Successively Solvable Shift-Add Systems — a Graphical Characterization

Xiaopeng Cheng, Ximing Fu, Yuanxin Guo, Kenneth W. Shum and Shenghao Yang†

***Abstract*—In order to reduce computational complexity in data encoding, one can use bitwise shifts and logical XOR operations instead of more costly calculations, and apply a fast decoding method called zigzag decoding. Existing works on zigzag decoding usually design special generator matrices that enable certain zigzag solving algorithms. In this paper, we study this class of fast decoding methods holistically. The shift operations are represented by a shift matrix, whose entries are integers or a special infinity symbol. A negative entry signifies that some symbols are truncated, and an infinity symbol means that the corresponding input sequence is not involved in the encoding process. Two notions of solvability, called successive solvability and zigzag solvability, are formulated. The former is employed in most of the existing works on zigzag decoding, and is a special case of the latter one. We prove in this paper that these two notions of solvability are equivalent when the shift matrix have no negative entries. An equivalent condition for a successively solvable shift-XOR system is derived in terms of a directed graph, when the shift matrix has only finite entries. This characterization reveals the structure and the interconnections between the problem instances.**

I. Introduction

Due to the advantage of low encoding and decoding computation costs, (non-cyclic) shift and XOR operations have been extensively studied in the past several years for constructing storage codes [1]–[6], regenerating codes [7]–[10], fountain codes [11]–[14] and network codes [15]. The encoding of shift-XOR codes involves only XOR calculations. Using n message sequences, each consisting of L bits, the encoding of a (non-systematic) coded sequence requires at most $(n-1)L$ XOR operations.

The decoding problem of these shift-XOR codes involves solving an $n \times n$ system of equations with binary polynomials as coefficients. When the coefficients satisfy certain conditions, we can apply a low-complexity zigzag decoding in solving the system of equations. Existing works have characterized several classes of codes that are zigzag decodable. When the generator matrix satisfies the refined increasing difference (RID) property, a zigzag decoding algorithm can be applied [1], [4], [10], which consumes the same number of XOR operations as in the encoding the n coded sequences. Another class of generator matrices that are zigzag solvable has a circulant structure [5]. Greedy algorithms have been developed to determine whether a given shift-XOR system is zigzag solvable [1], [16]. As far as we know, however, a systematic treatment of zigzag solvability is not available in literature.

In this paper, we study $n \times n$ shift-XOR systems where the shift operations can be represented by a *shift matrix* with integers, and possibly an ∞ symbol, as its entries. Here a negative entry means truncation of the corresponding sequence and ∞ means the corresponding sequence is not involved. Similar representation of shift-XOR systems has been used in literature, e.g., [16]. In Sec. II, we present the formal definitions of *zigzag solvability* and *successive solvability*. The latter is a special case of the former, by further requiring that the symbols in a sequence are solved in ascending order of the symbol indices. The definition of successive solvability is general enough to include RID and circulant generator matrices studied in [1], [2], [4], [5]. Moreover, the successive solvability and zigzag solvability are proved to be equivalent when the shift matrix does not have negative entries.

Our main result, presented in Sec. IV, is a necessary and sufficient condition for a shift-XOR system to be successively solvable when the shift matrix only has finite entries. Our characterization is based on a graphical description of shift matrices with finite entries. If one step of successive cancellation is applied to a shift-XOR system, a new system is obtained. This relation connects one shift matrix to another one, and a directed graph is hence created. We then show that a shift matrix with only finite entries is successively solvable if and only if it is in or connected to a strongly connected component formed by cycles of shift matrices. See details in Sec. III and IV. In the concluding remarks, we discuss how to extend this characterization to shift matrices with ∞ entries.

II. Shift-Add Systems and Solvabilities

We model a data symbol as a value in a finite abelian group $\mathcal{A}$ with binary operation $+$ and identity element 0. One example of $\mathcal{A}$ is the set $\{0,1\}$ with exclusive OR (XOR) as the binary operation. We denote sequences with entries from $\mathcal{A}$ by lowercase letters in boldface, e.g., $\mathbf{s}$, where the i-th entry is denoted by $\mathbf{s}[i]$, for $i = 0, 1, 2, \ldots$ The subsequence of $\mathbf{s}$ from the i-th entry onwards is denoted by $\mathbf{s}[i:]$. We use the convention that $\mathbf{s}[i] = 0$ for $i < 0$. We denote the set of nonnegative integers by $\mathbb{N}_0$ and the set of integers by $\mathbb{Z}$.

