Codes for Erasures over Directed Graphs

Lev Yohananov
Dept. of Computer Science
Technion-Israel Institute of Technology
Haifa 32000, Israel
Email: levyohananov@campus.technion.ac.il

Eitan Yaakobi
Dept. of Computer Science
Technion-Israel Institute of Technology
Haifa 32000, Israel
Email: yaakobi@cs.technion.ac.il

Abstract—In this work we continue the study of a new class of codes, called *codes over graphs*. Here we consider storage systems where the information is stored on the edges of a complete directed graph with n nodes. The failure model we consider is of *node failures* which are erasures of all edges, both incoming and outgoing, connected to the failed node. It is said that a code over graphs is a ρ-node-erasure-correcting code if it can correct the failure of any ρ nodes in the graphs of the code. While the construction of such optimal codes is an easy task if the field size is $\mathcal{O}(n^2)$, our main goal in the paper is the construction of codes over smaller fields. In particular, our main result is the construction of optimal binary codes over graphs which correct two node failures with a prime number of nodes.

I. Introduction

In this paper we follow up on a recent work from [5] which studies the correction of node failures in graphs. The main idea of the work in [5] was to look at information systems in which the information is represented by a graph. Such systems include for example neural networks [2] and associative memories [4] that mimic the operation of the brain in the sense of storing and processing information by the associations between the information content. A distributed storage system [1] is another example where every two nodes can share a link with the information that is shared between them.

The work in [5] studied a new class of codes, called *codes over graphs*. This model assumes that there are undirected complete graphs with n *nodes* (*vertices*) and the information is stored on the undirected edges which connect every two nodes in the graph. Here, we extend this model to *directed graphs* and construct codes over directed complete graphs; thus, the information is stored on the edges connecting every two nodes in the graph, including the self loops. Under this setup, there are n^2 edges in the graph which store n^2 symbols over some alphabet Σ. A code over graphs will be a set of directed graphs. Here, a *node failure* corresponds to the erasure of all edges in the out- and in-neighborhood of the graph. Thus, a code over graphs will be called a *ρ-node erasure-correcting code* if it can correct the failure of any ρ nodes in each graph in the code.

The main approach in constructing these codes follows the one from [5] in which we use the adjacency matrix of the graph. The adjacency matrix is a square $n \times n$ matrix where its (i, j)-entry corresponds to the symbol stored on the edge from node i to node j. Then, a failure of the i-th node in the graph translates to the erasure of the i-th row and the i-th column in the adjacency matrix of the graph. While there exist numerous constructions of array codes, most of them do not provide square matrices, but more than, they do not support the special structure of rows and columns erasure, as described above. The most relevant model to ours is the one studied by Roth [3] for the correction of crisscross error patterns. For crisscross error patterns it is assumed that some prescribed number of rows and columns have been erased, however the numbers of erased rows and erased columns can be different and with different indices. Therefore, this class of codes can be used to construct node-erasure-correcting codes, however, as we shall see, they will not be optimal.

For any ρ-node-erasure-correcting code, the failure of any ρ nodes translated to $2n\rho - \rho^2$ failed edges in the graph and thus the minimum number of redundancy edges of such a code is at least

$$n^2 - (n - \rho)^2 = 2n\rho - \rho^2.$$

If a code satisfies this bound with equality, it will be called *optimal*. Constructing optimal ρ-node-erasure-correcting code is an easy task if there is no restriction on the field size. For example, one can use an $[n^2, (n - \rho)^2, 2n\rho - \rho^2 + 1]$ MDS code over a field of size at least $n^2 - 1$. Hence, the primary focus in this paper is the construction of node-erasure-correcting codes over a small field, and in particular binary codes.

The rest of this paper is organized as follows. In Section II, we formally define the graph model and codes over graphs we study in this paper. In Section III, we present two constructions of ρ-node-erasure-correcting codes. The first one shows how to construct optimal codes over a field of size $q \geqslant n - 1$. The second construction is based on the codes from [3] for crisscorss error patterns and generates binary codes for all ρ, however they are not optimal. In Section IV, we present our main result in the paper of optimal double-node-erasure-correcting codes. Due to the lack of space some proofs of the results in the paper are omitted.

II. Definitions and Preliminaries

In this section we formally define the codes over graphs we study in the paper. We follow similar definitions from [5] and modify them for directed graphs. Let $G = (V_n, E)$ be a *directed graph*, where $V_n = \{v_0, v_1, \ldots, v_{n-1}\}$ is its set of n *nodes* (*vertices*) and $E \subseteq V_n \times V_n$ is its set of *edges*. A *labeling function* L of a graph G over an alphabet Σ is an assignment to the edges in G by symbols from Σ, i.e., the labeling is a function $L : E \to \Sigma$.

In this work we will only study complete directed graphs with self loops, that is, $E = V_n \times V_n$, with a labeling function $L : V_n \times V_n \to \Sigma$ and we will use the notation $G = (V_n, L)$ for such graphs. The *adjacency matrix* of a graph $G = (V_n, L)$ is an $n \times n$ matrix over Σ denoted by $A_G = [a_{i,j}]_{i=0,j=0}^{n-1,n-1}$, where $a_{i,j} = L(v_i, v_j)$ for all $0 \leqslant i, j \leqslant n - 1$. For an integer $n > 0$ we denote by $[n]$ the set $\{0, 1, \ldots, n - 1\}$. For a prime power q, the finite field of size q will be denoted by $\mathbb{F}_q$. A linear code of size n, dimension k, and minimum distance d over a field $\mathbb{F}_q$ will be denoted by an $[n, k, d]_q$ code.

Let Σ be a ring and G_1 and G_2 be two graphs over Σ with the same nodes set V_n. The operator $"+"$ between G_1 and G_2 over Σ, is defined by $G_1 + G_2 = G_3$, where G_3 is the unique graph satisfying $A_{G_1} + A_{G_2} = A_{G_3}$. Similarly, the operator $"\cdot"$ between G_1 and an element $\alpha \in \Sigma$, is denoted by $\alpha \cdot G_1 = G_3$, where G_3 is the unique graph satisfying $\alpha \cdot A_{G_1} = A_{G_3}$.

