem 3. As the update bandwidth of the underlying codes $\mathcal{C}_{\lceil\frac{n}{d-k+1}\rceil-1}(n,k,d)$ is $(d-k+1)^{\lceil\frac{n}{d-k+1}\rceil-1}(n+k-1)$ when $k(d-k+1)^{\lceil\frac{n}{d-k+1}\rceil-1}$ data symbols are updated, the update bandwidth of $\mathcal{C}_{\lceil\frac{n}{d-k+1}\rceil}(n,k,d)$ is $(n+k-1)(d-k+1)^{\lceil\frac{n}{d-k+1}\rceil}$ when the $k(d-k+1)^{\lceil\frac{n}{d-k+1}\rceil}$ data symbols are updated according to Theorem 4 and the update bandwidth of $\mathcal{C}_{\lceil\frac{n}{d-k+1}\rceil}(n,k,d)$ is optimal.

IV. Conclusion and Future Work

In this paper, we propose a new transformation for any MDS array code to obtain the transformed MDS array code that has optimal repair access for a single-node repair of the $d-k+1$ chosen nodes and the same update bandwidth as the underling MDS array code. By recursively applying the proposed transformation for the MDS array codes in [13] many times, we can obtain the multi-layer transformed MDS array codes that have optimal repair access for any single-node repair and optimal update bandwidth. One of our future work is how to combine our transformation and the existing efficient decoding methods [15], [16] and repair methods [17]–[19] of binary MDS array codes, so as to obtain MDS array codes with lower sub-packetization level, efficient repair access for any single node, efficient update bandwidth, and efficient decoding method.

References

[1] I. S. Reed and G. Solomon, "Polynomial Codes over Certain Finite Fields," *Journal of the Society for Industrial & Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[2] A. Dimakis, P. 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, September 2010.

[3] 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 Trans. Information Theory*, vol. 57, no. 8, pp. 5227–5239, August 2011.

[4] C. Suh and K. Ramchandran, "Exact-Repair MDS Code Construction Using Interference Alignment," *IEEE Trans. Information Theory*, vol. 57, no. 3, pp. 1425–1442, March 2011.

[5] E. E. Gad, R. Mateescu, F. Blagojevic, C. Guyot, and Z. Bandic, "Repair-Optimal MDS Array Codes over GF(2)," in *Proc. IEEE Int. Symp. Inf. Theory*, 2013, pp. 887–891.

[6] H. Hou, K. W. Shum, M. Chen, and H. Li, "BASIC Codes: Low-Complexity Regenerating Codes for Distributed Storage Systems," *IEEE Trans. Information Theory*, vol. 62, no. 6, pp. 3053–3069, 2016.

[7] M. Ye and A. Barg, "Explicit Constructions of High-Rate MDS Array Codes with Optimal Repair Bandwidth," *IEEE Trans. Information Theory*, vol. 63, no. 4, pp. 2001–2014, 2017.

[8] ——, "Explicit Constructions of Optimal-Access MDS Codes with Nearly Optimal Sub-Packetization," *IEEE Trans. Information Theory*, p. 63076317, 2017.

[9] J. Li, X. Tang, and C. Tian, "A Generic Transformation to Enable Optimal Repair in MDS Codes for Distributed Storage Systems," *IEEE Trans. Information Theory*, vol. 64, no. 9, pp. 6257–6267, 2018.

[10] M. Vajha, V. Ramkumar, B. Puranik, G. Kini, E. Lobo, B. Sasidharan, P. V. Kumar, A. Barg, M. Ye, S. Narayanamurthy, S. Hussain, and S. Nandi, "Clay Codes: Moulding MDS Codes to Yield an MSR Code," in *16th USENIX Conference on File and Storage Technologies (FAST 18)*, Oakland, CA, 2018, pp. 139–154.

[11] H. Hou and P. P. Lee, "Binary MDS Array Codes with Optimal Repair," *IEEE Trans. Information Theory*, vol. 66, no. 3, pp. 1405–1422, Mar. 2020.

[12] H. Hou, P. P. C. Lee, and Y. S. Han, "Multi-Layer Transformed MDS Codes with Optimal Repair Access and Low Sub-Packetization," *arXiv preprint: arXiv:1907.08938*, 2019.

[13] Z. Li and S.-J. Lin, "Update Bandwidth for Distributed Storage," in *Proc. IEEE Int. Symp. Inf. Theory*, 2019, pp. 1577–1581.

[14] J. Li and X. Tang, "A Note on the Transformation to Enable Optimal Repair in MDS Codes for Distributed Storage Systems," *arXiv preprint: arXiv:1901.06067v1*, 2019.

[15] H. Hou and Y. S. Han, "A new construction and an efficient decoding method for rabin-like codes," *IEEE Trans. Communications*, vol. 66, no. 2, pp. 521–533, 2018.

[16] H. Hou, Y. S. Han, K. W. Shum, and H. Li, "A Unified Form of EVENODD and RDP Codes and Their Efficient Decoding," *IEEE Trans. Communications*, vol. 66, no. 11, pp. 5053–5066, 2018.

[17] H. Hou, P. P. C. Lee, Y. S. Han, and Y. Hu, "Triple-Fault-Tolerant Binary MDS Array Codes with Asymptotically Optimal Repair," in *Proc. IEEE Int. Symp. Inf. Theory*, Aachen, June 2017, pp. 839–843.

[18] H. Hou, Y. S. Han, P. P. C. Lee, Y. Hu, and H. Li, "A New Design of Binary MDS Array Codes with Asymptotically Weak-Optimal Repair," *IEEE Trans. Information Theory*, vol. 65, no. 11, pp. 7095–7113, 2019.

[19] H. Hou and Y. S. Han, "A Class of Binary MDS Array Codes with Asymptotically Weak-Optimal Repair," *SCIENCE CHINA Information Sciences*, vol. 61, no. 10, pp. 1–12, 2018.