The authors are with The Chinese University of Hong Kong, Shenzhen, Shenzhen, China. X. Fu is also with University of Science and Technology of China. S. Yang is also with Shenzhen Key Laboratory of IoT Intelligent Systems and Wireless Network Technology, and Shenzhen Research Institute of Big Data, Shenzhen, China. This work was funded in part by Shenzhen Science and Technology Innovation Committee (Grant ZDSYS20170725140921348, JCYJ20180508162604311).

†Corresponding author. Email: shyang@cuhk.edu.cn

Formally speaking, an infinite sequence $\mathbf{s}$ is a mapping from $\mathbb{Z}$ to $\mathcal{A}$ with $i \mapsto 0$ for $i < 0$.

A. Shift-Add System and Shift-Add Code

Definition 1 (shift operator z^t)**.** The *shift operator* z^t is defined such that for any infinite sequence $\mathbf{s}$, the infinite sequence $z^t\mathbf{s}$ is defined by $(z^t\mathbf{s})[\ell] = \mathbf{s}[\ell - t]$ for $\ell \geq \max\{t, 0\}$ and $(z^t\mathbf{s})[\ell] = 0$ otherwise.

A shift-add system takes n infinite sequences $\mathbf{x}_1, \ldots, \mathbf{x}_n$ as input and produces m infinite sequences $\mathbf{y}_1, \ldots, \mathbf{y}_m$ as output. For $i = 1, 2, \ldots, m$, the ith output sequence $\mathbf{y}_i$ is obtained by

$$\mathbf{y}_i := \sum_{j=1}^{m} z^{a_{ij}}\mathbf{x}_j, \tag{1}$$

where a_{ij} are integers in $\mathbb{Z}$ for $j = 1, 2, \ldots, n$. The summation of sequences are performed componentwise. The integer a_{ij} represents the number of positions we shifts the sequence $\mathbf{x}_j$. We also use the convention that $z^\infty = 0$, the scalar zero, so that $z^\infty\mathbf{s}$ is the all-zero sequence. When $a_{ij} = \infty$, the sequence $\mathbf{x}_j$ is not involved in the summation in (1).

Definition 2. We call a matrix $\Phi = (z^{a_{ij}})$ a *shift matrix*, where a_{ij} are integers or ∞. A shift matrix $\Phi = (z^{a_{ij}})$ can be more conveniently represented by an integer matrix $A = (a_{ij})$. From here on, we make no distinction between Φ and the integer matrix A, and refer to both matrices as shift matrix.

Shift-XOR systems have been used in literature to construct storage codes [1]–[10]. Here we use an example to illustrate the application.

Example 1. Consider a storage system of 4 storage nodes that can tolerate one node failure. A file formed by three binary sequences $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ can be stored as coded sequences $\mathbf{y}_1, \ldots, \mathbf{y}_4$:

$$\begin{bmatrix}\mathbf{y}_1\\ \mathbf{y}_2\\ \mathbf{y}_3\\ \mathbf{y}_4\end{bmatrix} = \begin{bmatrix}1 & z & z^2\\ 1 & z^2 & z^4\\ 1 & z^3 & z^6\\ 1 & 1 & 1\end{bmatrix}\begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\\ \mathbf{x}_3\end{bmatrix},$$

where $\mathbf{y}_i$ is stored at the ith storage node. To retrieve the file from the system, it has been shown in [1], [2] that any 3 coded sequences can be used to recover the file. Suppose $\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3$ are read. The decoding is to solve a shift-XOR system with the shift matrix

$$\begin{bmatrix}0 & 1 & 2\\ 0 & 2 & 4\\ 0 & 3 & 6\end{bmatrix}. \tag{2}$$

B. Zigzag Solvability and Successive Solvability

To discuss in general how to solve a shift-add system, we first define what is called *infinite linear system*, which is more general than (1). Given a sequence of symbols $\mathbf{s}$, consider a sequence $\mathbf{u}$ whose entries are the sum of some entries in $\mathbf{s}$. For $j \in \mathbb{N}_0$, suppose the jth entry $\mathbf{u}[j]$ is obtained by

$$\mathbf{u}[j] = \sum_{i \in E_j} \mathbf{s}[i] \tag{3}$$

for some finite index set $E_j \subset \mathbb{N}_0$. The additions in the above equation are performed in $\mathcal{A}$. The index sets E_j, for $j \in \mathbb{N}_0$, define an encoding function: $\mathbf{s} \mapsto \mathbf{u}$. We say that the infinite linear system (3) is *decodable*, or *solvable*, if the encoding function is injective on all sequences of symbols.

For practical implementation, an efficient method for solving the infinite system in (3) is preferred. One specific class of encoding functions allows a simple sequential decoding procedure as defined as follows.