Definition 1. *A **code over graphs** of size M and length n over an alphabet Σ is a set of directed complete graphs $\mathcal{C_G} = \{G_i = (V_n, L_i) | i \in [M]\}$ over Σ. We denote such a code by $\mathcal{G}$-$(n, M)_\Sigma$ and in case $\Sigma = \{0, 1\}$, it will simply be denoted by $\mathcal{G}$-(n, M). The **dimension** of a code over graphs*

$C_{\mathcal{G}}$ is $k_{\mathcal{G}} = \log_{|\Sigma|} M$, the **rate** is $R_{\mathcal{G}} = k_{\mathcal{G}}/n^2$, and the **redundancy** is defined by $r_{\mathcal{G}} = n^2 - k_{\mathcal{G}}$. A code over graphs $C_{\mathcal{G}}$ over a ring Σ will be called **linear** if for all $G_1, G_2 \in C_{\mathcal{G}}$ and $\alpha, \beta \in \Sigma$ it holds that $\alpha \cdot G_1 + \beta \cdot G_2 \in C_{\mathcal{G}}$. We denote this family of codes by $\mathcal{G}\text{-}[n, k_{\mathcal{G}}]_{\Sigma}$.

A linear code over graphs whose first k nodes contain the $k_{\mathcal{G}} = k^2$ unmodified information symbols on their edges, is called a **systematic code over graphs**. All other $n^2 - k^2$ edges in the graph are called **redundancy edges**. In this case we say that there are k **information nodes**, $r = n - k$ **redundancy nodes**, and the number of **information edges** is $k_{\mathcal{G}} = k^2$. The redundancy is $r_{\mathcal{G}} = n^2 - k^2$ and the rate is $R_{\mathcal{G}} = k^2/n^2$. We denote such a code by $\mathcal{SG}\text{-}[n, k]_{\Sigma}$.

Let $G = (V_n, L)$ be a graph. For $i \in [n]$, the *out-neighborhood set*, *in-neighborhood set*, of the i-th node is defined to be the set

$$N_i^{\text{out}} = \{(v_i, v_j) \mid j \in [n]\}, \quad N_i^{\text{in}} = \{(v_j, v_i) \mid j \in [n]\},$$

respectively, and the *neighborhood set* of the i-th node is the set $N_i = N_i^{\text{out}} \cup N_i^{\text{in}}$. Note that the i-th out-neighborhood set, in-neighborhood set, corresponds to the i-th column, row, in the adjacency matrix, respectively, and the i-th neighborhood set is the union of the i-th column and the i-th row in the adjacency matrix. A *node failure* of the i-th node is the event in which all the edges in the neighborhood set of the i-th node, i.e. N_i, are erased. We will also denote this set by F_i and refer to it by the *failure set* of the i-th node. For convenience, we also define the *out-failure set, in-failure set* of the i-th node by $F_i^{\text{out}} = N_i^{\text{out}}, F_i^{\text{in}} = N_i^{\text{in}}$, respectively.

When a node failures happens, the failed node is known and it is required to complete the values of the edges in its neighborhood set. This failure model leads us to the following definition.

Definition 2. *A code over graphs is called a ρ-**node-erasure-correcting code** if it can correct the failure of any ρ nodes in each graph in the code.*

The minimum redundancy $r_{\mathcal{G}}$ of any ρ-node-erasure-correcting code of length n is

$$r_{\mathcal{G}} \geqslant n^2 - (n - \rho)^2 = 2n\rho - \rho^2.$$

A code over graphs satisfying this inequality with equality will be called *optimal*. Hence for systematic code over graphs the number of redundancy nodes is at least ρ. For all n and ρ, an optimal ρ-node-erasure-correcting code can be constructed using an $[n^2, (n - \rho)^2, 2n\rho - \rho^2 + 1]_q$ MDS code over a field of size $q \geqslant n^2 - 1$. The main goal of this work is to construct node-erasure-correcting codes over small fields.

III. GENERAL CONSTRUCTIONS OF CODES OVER GRAPHS

In the previous section we saw that optimal ρ-node-erasure-correcting codes are easy to construct over a field of size at least $n^2 - 1$. In this section we will present two constructions that reduce the large field size. Namely, in Section III-A, we will show an improvement of this last result and present constructions of optimal ρ-node-erasure-correcting codes over a field of size at least $n - 1$. In order to further reduce the field size, in Section III-B, we present constructions of binary codes, however these codes will not be optimal.

A. Optimal Codes over a Field of Size $q \geqslant n - 1$

Let $G = (V_n, L)$ be a graph over a field $\mathbb{F}_q$ and $U \subseteq V_n \times V_n$ the subset of its edges. We define $\boldsymbol{c}_U \in \mathbb{F}_q^{|U|}$ to be a vector over $\mathbb{F}_q$ of length $|U|$, where its entries are the labels of the edges in the set U in their lexicographical order. For example, if $U = \{(v_2, v_4), (v_1, v_0), (v_3, v_6), (v_3, v_2)\}$, then $\boldsymbol{c}_U = (L(v_1, v_0), L(v_2, v_4), L(v_3, v_2), L(v_3, v_6))$. We

first start with the following claim on the intersections of neighborhood sets.

Claim 3. *Let J be a subset of $[n]$ of size $\rho \geqslant 1$. Then, the following properties hold:*

(a) *For all $\ell \in [n]$, $|N_\ell^{\text{out}} \cap (\bigcup_{k \in J} F_k^{\text{in}})| = \rho$.*
(b) *For all $m \in [n]$, $|N_m^{\text{in}} \cap (\bigcup_{k \in J} F_k^{\text{out}})| = \rho$.*
(c) *For all $\ell \in [n] \setminus J$, $|N_\ell^{\text{out}} \cap (\bigcup_{k \in J} F_k)| = \rho$.*

We are now ready to present the construction of ρ-node-erasure-correcting codes. Let us consider the adjacency matrix of each graph in the code in order to explain the main idea of the construction. Each of the first $n - \rho$ columns in the adjacency matrix belongs to an $[n, n - \rho, \rho + 1]_q$ MDS code, where $q \geqslant n - 1$, and all rows belong to the same code as well. This construction is formalized as follows.