Definition 3. An infinite linear system as in (3) is *zigzag solvable* if there exist two functions $f : \mathbb{N}_0 \to \mathbb{N}_0$ and $g : \mathbb{N}_0 \to \mathbb{N}_0$ such that

1) f is bijective;
2) $\{f(i)\} \subseteq E_{g(i)} \subseteq \{f(0), f(1), \ldots, f(i)\}$ for all $i \in \mathbb{N}_0$.

If we can find functions f and g satisfying the requirements in Definition 3, we can recover $\mathbf{s}$ from $\mathbf{u}$. The function f specifies a decoding order for the symbols in $\mathbf{s}$.

For the system in (1), we re-arrange the entries in the input sequences $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in a single sequence $\mathbf{s}$, and all entries in output sequences $\mathbf{y}_1, \ldots, \mathbf{y}_m$ into another sequence $\mathbf{u}$. For $i \in \mathbb{N}_0$, we let

$$\mathbf{s}[in + j - 1] := \mathbf{x}_j[i], \qquad \mathbf{u}[im + j - 1] := \mathbf{y}_j[i]. \tag{4}$$

The encoding equation

$$\mathbf{y}_j[i] = \mathbf{x}_1[i - a_{j1}] + \mathbf{x}_2[i - a_{j2}] + \cdots + \mathbf{x}_n[i - a_{jn}]$$

can be represented as $\mathbf{u}[im+j-1] = \sum_{k \in E_{im+j-1}} \mathbf{s}[k]$ where

$$E_{im+j-1} = \mathbb{N}_0 \cap \{(i - a_{j1})n, (i - a_{j2})n + 1, \ldots, (i - a_{jn})n + n - 1\}. \tag{5}$$

For notational convenience, we regard $\mathbf{x}_j[i]$'s and $\mathbf{y}_j[i]$'s also as the indices $in + j - 1$ and $im + j - 1$, respectively, in the following discussion of this section, and re-write (5) as

$$E_{\mathbf{y}_j[i]} = \big\{\mathbf{x}_1[i - a_{j1}], \mathbf{x}_2[i - a_{j2}], \ldots, \mathbf{x}_n[i - a_{jn}]\big\} \cap \mathbb{N}_0.$$

With the above notation, the definition of zigzag solvability applies to shift-add systems as well. For practical implementation, it would be more preferable if the data symbols in $\mathbf{x}_i$ can be decoded in the order $\mathbf{x}_i[0]$, $\mathbf{x}_i[1]$, $\mathbf{x}_i[2]$, ... We formulate a stronger notion of solvability as follows.

Definition 4 (successive solvability)**.** A shift-add system in (1) is *successively solvable* if it is zigzag solvable and the functions f and g in Definition 3 satisfy an additional condition:

3) $f^{-1}(\mathbf{x}_j[\ell_1]) < f^{-1}(\mathbf{x}_j[\ell_2])$ whenever $0 \leq \ell_1 < \ell_2$, for all $j = 1, 2, \ldots, n$.

The last condition in Definition 4 guarantees that symbol $\mathbf{x}_j[\ell_1]$ is computed before $\mathbf{x}_j[\ell_2]$ if $\ell_1 < \ell_2$. For a successively solvable shift-add system, there exists a zigzag solving algorithm that solves the symbols of each variable sequence from "left" to "right". That is, the algorithm will solve the symbols of $\mathbf{x}_i$ in the order $\mathbf{x}_i[0]$, $\mathbf{x}_i[1]$, $\mathbf{x}_i[2]$, ... This definition of successively solvability is general enough

TABLE I
SUCCESSIVE DECODING OF THE SYSTEM WITH THE SHIFT MATRIX (2). IN THE LAST THREE ROWS, $k = 2, 3, \ldots$

Iteration i	$f(i)$	Equation used in solving $f(i)$
0	$\mathbf{x}_1[0]$	$\mathbf{y}_3[0] = \mathbf{x}_1[0]$
1	$\mathbf{x}_1[1]$	$\mathbf{y}_3[1] = \mathbf{x}_1[1]$
2	$\mathbf{x}_1[2]$	$\mathbf{y}_3[2] = \mathbf{x}_1[2]$
3	$\mathbf{x}_2[0]$	$\mathbf{y}_2[2] = \mathbf{x}_2[0] + \mathbf{x}_1[2]$
$3k-2$	$\mathbf{x}_1[k+1]$	$\mathbf{y}_3[k+1] = \mathbf{x}_1[k+1] + \mathbf{x}_2[k-2] + \mathbf{x}_3[k-5]$
$3k-1$	$\mathbf{x}_2[k-1]$	$\mathbf{y}_2[k+1] = \mathbf{x}_1[k+1] + \mathbf{x}_2[k-1] + \mathbf{x}_3[k-3]$
$3k$	$\mathbf{x}_3[k-2]$	$\mathbf{y}_1[k] = \mathbf{x}_1[k] + \mathbf{x}_2[k-1] + \mathbf{x}_3[k-2]$

TABLE II
ZIGZAG DECODING IN EXAMPLE 2. IN THE LAST THREE ROWS, $k = 2, 3, \ldots$

Iteration i	$f(i)$	Equation used in solving $f(i)$
0	$\mathbf{x}_2[1]$	$\mathbf{y}_2[0] = \mathbf{x}_2[1]$
1	$\mathbf{x}_1[1]$	$\mathbf{y}_1[1] = \mathbf{x}_1[1] + \mathbf{x}_2[1]$
2	$\mathbf{x}_2[0]$	$\mathbf{y}_3[0] = \mathbf{x}_2[0] + \mathbf{x}_1[1]$
3	$\mathbf{x}_1[0]$	$\mathbf{y}_1[0] = \mathbf{x}_1[0] + \mathbf{x}_2[0]$
$3k-2$	$\mathbf{x}_2[k]$	$\mathbf{y}_2[k-1] = \mathbf{x}_2[k]$
$3k-1$	$\mathbf{x}_1[k]$	$\mathbf{y}_1[k] = \mathbf{x}_1[k] + \mathbf{x}_2[k]$
$3k$	$\mathbf{x}_3[k-2]$	$\mathbf{y}_3[k-1] = \mathbf{x}_1[k] + \mathbf{x}_2[k-1] + \mathbf{x}_2[k-2]$

to include the existing algorithms for solving special shift-add systems in literature [1], [2], [4], [5], [10].

For example, the shift-add system with the shift matrix (2) is successively solvable. The order of decoding the data symbols are listed in Table I. However, a zigzag solvable system is not necessarily successively solvable. We present a counter-example below.

Example 2. Consider the shift-add system given by the shift matrix

$$\begin{pmatrix} 0 & 0 & \infty \\ \infty & -1 & \infty \\ -1 & 0 & 1 \end{pmatrix}.$$

This shift-add system is zigzag solvable. The order of decoding the symbols are shown in Table II. Note that the definition of successive decoding is violated. In fact, there is no way to successively decode the system. We have to first decode $\mathbf{x}_2[\ell]$, for some $\ell \geq 1$ to kick-start the zigzag decoding procedure.

The shift-add system in Example 2 is somewhat pathological. However, the next proposition states that for systems satisfying some mild conditions, the two notions of solvability are equivalent.

Proposition 1. *For a shift-add system corresponding to a non-negative shift matrix, it is zigzag solvable if and only if it is successively solvable.*

Henceforth in this paper, we focus on square shift matrices.

III. REDUCTION OF SHIFT-ADD SYSTEMS

In this section we introduce some notation that can help to describe the transition of the shift matrices when solving the corresponding shift-add systems successively. Parts of the discussion here can also be found in literature, e.g., [16], but the systematic treatment of equivalence and reducing operations is novel. To motivate the definitions to be introduced in this section, we first consider a 3×3 example.

Example 3. Consider the following shift-add system.

$$\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \mathbf{y}_3 \end{bmatrix} = \begin{bmatrix} z^{-2} & z^{-1} & 1 \\ z^{-2} & 1 & z^2 \\ 1 & z^3 & z^6 \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix}, \tag{6}$$

where $\mathbf{x}_i$, for $i = 1, 2, 3$, are the input sequences, and $\mathbf{y}_i$, for $i = 1, 2, 3$, are the output sequences. The first two symbols in the sequence $\mathbf{x}_1$ are truncated in $\mathbf{y}_1, \mathbf{y}_2$ and appear in the linear system only through $\mathbf{y}_3$.

This system is successively solvable. We first solve the first 3 symbols in $\mathbf{x}_1$ using the first 3 symbols in $\mathbf{y}_3$. After subtracting $\mathbf{x}[0]$, $\mathbf{x}[1]$ and $\mathbf{x}[2]$ from $\mathbf{y}_1$ and $\mathbf{y}_2$, the linear system is reduced to

$$\begin{bmatrix} \mathbf{y}'_1 \\ \mathbf{y}'_2 \\ \mathbf{y}'_3 \end{bmatrix} = \begin{bmatrix} z & z^{-1} & 1 \\ z & 1 & z^2 \\ 1 & 1 & z^3 \end{bmatrix} \begin{bmatrix} \mathbf{x}_1[3:] \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix}. \tag{7}$$

Note that the shift matrix in (7) is obtained from the shift matrix in (6) by multiplying the first column by z^3 and then dividing the third row by z^{-3}. Correspondingly, $\mathbf{x}[0]$, $\mathbf{x}[1]$ and $\mathbf{x}[2]$ as well as $\mathbf{y}[0]$, $\mathbf{y}[1]$ and $\mathbf{y}[2]$ are removed from the linear system.