Construction 1. *Let n and ρ be two positive integers such that $n > \rho$. Let $\mathcal{C}$ be an $[n, n - \rho, \rho + 1]_q$ MDS code, for $q \geqslant n - 1$. The code $C_{\mathcal{G}_1}$ is defined as follows,*

$$C_{\mathcal{G}_1} = \left\{ G = (V_n, L) \; \middle| \; \begin{array}{ll} \forall m \in [n - \rho], & \boldsymbol{c}_{N_m^{\text{in}}} \in \mathcal{C} \\ \forall \ell \in [n], & \boldsymbol{c}_{N_i^{\text{out}}} \in \mathcal{C} \end{array} \right\}.$$

Theorem 4. *For all ρ and n such that $\rho < n$, the code $C_{\mathcal{G}_1}$ is a $\mathcal{G}\text{-}[n, k_{\mathcal{G}} = (n - \rho)^2]_q$ ρ-node-erasure-correcting code, where $q \geqslant n - 1$.*

Note that the code $C_{\mathcal{G}_1}$ can also be a systematic $\mathcal{SG}\text{-}[n, n - \rho]_q$ code where its first $n - \rho$ nodes are the information nodes. In the adjacency matrix, this corresponds to having the information symbols in the upper left $(n - \rho) \times (n - \rho)$ matrix. Then, each of the first $n - \rho$ columns in encoded systematically by a systematic encoder of the MDS code $\mathcal{C}$, and then the same procedure is invoked on each of the n rows.

B. Binary Construction of Codes over Graphs

In this section we present constructions of binary ρ-node-erasure-correcting codes for arbitrary ρ. This construction will be based upon a construction by Roth for the correction of crisscross error patterns [3].

Let $\Gamma = [\gamma_{i,j}]_{i=0,j=0}^{n-1,n-1}$ be an $n \times n$ matrix over a field $\mathbb{F}$. A *cover* of Γ is defined to be a pair of two sets (S, T) where $S, T \subseteq [n]$ such that for all $i, j \in [n]$ if $\gamma_{i,j} \neq 0$ then either $i \in S$ or $j \in T$, [3]. The *cover-weight* of Γ is defined to be the minimum size of any cover (S, T) of Γ, that is,

$$w(\Gamma) = \min_{(S,T) \text{ is a cover of } \Gamma} \{|S| + |T|\}.$$

An $[n \times n, k, d]$ linear array code $\mathcal{C}$ over a field $\mathbb{F}$ is a k-dimensional linear space of $n \times n$ matrices over $\mathbb{F}$, where the minimum cover-weight of all nonzero matrices in $\mathcal{C}$ is d. It was claimed in [3] that the code $\mathcal{C}$ can correct the erasure of any $d - 1$ rows or columns in the array. Furthermore, the singleton bound for such array codes states that [3] $k \leqslant n(n - d + 1)$, and here we refer to array codes which meet this bound as *optimal array codes*. In [3], a construction of optimal array codes $[n \times n, n(n - r), r + 1]$ was given for all $r < n$. In fact, another stronger property of this code was proved in [3], in which the rank of every matrix in the code is at least $r + 1$.

We are now ready to present the construction of binary ρ-node-erasure-correcting codes.

Construction 2. *Let $\mathcal{C}$ be an $[n \times n, n(n - 2\rho), 2\rho + 1]$ binary optimal array code from [3], where $\rho < n/2$. The code over graphs $C_{\mathcal{G}_2}$ is defined as follows,*

$$C_{\mathcal{G}_2} = \{G = (V_n, L) \mid A_G \in \mathcal{C}\}.$$

Theorem 5. *For all $\rho < n/2$, the code $C_{\mathcal{G}_2}$ is a $\mathcal{G}\text{-}[n, k_{\mathcal{G}} = n(n - 2\rho)]$ ρ-node-erasure-correcting code.*

The construction of binary optimal array codes $[n \times n, n(n-r), r+1]$ from [3] has also a systematic construction, where the first $n-2\rho$ rows of each matrix store the information bits and the last 2ρ rows store the redundancy bits. Therefore, we can use this family of codes also for the construction of systematic $\mathcal{SG}\text{-}[n, k = n - 2\rho]$ codes over graphs for $\rho < n/2$.

We saw in this section that optimal codes exist for a field of size at least $n-1$, while the binary construction does not provide optimal codes. Our next task is to achieve these two properties simultaneously, that is, optimal binary codes. In the next section we show how to accomplish this task for two node failures, when the number of nodes is a prime number. The general case for arbitrary number of node failures is left for future work.

IV. DOUBLE-NODE-ERASURE-CORRECTING CODES

In this section we present a construction of optimal binary double-node-erasure-correcting codes. We first start by reviewing a construction from [5] for the correction of two node failures for undirected graphs. We then show another construction of such codes for undirected graphs, and lastly we show how to combine between these two constructions in order to generate a code correcting two node failures for directed graphs.

We refer to a graph with only undirected edges as an *undirected graph*. Here, we consider only complete undirected graphs and denote them by $G_\mathcal{U} = (V_n, L_\mathcal{U})$, where $V_n = \{v_0, v_1, \ldots, v_{n-1}\}$ is the set of n nodes, and there exists an undirected edge between every two nodes, including self loops. As for the directed case, $L_\mathcal{U}$ is a *labeling function* that assigns every edge with a symbol over some alphabet Σ. An edge between node v_i to node v_j is denoted by $\langle v_i, v_j \rangle$ where the order in this pair does not matter, that is, the pair $\langle v_i, v_j \rangle$ is identical to the pair $\langle v_j, v_i \rangle$.

An undirected graph $G_\mathcal{U}$ can be represented by its *lower-triangle-adjacency matrix* of order $n \times n$, that is, $A_{G_\mathcal{U}} = [a_{i,j}]_{i=0, j=0}^{n-1, n-1}$ such that $a_{i,j} = L_\mathcal{U}(\langle v_i, v_j \rangle)$ if $i \geqslant j$ and otherwise $a_{i,j} = 0$. It can also be represented by an *upper-triangle-adjacency matrix* by taking the transpose of $A_{G_\mathcal{U}}$.

A *code over undirected graphs* of size M and length n over an alphabet Σ is a set of undirected graphs $\mathcal{C}_\mathcal{U} = \{G_{\mathcal{U}_i} = (V_n, L_{\mathcal{U}_i}) | i \in [M]\}$ over Σ. A *linear code over undirected graphs* is defined in a similar way to the directed case and a linear code over undirected graphs whose first k nodes contain the $\binom{k+1}{2}$ unmodified information symbols on their edges, is called a *systematic code over undirected graphs* and will be denoted by $\mathcal{UG}\text{-}[n, k]_\Sigma$.

The neighborhood set of the i-th node in a graph $G_\mathcal{U} = (V_n, L_\mathcal{U})$ is the set $N_i = \{\langle v_i, v_j \rangle | j \in [n]\}$. A *node failure* of the i-th node is the event in which all the edges in the neighborhood set of the i-th node are erased and we will denote this set by $\widehat{F}_i = N_i$ to indicate a failure set. A code over undirected graphs is called an *undirected ρ-node-erasure-correcting code* if it can correct the failure of any ρ nodes in each graph in the code.

Let $n \geqslant 5$ be a prime number and $G_\mathcal{U} = (V_n, L_\mathcal{U})$ be an undirected graph with n vertices. We use the notation $\langle a \rangle_n$ to denote the value of ($a \bmod n$). Let us define for $h \in [n-1]$

$$S_h = \begin{cases} \{\langle v_h, v_\ell \rangle \mid \ell \in [n-1]\} & , h \in [n-2], \\ \{\langle v_\ell, v_\ell \rangle \mid \ell \in [n-1]\} & , h = n-2, \end{cases}$$

and for $m \in [n]$,

$$D_m = \{\langle v_k, v_\ell \rangle | k, \ell \in [n] \setminus \{n-2\}, \langle k+\ell \rangle_n = m\} \cup \{\langle v_{n-1}, v_{n-2} \rangle\}.$$

The sets S_h where $h \in [n-2]$, will be used to represent parity constraints on the neighborhood of each of the first $n-2$ nodes in the undirected graph (which correspond to "opposite-L paths" in the lower-triangle-adjacency matrix), while the set S_{n-2} will be used to impose a parity constraint on the self loops of the first $n-1$ nodes. Similarly, the sets D_m for $m \in [n]$, will represent parity constraints on the diagonals of the lower-triangle-adjacency matrix of the graph.

Example 1. The sets S_h, D_m for $n = 7$ are marked in Fig. 2. Entries on lines with the same color belong to the same parity constraint.
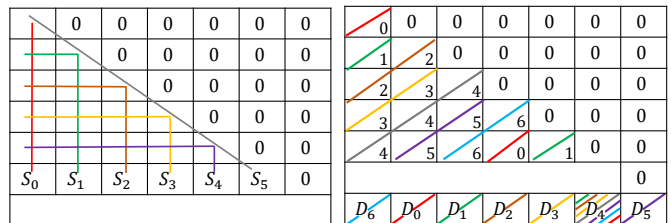
(a) Neighborhood Parity Paths (b) Diagonal Parity Paths

Fig. 1. The neighborhoods and diagonals sets.

In [5], we presented the following construction of systematic binary $\mathcal{UG}\text{-}[n, n-2]$ undirected double-node-erasure-correcting codes.

Construction 3. For all $n \geqslant 5$ prime number let $\mathcal{C}_{\mathcal{U}_1}$ be the following code over graphs,

$$\mathcal{C}_{\mathcal{U}_1} = \left\{ G_\mathcal{U} = (V_n, L_\mathcal{U}) \left| \begin{array}{l} (a) \sum_{\langle v_i, v_j \rangle \in S_h} L_\mathcal{U}(v_i, v_j) = 0, h \in [n-1] \\ (b) \sum_{\langle v_i, v_j \rangle \in D_m} L_\mathcal{U}(v_i, v_j) = 0, m \in [n] \end{array} \right. \right\}.$$

The correction of this construction was proved in [5] by explicitly showing its decoding procedure for any two failed nodes v_i and v_j. The more challenging case in which $i, j \in [n-2]$ works as follows. For $h \in [n-1] \setminus \{i, j\}$ and $m \in [n]$ let $\overline{S}_h$ and $\overline{D}_m$ be the sets $\overline{S}_h = S_h \setminus (\widehat{F}_i \cup \widehat{F}_j)$ and $\overline{D}_m = D_m \setminus (\widehat{F}_i \cup \widehat{F}_j)$. Denote the *syndromes* $\widehat{S}_h, \widehat{D}_m$ by

$$\widehat{S}_h = \sum_{\langle v_k, v_\ell \rangle \in \overline{S}_h} L_\mathcal{U}(v_k, v_\ell), \quad \widehat{D}_m = \sum_{\langle v_k, v_\ell \rangle \in \overline{D}_m} L_\mathcal{U}(v_k, v_\ell).$$

respectively. Let $d = \langle j - i \rangle_n$, $x = \langle -1 - d^{-1} \rangle_n$ and $y = \langle -1 + d^{-1} \rangle_n$. The decoding procedure for this case is presented in Algorithm 1.

Algorithm 1

```
1:  b_prev ← 0                                  15:  b_prev ← 0
2:  for t = 0, 1, ..., x do                      16:  for t = 0, 1, ..., y do
3:      s_1 ← ⟨-d(t+1) - 2⟩_n                     17:      s_1 ← ⟨d(t+1) - 2⟩_n
4:      s_2 ← ⟨s_1 + j⟩_n                         18:      s_2 ← ⟨s_1 + i⟩_n
5:      if (s_1 ∉ {i, j, n-1}) then              19:      if (s_1 ∉ {i, j, n-1}) then
6:          L_U(v_s1, v_j) ← D̂_s2 + b_prev       20:          L_U(v_s1, v_i) ← D̂_s2 + b_prev
7:          L_U(v_s1, v_i) ← Ŝ_s1 + L_U(v_s1, v_j) 21:         L_U(v_s1, v_j) ← Ŝ_s1 + L_U(v_s1, v_i)
8:          b_prev ← L_U(v_s1, v_i)              22:          b_prev ← L_U(v_s1, v_j)
9:      if (s_1 = j) then                        23:      if (s_1 = i) then
10:         L_U(v_s1, v_j) ← D̂_s2 + b_prev       24:          L_U(v_s1, v_i) ← D̂_s2 + b_prev
11:         L_U(v_s1, v_i) ← Ŝ_{n-2} + L_U(v_s1, v_i) 25:      L_U(v_i, v_j) ← Ŝ_{n-2} + L_U(v_s1, v_i)
12:         b_prev ← L_U(v_s1, v_i)              26:          b_prev ← L_U(v_j, v_i)
13:     if s_1 = n-1 then                        27:      if s_1 = n-1 then
14:         L_U(v_s1, v_j) ← D̂_s2 + b_prev       28:          L_U(v_s1, v_i) ← D̂_s2 + b_prev
```