We can then obtain the first symbol in $\mathbf{x}_2$ from the first symbol of $\mathbf{y}'_2$. We subtract the value of $\mathbf{x}_2[0]$ from $\mathbf{y}'_1$ and $\mathbf{y}'_3$, and transform the linear system to

$$\begin{bmatrix} \mathbf{y}''_1 \\ \mathbf{y}''_2 \\ \mathbf{y}''_3 \end{bmatrix} = \begin{bmatrix} z & 1 & 1 \\ 1 & 1 & z \\ 1 & z & z^3 \end{bmatrix} \begin{bmatrix} \mathbf{x}_1[3:] \\ \mathbf{x}_2[1:] \\ \mathbf{x}_3 \end{bmatrix}. \tag{8}$$

The shift matrix in (8) is obtained from the shift matrix in (7) by multiplying the second column by z and then dividing the second row by z.

From the first symbols in $\mathbf{y}''_3$, $\mathbf{y}''_1$ and $\mathbf{y}''_2$ we can obtain the first symbols in $\mathbf{x}_1[3:]$, $\mathbf{x}_2[1:]$, and $\mathbf{x}_3$, by solving a system of three equations that can be permuted to an lower triangular form,

$$\begin{bmatrix} \mathbf{y}''_3[0] \\ \mathbf{y}''_2[0] \\ \mathbf{y}''_1[0] \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_1[3] \\ \mathbf{x}_2[1] \\ \mathbf{x}_3[0] \end{bmatrix}.$$

After subtracting $\mathbf{x}_1[3]$, $\mathbf{x}_2[1]$ and $\mathbf{x}_3[0]$, the linear system reduces to another linear system with the same shift matrix as in (8). We can thus proceed to successively decode the rest of the data symbols.

Definition 5. We denote the ith row of shift matrix A by $\mathbf{a}_i$, and an all-one row vector by $\mathbf{1}$. For two row vectors $\mathbf{a} = (a_j)$ and $\mathbf{b} = (b_j)$ of the same length with integer components, we write $\mathbf{a} \succeq \mathbf{b}$ if $a_j \geq b_j$ for all j. In particular, for a constant c, we write $\mathbf{a} \succeq c$ if $\mathbf{a} \succeq c\mathbf{1}$.

Definition 6. We say that two $n \times n$ shift matrices A and B are *equivalent* if for each $i = 1, 2, \ldots, n$, we have

$$\mathbf{a}_i = \mathbf{b}_i + c_i\mathbf{1}$$

for some integers $c_1, c_2, \ldots, c_n$, with $c_i = 0$ whenever $\mathbf{a}_i$ or $\mathbf{b}_i$ contain some negative entries. We adopt the convention $\infty + c = \infty - c = \infty$ for any integer c. We write $A \sim B$ if A is equivalent to B.

The relation defined in Definition 6 is an equivalence relation on all $n \times n$ shift matrices. Any two equivalent shift matrices define the same infinite linear system.

Example 4. The shift matrices

$$\begin{bmatrix} 1 & -1 & 0 \\ 1 & 0 & 2 \\ 3 & 3 & 6 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1 & -1 & 0 \\ 1 & 0 & 2 \\ 0 & 0 & 3 \end{bmatrix}$$

are equivalent. They represent the same system as in (7).

Definition 7. The equivalence class of matrix A is denoted as $\{A\}$. A shift matrix A with $\min(\mathbf{a}_i) = 0$ whenever $\mathbf{a}_i \succeq 0$ is said to be in *canonical form*. Each equivalence class includes a unique matrix A in canonical form, which is called the (canonical) *representative* of the equivalence class.

We can assume that a shift matrix does not contain any row whose entries are all ∞, otherwise the corresponding output sequence is the all-zero sequence and contains no information about the input symbol sequences. Hence the canonical representative of the equivalence class is well-defined. From now on, when we talk about a shift matrix, we usually refer to the canonical representative of its equivalence class.

Definition 8. Given a shift matrix A, the (i, j)-entry is called a *pivot* if (i) $a_{ij} \geq 0$, and (ii) $a_{ij} < a_{ij'}$ for all $j' \neq j$. We say that a shift matrix A is *reductive* if A contains a pivot.

For a shift matrix A with the (i, j)-entry as a pivot, the first symbol in the ith output sequence is precisely equal to the first symbol in the jth input sequence. We can thus solve the first symbol in the jth input sequence by reading the first symbol in the ith output sequence, subtract it from the linear system, and reduce the system to another linear system. We call this process *reducing operation with respect to the ith row*. For instance, in Example 4, the $(2, 2)$-entry is a pivot, and we can directly read off the first symbol in sequence $\mathbf{x}_2$ from the first symbol in $\mathbf{y}_2$. Meanwhile there is no pivot in the first and third row.