Next we present another construction of systematic binary $\mathcal{UG}\text{-}[n, n-2]$ undirected double-node-erasure-correcting codes which is very similar to the codes from Construction 3. Here, we present this construction by its constraints on its upper-triangle-adjacency matrix representation of the graphs. For $h \in [n-1]$ denote,

$$S'_h = \begin{cases} \{\langle v_h, v_\ell \rangle \mid \ell \in [n] \setminus \{n-2\}\} & , h \in [n-2], \\ \{\langle v_\ell, v_\ell \rangle \mid \ell \in [n] \setminus \{n-2\}\} & , h = n-2, \end{cases}$$

and for $m \in [n]$,

$$D'_m = \{\langle v_k, v_\ell \rangle | k, \ell \in [n-1], \langle k+\ell \rangle_n = m\} \cup \{\langle v_{n-2}, v_{n-1} \rangle\}.$$

As before, the sets S'_h for $h \in [n-1]$ and D'_m for $m \in [n]$, will be used to represent parity constraints on the upper-triangle-adjacency matrix.

Example 2. The sets S'_h, D'_m for $n = 7$ are marked in Fig. 2. Entries on lines with the same color belong to the same parity constraint.
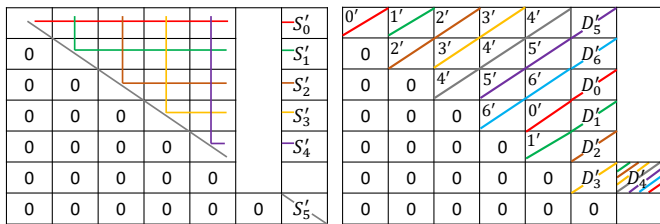
(a) Neighborhood Parity Paths (b) Diagonal Parity Paths

Fig. 2. The neighborhoods and diagonals sets.

Our second construction of $\mathcal{UG}\text{-}[n, n-2]$ undirected double-node-erasure-correcting codes works as follows.

Construction 4. For all $n \geqslant 5$ prime number let $\mathcal{C}_{\mathcal{U}_2}$ be the following code over graphs,

$$\mathcal{C}_{\mathcal{U}_2} = \left\{ G_{\mathcal{U}} = (V_n, L_{\mathcal{U}}) \middle| \begin{array}{l} (a) \sum_{\langle v_i, v_j \rangle \in S'_h} L_{\mathcal{U}}(v_i, v_j) = 0, h \in [n-1] \\ (b) \sum_{\langle v_i, v_j \rangle \in D'_m} L_{\mathcal{U}}(v_i, v_j) = 0, m \in [n] \end{array} \right\}.$$

We will not prove here the correctness of the code $\mathcal{C}_{\mathcal{U}_2}$ since its construction is very similar to one of the code $\mathcal{C}_{\mathcal{U}_1}$. However, note that when constructing the code $\mathcal{C}_{\mathcal{U}_2}$, we switched the roles of the last two redundancy nodes such that the first node is the *diagonal parity node* and the second node is the *single parity node*. However, we still present here a decoding algorithm of this code for the more challenging case when the failed nodes are v_i, v_j and $i, j \in [n-2]$. Its correctness is similar to the one of Algorithm 1 as done in [5]. For $h \in [n-1] \setminus \{i, j\}$ and $m \in [n]$ let $\overline{S}_h$ and $\overline{D}_m$ be the sets $\overline{S}_h = S'_h \setminus (\widehat{F}_i \cup \widehat{F}_j)$ and $\overline{D}_m = D'_m \setminus (\widehat{F}_i \cup \widehat{F}_j)$. Denote the syndromes $\widehat{S}'_h, \widehat{D}'_m$ by

$$\widehat{S}'_h = \sum_{\langle v_k, v_\ell \rangle \in \overline{S}_h} L_{\mathcal{U}}(v_k, v_\ell), \widehat{D}'_m = \sum_{\langle v_k, v_\ell \rangle \in \overline{D}_m} L_{\mathcal{U}}(v_k, v_\ell).$$

Let $x' = \langle -1 + d^{-1} \rangle_n$ and $y' = \langle -1 - d^{-1} \rangle_n$. The decoding procedure for this case is described in Algorithm 2.

In order to construct codes over directed graphs, we will use the two codes above for undirected graphs to get a family of directed systematic binary $\mathcal{SG}\text{-}[n, n-2]$ double-node-erasure-correcting codes.

Algorithm 2

1: $b_{prev} \leftarrow 0$
2: **for** $t = 0, 1, \ldots, x'$ **do**
3: $\quad s_1 \leftarrow \langle -d(t+1) - 1 \rangle_n$
4: $\quad s_2 \leftarrow \langle s_1 + j \rangle_n$
5: $\quad$ **if** $(s_1 \notin \{i, j, n-2\})$ **then**
6: $\qquad L_{\mathcal{U}}(v_{s_1}, v_j) \leftarrow \widehat{D}'_{s_2} + b_{prev}$
7: $\qquad L_{\mathcal{U}}(v_{s_1}, v_i) \leftarrow \widehat{S}'_{s_1} + L(v_{s_1}, v_j)$
8: $\qquad b_{prev} \leftarrow L_{\mathcal{U}}(v_{s_1}, v_i)$
9: $\quad$ **if** $(s_1 = j)$ **then**
10: $\qquad L_{\mathcal{U}}(v_{s_1}, v_j) \leftarrow \widehat{D}'_{s_2} + b_{prev}$
11: $\qquad L_{\mathcal{U}}(v_i, v_i) \leftarrow \widehat{S}'_{n-2} + L_{\mathcal{U}}(v_{s_1}, v_j)$
12: $\qquad b_{prev} \leftarrow L_{\mathcal{U}}(v_i, v_i)$
13: $\quad$ **if** $s_1 = n - 2$ **then**
14: $\qquad L_{\mathcal{U}}(v_{s_1}, v_j) \leftarrow \widehat{D}'_{s_2} + b_{prev}$