After a reducing operation, the shift matrix is modified to another shift matrix. The difference between the two shift matrices before and after a reducing operation is given by the matrix in the following:

Definition 9 (reducing operator)**.** For $k, \ell \in \{1, 2, \ldots, n\}$, we define the $n \times n$ matrix $R_{k\ell}$ by $(R_{k\ell})_{ij} = \delta_{j,\ell} - \delta_{i,k}$, where $\delta_{i,k}$ and $\delta_{j,\ell}$ are the Kronecker's delta function.

The matrix $R_{k\ell}$ can be obtained from the zero matrix by adding 1 to each of the entries in the ℓth column and subtracting 1 from each of the entries in the kth row. If a shift matrix A is reductive with the (k, ℓ)-entry as a pivot, we obtain the matrix $R_{k\ell} + A$ if we perform a reducing operation with respect to the kth row.

Lemma 2. *Suppose the (k, ℓ)-entry of an $n \times n$ shift matrix A is a pivot. If $A' \sim A$, then the (k, ℓ)-entry is a pivot of A', and $(R_{k\ell} + A) \sim (R_{k\ell} + A')$.*

Lemma 2 implies that we can define the reducing operation on the equivalence classes of shift matrices.

Example 5. The process of solving the shift-add system with the shift matrix (2) can be expressed in terms of the matrices $R_{k\ell}$'s as follows. The initial shift matrix is

$$A_0 = \begin{bmatrix} 0 & 1 & 2 \\ 0 & 2 & 4 \\ 0 & 3 & 6 \end{bmatrix}.$$

We apply the reducing operation with respect to the 3rd row three times. The shift matrix becomes

$$A_1 := R_{31} + R_{31} + R_{31} + A_0 = \begin{bmatrix} 3 & 1 & 2 \\ 3 & 2 & 4 \\ 0 & 0 & 3 \end{bmatrix},$$

which is exactly the shift matrix of the system after three iterations in Table I. The $(2, 2)$-entry in A_1 is a pivot, and we perform a reducing operation on the second row. We obtain

$$A_2 := R_{22} + A_1 = \begin{bmatrix} 3 & 2 & 2 \\ 2 & 2 & 3 \\ 0 & 1 & 3 \end{bmatrix},$$

which is the shift matrix of the system after iteration 3 in Table I. The decoding process continues by repeatedly applying reducing operations on the 3rd row, the 2nd row and the 1st row. The shift matrices after these three reducing operations does not change: $A_2 = R_{13} + R_{22} + R_{31} + A_2$. The successively solving algorithm can be represented by a sequence of reducing operators:

$$R_{31}, R_{31}, R_{31}, R_{22}, \underline{R_{31}, R_{22}, R_{13}}, \ldots$$

repeating the underlined part *ad infinitum*.

In general the order of the reducing operators cannot be changed. For instance, in Example 5, the order of applying the reducing operations to row 3, row 2 and row 1 is important. It is not valid if we apply the reducing operations in the order of row 1, row 2 and row 3, because we are not able to find the appropriate pivot in each step.

IV. Graphical Characterization of Successively Solvability

In this section we consider shift-add systems whose shift matrices do not contain the ∞ symbol, i.e., each output sequence is a function of all input sequences.

Definition 10. For a fixed positive integer n, we define a directed (multi)graph $\mathcal{G}_n$ whose vertices are equivalence classes of shift matrices with finite entries. Given two shift matrices A and B with finite entries, there is a directed edge from $\{A\}$ to $\{B\}$ if A has a pivot (k, ℓ) and $R_{k\ell} + A \sim B$.

For the ease of notation, we write $A \xrightarrow{R_{k\ell}} B$ if there is a directed edge from $\{A\}$ to $\{B\}$ with respect to the (k, ℓ) pivot

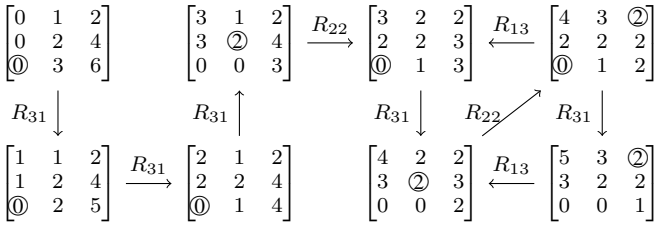

Fig. 1. Successively solving the shift-add system with the shift matrix (2). Pivots are circled.