15: $b_{prev} \leftarrow 0$
16: **for** $t = 0, 1, \ldots, y'$ **do**
17: $\quad s_1 \leftarrow \langle d(t+1) - 1 \rangle_n$
18: $\quad s_2 \leftarrow \langle s_1 + i \rangle_n$
19: $\quad$ **if** $(s_1 \notin \{i, j, n-2\})$ **then**
20: $\qquad L_{\mathcal{U}}(v_{s_1}, v_i) \leftarrow \widehat{D}'_{s_2} + b_{prev}$
21: $\qquad L_{\mathcal{U}}(v_{s_1}, v_j) \leftarrow \widehat{S}'_{s_1} + L_{\mathcal{U}}(v_{s_1}, v_i)$
22: $\qquad b_{prev} \leftarrow L_{\mathcal{U}}(v_{s_1}, v_j)$
23: $\quad$ **if** $(s_1 = i)$ **then**
24: $\qquad L_{\mathcal{U}}(v_{s_1}, v_i) \leftarrow \widehat{D}'_{s_2} + b_{prev}$
25: $\qquad L_{\mathcal{U}}(v_j, v_j) \leftarrow \widehat{S}'_{n-2} + L_{\mathcal{U}}(v_{s_1}, v_i)$
26: $\qquad b_{prev} \leftarrow L_{\mathcal{U}}(v_j, v_j)$
27: $\quad$ **if** $s_1 = n - 2$ **then**
28: $\qquad L_{\mathcal{U}}(v_{s_1}, v_i) \leftarrow \widehat{D}'_{s_2} + b_{prev}$

For $i, j \in [n]$, not necessarily distinct, let $\langle v_i, v_j \rangle^\downarrow$ be the edge directed from $v_{\max\{i,j\}}$ to $v_{\min\{i,j\}}$, i.e., $\langle v_i, v_j \rangle^\downarrow = (v_{\max\{i,j\}}, v_{\min\{i,j\}})$, and similarly $\langle v_i, v_j \rangle^\uparrow = (v_{\min\{i,j\}}, v_{\max\{i,j\}})$ is the edge directed from $v_{\min\{i,j\}}$ to $v_{\max\{i,j\}}$. For $h \in [n-2]$ the *neighborhood-edge sets* $S_h^\downarrow, S_h^\uparrow$ are defined by

$$S_h^\downarrow = \{ \langle v_i, v_j \rangle^\downarrow | \langle v_i, v_j \rangle \in S_h \}, S_h^\uparrow = \{ \langle v_i, v_j \rangle^\uparrow | \langle v_i, v_j \rangle \in S'_h \}.$$
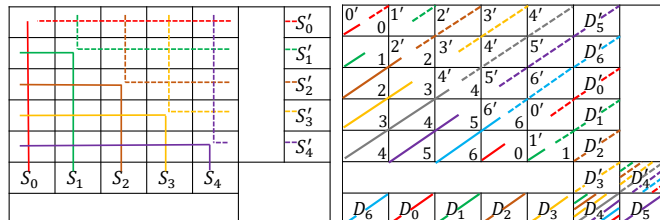
Furthermore, for $m \in [n]$ the *diagonal-edge sets* $D_m^\downarrow, D_m^\uparrow$ are defined by

$$D_m^\downarrow = \{ \langle v_i, v_j \rangle^\downarrow | \langle v_i, v_j \rangle \in D_m \}, D_m^\uparrow = \{ \langle v_i, v_j \rangle^\uparrow | \langle v_i, v_j \rangle \in D'_m \},$$

and for $t \in [n]$ the *failure-edge sets* $F_t^\downarrow, F_t^\uparrow$ are defined by

$$F_t^\downarrow = \{ \langle v_i, v_j \rangle^\downarrow | \langle v_i, v_j \rangle \in \widehat{F}_t \}, F_t^\uparrow = \{ \langle v_i, v_j \rangle^\uparrow | \langle v_i, v_j \rangle \in \widehat{F}_t \}.$$

Example 3. The sets $S_h^\downarrow, S_h^\uparrow, D_m^\downarrow, D_m^\uparrow$ for $n = 7$ are marked in Fig. 3. Entries on lines with the same color belong to the same parity constraint.

(a) Neighborhood Parity Paths (b) Diagonal Parity Paths

Fig. 3. The neighborhoods and diagonals sets.

The following claims for directed sets are very similar to the corresponding claims that were stated in [5].

Claim 6. *For all distinct* $i, j \in [n-2]$, $D_{\langle i+j \rangle_n}^\downarrow \cap F_j^\downarrow = \{ (v_j, v_i) \}$ *and* $D_{\langle i+j \rangle_n}^\uparrow \cap F_j^\uparrow = \{ (v_i, v_j) \}$.

We are now ready to present the construction of double-node-erasure-correcting codes.

Construction 5. For all $n \geqslant 5$ prime number let $\mathcal{C}_{\mathcal{G}_4}$ be the following code.

$$\mathcal{C}_{\mathcal{G}_4} = \left\{ G = (V_n, L) \middle| \begin{array}{l} (a) \sum_{(v_i, v_j) \in S_h^\downarrow} L(v_i, v_j) = 0, h \in [n-2] \\ (b) \sum_{(v_i, v_j) \in D_m^\downarrow} L(v_i, v_j) = 0, m \in [n] \\ (c) \sum_{(v_i, v_j) \in S_h^\uparrow} L(v_i, v_j) = 0, h \in [n-2] \\ (d) \sum_{(v_i, v_j) \in D_m^\uparrow} L(v_i, v_j) = 0, m \in [n] \end{array} \right\}.$$

Note that in this construction we did not use the constraints that were derived from the two sets S_{n-2} and S'_{n-2} (i.e., the constraints on the main diagonal). Even though we do not explicitly prove it here, it is not hard to notice that this construction is systematic where the information is stored on the edges of the first $n-2$ nodes. Hence, in the next proof for the correctness of the construction we will refer to it as a systematic construction.

Theorem 7. *The code $\mathcal{C}_{\mathcal{G}_4}$ is an optimal binary double-node-erasure-correcting code.*

Proof: Assume that nodes $i, j \in [n]$, where $i < j$ are the failed nodes. We will show the correctness of this construction by explicitly showing its decoding algorithm. We will only consider the more difficult case of $i, j \in [n-2]$.

For $h \in [n-1] \setminus \{i, j\}$ denote the sets $\overline{S}_h^\downarrow = S_h^\downarrow \setminus (\widehat{F}_i^\downarrow \cup \widehat{F}_j^\downarrow)$ and $\overline{S}_h^\uparrow = S_h^\uparrow \setminus (\widehat{F}_i^\uparrow \cup \widehat{F}_j^\uparrow)$ and for $m \in [n]$ denote the sets $\overline{D}_m^\downarrow = D_m^\downarrow \setminus (\widehat{F}_i^\downarrow \cup \widehat{F}_j^\downarrow)$ and $\overline{D}_m^\uparrow = D_m^\uparrow \setminus (\widehat{F}_i^\uparrow \cup \widehat{F}_j^\uparrow)$. Then, the *neighborhood syndromes* $\widehat{S}_h^\downarrow, \widehat{S}_h^\uparrow$ are defined by

$$\widehat{S}_h^\downarrow = \sum_{\langle v_k, v_\ell \rangle \in \overline{S}_h^\downarrow} L_{\mathcal{U}}(v_k, v_\ell), \widehat{S}_h^\uparrow = \sum_{\langle v_k, v_\ell \rangle \in \overline{S}_h^\uparrow} L_{\mathcal{U}}(v_k, v_\ell),$$

and the *diagonal syndromes* $\widehat{D}_m^\downarrow, \widehat{D}_m^\uparrow$ are defined by

$$\widehat{D}_m^\downarrow = \sum_{\langle v_k, v_\ell \rangle \in \overline{D}_m^\downarrow} L_{\mathcal{U}}(v_k, v_\ell), \widehat{D}_m^\uparrow = \sum_{\langle v_k, v_\ell \rangle \in \overline{D}_m^\uparrow} L_{\mathcal{U}}(v_k, v_\ell).$$

Let $d = \langle j - i \rangle_n$, $x = \langle -1 - d^{-1} \rangle_n$, $y = \langle -1 + d^{-1} \rangle_n$, $x' = \langle -1 + d^{-1} \rangle_n$ and $y' = \langle -1 - d^{-1} \rangle_n$. The decoding procedure for the code $\mathcal{C}_{\mathcal{G}_4}$ in this case is described in Algorithm 3.

This algorithm consists of four loops marked as Loop I, II, III, and IV. For $Y \in \{I, II, III, IV\}$, denote by $s_{1,Y}^{(t)}$ the

Algorithm 3

Loop I	Loop II	Loop III	Loop IV
1: $b_{prev} \leftarrow 0$	15: $b_{prev} \leftarrow 0$	29: $b_{prev} \leftarrow 0$	43: $b_{prev} \leftarrow 0$
2: **for** $t = 0, 1, \ldots, x$ **do**	16: **for** $t = 0, 1, \ldots, y$ **do**	30: **for** $t = 0, 1, \ldots, x'$ **do**	44: **for** $t = 0, 1, \ldots, y'$ **do**
3: $s_1 \leftarrow \langle -d(t+1) - 2 \rangle_n$	17: $s_1 \leftarrow \langle d(t+1) - 2 \rangle_n$	31: $s_1 \leftarrow \langle -d(t+1) - 1 \rangle_n$	45: $s_1 \leftarrow \langle d(t+1) - 1 \rangle_n$
4: $s_2 \leftarrow \langle s_1 + j \rangle_n$	18: $s_2 \leftarrow \langle s_1 + i \rangle_n$	32: $s_2 \leftarrow \langle s_1 + j \rangle_n$	46: $s_2 \leftarrow \langle s_1 + i \rangle_n$
5: **if** $(s_1 \notin \{i, j, n-1\})$ **then**	19: **if** $(s_1 \notin \{i, j, n-1\})$ **then**	33: **if** $(s_1 \notin \{i, j, n-2\})$ **then**	47: **if** $(s_1 \notin \{i, j, n-2\})$ **then**
6: $\quad L(\langle v_{s_1}, v_j \rangle^{\downarrow}) \leftarrow \widehat{D}^{\downarrow}_{s_2} + b_{prev}$	20: $\quad L(\langle v_{s_1}, v_i \rangle^{\downarrow}) \leftarrow \widehat{D}^{\downarrow}_{s_2} + b_{prev}$	34: $\quad L(\langle v_{s_1}, v_j \rangle^{\uparrow}) \leftarrow \widehat{D}^{\uparrow}_{s_2} + b_{prev}$	48: $\quad L(\langle v_{s_1}, v_i \rangle^{\uparrow}) \leftarrow \widehat{D}^{\uparrow}_{s_2} + b_{prev}$
7: $\quad L(\langle v_{s_1}, v_i \rangle^{\downarrow}) \leftarrow \widehat{S}^{\downarrow}_{s_1} + L(\langle v_{s_1}, v_j \rangle^{\downarrow})$	21: $\quad L(\langle v_{s_1}, v_i \rangle^{\downarrow}) \leftarrow \widehat{S}^{\downarrow}_{s_1} + L(\langle v_{s_1}, v_i \rangle^{\downarrow})$	35: $\quad L(\langle v_{s_1}, v_i \rangle^{\uparrow}) \leftarrow \widehat{S}^{\uparrow}_{s_1} + L(\langle v_{s_1}, v_j \rangle^{\uparrow})$	49: $\quad L(\langle v_{s_1}, v_j \rangle^{\uparrow}) \leftarrow \widehat{S}^{\uparrow}_{s_1} + L(\langle v_{s_1}, v_i \rangle^{\uparrow})$
8: $\quad b_{prev} \leftarrow L(\langle v_{s_1}, v_i \rangle^{\downarrow})$	22: $\quad b_{prev} \leftarrow L(\langle v_{s_1}, v_j \rangle^{\downarrow})$	36: $\quad b_{prev} \leftarrow L(\langle v_{s_1}, v_i \rangle^{\uparrow})$	50: $\quad b_{prev} \leftarrow L(\langle v_{s_1}, v_j \rangle^{\uparrow})$
9: **if** $(s_1 = j)$ **then**	23: **if** $(s_1 = i)$ **then**	37: **if** $(s_1 = j)$ **then**	51: **if** $(s_1 = i)$ **then**
10: $\quad L(v_j, v_j) \leftarrow \widehat{D}^{\downarrow}_{s_2} + b_{prev}$	24: $\quad L(v_i, v_i) \leftarrow \widehat{D}^{\downarrow}_{s_2} + b_{prev}$	38: $\quad L(v_j, v_j) \leftarrow \widehat{D}^{\uparrow}_{s_2} + b_{prev}$	52: $\quad L(v_i, v_i) \leftarrow \widehat{D}^{\uparrow}_{s_2} + b_{prev}$
11: $\quad$ Wait until (v_i, v_i) is corrected.	25: $\quad$ Wait until (v_j, v_j) is corrected.	39: $\quad$ Wait until (v_i, v_i) is corrected.	53: $\quad$ Wait until (v_j, v_j) is corrected.
12: $\quad b_{prev} \leftarrow L(v_i, v_i)$	26: $\quad b_{prev} \leftarrow L(v_j, v_j)$	40: $\quad b_{prev} \leftarrow L(v_i, v_i)$	54: $\quad b_{prev} \leftarrow L(v_j, v_j)$
13: **if** $s_1 = n-1$ **then**	27: **if** $s_1 = n-1$ **then**	41: **if** $s_1 = n-2$ **then**	55: **if** $s_1 = n-2$ **then**
14: $\quad L(v_{n-2}, v_j) \leftarrow \widehat{D}^{\downarrow}_{s_2} + b_{prev}$	28: $\quad L(v_{n-1}, v_i) \leftarrow \widehat{D}^{\downarrow}_{s_2} + b_{prev}$	42: $\quad L(v_j, v_{n-2}) \leftarrow \widehat{D}^{\downarrow}_{s_2} + b_{prev}$	56: $\quad L(v_i, v_{n-2}) \leftarrow \widehat{D}^{\uparrow}_{s_2} + b_{prev}$