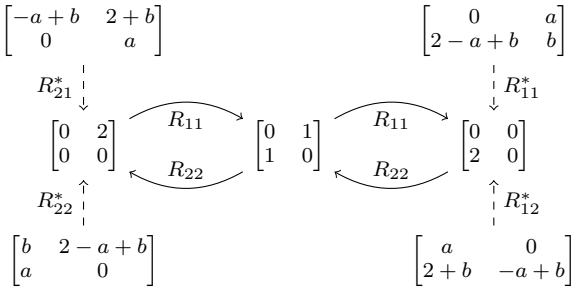

Fig. 2. An extended cycle component in $\mathcal{G}_2$. Here a and b are positive integers. The dashed arrow indicate a directed edge belong to an extended cycle component by not in a cycle component, and the superscript * means that the corresponding reducing operations are applied multiple times.

in A. It follows from definition that there is no self-loop in $\mathcal{G}_n$. When there are multiple pivots in the same column of A, there are multiple edges from $\{A\}$ to $\{B\}$

Definition 11. For positive integer L, we define a *path* of length L in $\mathcal{G}_n$ as a sequence of edges $A_k \xrightarrow{R_{i_k j_k}} B_k$, $k = 1, 2, \ldots, L$, where $B_k \sim A_{k+1}$ for $k = 1, 2, \ldots, L-1$. When $B_L \sim A_1$, the path is called a *cycle*. Given two shift matrices, A and B, we say that a vertex $\{B\}$ is *reachable* from vertex $\{A\}$ in $\mathcal{G}_n$ if there is a path from $\{A\}$ to $\{B\}$.

Remark. A path in this paper need not be simple, i.e., a path may contain repeated vertices. When we refer to a path consisting of distinct vertices, we will emphasize that it is a *simple path*.

The successively solving algorithm of the system discussed in Example 5 can be illustrated as a subgraph of $\mathcal{G}_3$ in Fig. 1. Although the graph $\mathcal{G}_n$ is an infinite graph, its degrees are finite, and the arguments in what follows can be done on a finite part of it.

Lemma 3. *For each shift matrix A, the set of vertices in $\mathcal{G}_n$ that are reachable from $\{A\}$ is finite.*

Theorem 4. *For a cycle in $\mathcal{G}_n$, there is a permutation σ of $\{1, 2, \ldots, n\}$ such that the reducing operations in the cycle are represented by $R_{i\sigma(i)}$ for $i = 1, \ldots, n$.*

Theorem 5. *Let $\{A\}$ be an equivalence class of shift matrices contained in a cycle of $\mathcal{G}_n$. If there is a path from $\{A\}$ to $\{B\}$ for some shift matrix B, then there is a path from $\{B\}$ to $\{A\}$, i.e., $\{A\}$ and $\{B\}$ are contained in a cycle.*

Definition 12. A directed graph $\mathcal{H}$ in general is said to be *strongly connected* if for any two vertices v and u in $\mathcal{H}$, there is a path from v to u and a path from u to v. A *strongly connected component* is a maximal strongly connected subgraph. In $\mathcal{G}_n$, a strongly connected component is also called a *cycle component*. The permutation σ defined in Theorem 4 is uniquely defined throughout a cycle component, and is called the *character* of the cycle component.

The contrapositive of Theorem 5 suggests that for two equivalence classes of shift matrices $\{A\}$ and $\{A'\}$ in different cycle components, there exists no path between $\{A\}$ and $\{A'\}$ in $\mathcal{G}_n$. This property is extended in the next theorem.

Theorem 6. *Suppose $\{B\}$ and $\{C\}$ are two vertices contained in two distinct cycle components in $\mathcal{G}_n$, and there is a path from $\{A\}$ to $\{B\}$. Then there is no path from $\{A\}$ to the cycle component containing $\{C\}$.*

Definition 13. Given a cycle component in $\mathcal{G}_n$, we define its *extended cycle component* be the set of vertices $\{A\}$ in $\mathcal{G}_n$ that connects to the given cycle component, i.e., there is a path from $\{A\}$ to any vertex in the cycle component.

We note that the notion of extended cycle component is well-defined by Theorem 6.

Theorem 7. *An $n \times n$ shift-add system defined by shift matrix A with only finite entries is successively solvable if and only if $\{A\}$ is contained in an extended cycle component of $\mathcal{G}_n$.*

We illustrate an extended cycle component in $\mathcal{G}_2$ in Fig. 2. The dashed arrow indicate a directed edge belong to an extended cycle component by not in a cycle component, and the superscript * means that the corresponding reducing operations are applied multiple times. We note that the shift matrices in the middle has two pivots, and hence we can go to the left or the right from this shift matrix. The shift-add system associated to the shift matrices in Fig. 2 are all successively solvable. The character of the cycle component is $\sigma(1) = 1$ and $\sigma(2) = 2$. We only see R_{11} and R_{22} within the cycle component, but we may have other types of reducing operations outside the cycle component.

V. Concluding Remarks

Our characterization of successively solvable shift matrices also sheds some light on shift matrices with ∞ entries. For a successively solvable shift matrix A with only finite entries, there must exist a path P from $\{A\}$ that includes a cycle. For any subset $\mathcal{C}$ of entries (i, j) in A such that R_{ij} is not used by any edge in the path P, let A' be the matrix obtained by modifying these entries of A in $\mathcal{C}$ to be ∞. The sequence of reducing operations used in P can be performed on A' successively, and hence it can be argued that A' is also successively solvable.

REFERENCES

[1] C. W. Sung and X. Gong, "A ZigZag-decodable code with the MDS property for distributed storage systems," in *IEEE Int. Symp. on Inf. Theory*, Jul. 2013, pp. 341–345.

[2] X. Fu, Z. Xiao, and S. Yang, "Overhead-free in-place recovery scheme for XOR-based storage codes," in *IEEE Int. Conf. Trust, Security and Privacy in Computing and Communications*, Sep. 2014, pp. 552–557.

[3] M. Dai, X. Wang, H. Wang, X. Lin, and B. Chen, "Bandwidth overhead-free data reconstruction scheme for distributed storage code with low decoding complexity," *IEEE Access*, vol. 5, pp. 6824–6832, 2017.

[4] X. Gong and C. W. Sung, "Zigzag decodable codes: Linear-time erasure codes with applications to data storage," *J. Comput. Syst. Sci.*, vol. 89, pp. 190–208, 2017.

[5] M. Dai, C. W. Sung, H. Wang, X. Gong, and Z. Lu, "A new Zigzag-decodable code with efficient repair in wireless distributed storage," *IEEE Trans. Mob. Comput.*, vol. 16, no. 5, pp. 1218–1230, 2017.

[6] M. Dai, B. Mao, X. Gong, C. W. Sung, W. Zhuang, and X. Lin, "Zigzag-division multiple access for wireless networks with long and heterogeneous delays," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 55, no. 6, pp. 2822–2835, 2019.

[7] H. Hou, K. W. Shum, M. Chen, and H. Li, "BASIC regenerating code: Binary addition and shift for exact repair," in *IEEE Int. Symp. on Inf. Theory*, Jul. 2013, pp. 1621–1625.

[8] X. Fu, Z. Xiao, and S. Yang, "Overhead-free in-place recovery and repair schemes of XOR-based regenerating codes," in *IEEE Int. Symp. on Inf. Theory*, Jul. 2015, pp. 341–345.

[9] H. Hou, P. P. C. Lee, and Y. S. Han, "Zigzag-decodable reconstruction codes with asymptotically optimal repair for all nodes," *IEEE Transactions on Communications*, vol. 18, no. 10, pp. 5999–6011, Oct. 2020.

[10] X. Fu, S. Yang, and Z. Xiao, "Decoding and repair schemes for shift-XOR regenerating codes," *IEEE Transactions on Information Theory*, vol. 66, no. 12, pp. 7371–7386, 2020.

[11] T. Nozaki, "Fountain codes based on zigzag decodable coding," in *Proc. IEEE Int. Symp. on Information Theory and its Applications*, 2014, pp. 274–278.

[12] B. Jun, P. Yang, J. No, and H. Park, "New fountain codes with improved intermediate recovery based on batched zigzag coding," *IEEE Trans. Commun.*, vol. 65, no. 1, pp. 23–36, 2017.

[13] Y. Murayama and T. Nozaki, "Efficient scheduling of serial iterative decoding for zigzag decodable fountain codes," in *International Symposium on Information Theory and Its Applications, ISITA 2018, Singapore, October 28-31, 2018*. IEEE, 2018, pp. 286–290.

[14] P. Shi, Z. Wang, D. Li, and W. Xiang, "Zigzag decodable online fountain codes with high intermediate symbol recovery rates," *IEEE Trans. Commun.*, vol. 68, no. 11, pp. 6629–6641, 2020.

[15] C. W. Sung and X. Gong, "Combination network coding: Alphabet size and zigzag decoding," in *2014 International Symposium on Information Theory and its Applications*. IEEE, 2014, pp. 699–703.

[16] M. Dai, B. Mao, X. Gong, C. W. Sung, W. Zhuang, and X. Lin, "Zigzag-division multiple access for wireless networks with long and heterogeneous delays," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 55, no. 6, pp. 2822–2835, 2019.