value of the variable s_1 on iteration t of Loop Y. These values of $s_{1,Y}^{(t)}$ are given by:

$$s_{1,\mathrm{I}}^{(t)} = \langle -d(t+1) - 2 \rangle_n, s_{1,\mathrm{II}}^{(t)} = \langle d(t+1) - 2 \rangle_n,$$
$$s_{1,\mathrm{III}}^{(t)} = \langle -d(t+1) - 1 \rangle_n, s_{1,\mathrm{IV}}^{(t)} = \langle d(t+1) - 1 \rangle_n.$$

Next, we denote the following four sets:

$$A = \{s_{1,\mathrm{I}}^{(t)} : t \in [x+1]\}, B = \{s_{1,\mathrm{II}}^{(t)} : t \in [y+1]\},$$
$$A' = \{s_{1,\mathrm{III}}^{(t')} : t' \in [x'+1]\}, B' = \{s_{1,\mathrm{IV}}^{(t')} : t \in [y'+1]\}.$$

Claim 8. *The indices i, j satisfy the following property: $i, j \in A \cap B'$ or $i, j \in A' \cap B$, but not in both.*

The decoding Algorithm 3 for this case combines Algorithm 1 and Algorithm 2, where Algorithm 1 is used to decode the lower-triangle-adjacency and Algorithm 2 is used to decode the upper-triangle-adjacency matrix. However, since we did not use the constraints of the two sets S_{n-2} and S'_{n-2} on the main diagonal, we had to replace Step 11, 25 in Algorithm 1, Algorithm 2 with the command *wait until (v_i, v_i) is corrected*, *wait until (v_j, v_j) is corrected*, respectively. According to Claim 8, the indices i, j satisfy $i, j \in A \cap B'$ or $i, j \in A' \cap B$ but not both. Without loss of generality, assume that $i, j \in A \cap B'$. Therefore, in this case, Loops II and III of Algorithm 3 will not be affected by the main diagonal constraint. This holds since the edges (v_i, v_i) and (v_j, v_j) are not corrected in these two loops as the conditions in Steps 23 and 37 will not hold. Hence, these two loops operate and succeed exactly as done in Algorithm 1 and Algorithm 2. This does not hold for Loops I and IV. Namely, Loop I, IV operates exactly as Algorithm 1, Algorithm 2 until Loop I, IV reaches Step 11, 53, respectively. Here we notice that according to Algorithm 1, in Step 11, the algorithm was supposed to correct the edge (v_i, v_i) according to the constraint on the mail diagonal. Similarly, in Step 53, the algorithm was supposed to correct the edge (v_j, v_j) according to the constraint on the mail diagonal. However, since the edge (v_i, v_i) is corrected in Loop IV and the edge (v_j, v_j) is corrected in Loop I, all we need to do in Step 11 is to wait for the edge (v_i, v_i) to be corrected and in the same way in Step 53 for the edge (v_j, v_j) to be corrected. Then, the rest of these two loops proceed to correct the remaining edges as done in Algorithm 1 and Algorithm 2.

Lastly, from Claim 6, $D^{\downarrow}_{\langle i+j \rangle_n} \cap F^{\downarrow}_j = \{(v_j, v_i)\}$ and $D^{\uparrow}_{\langle i+j \rangle_n} \cap F^{\uparrow}_j = \{(v_i, v_j)\}$, so the last two information edges (v_j, v_i) and (v_i, v_j) are corrected by constraints $\widehat{D}^{\downarrow}_{i+j}$ and $\widehat{D}^{\uparrow}_{i+j}$, respectively. Since all of the information edges were corrected, we can correct the remaining uncorrected redundancy edges $(v_{n-2}, v_i), (v_{n-2}, v_j), (v_i, v_{n-1})$ and (v_j, v_{n-1}) using our encoding rules. ∎

The decoding algorithm presented in the proof of Theorem 7 is demonstrated in the next example.

Example 4.

(a) Simulation of the algorithm

First Loop (Red)				Second Loop (Green)			
s_2	s_1	$\langle v_{s_1}, v_j \rangle^{\downarrow}$	$\langle v_{s_1}, v_i \rangle^{\downarrow}$	s_2	s_1	$\langle v_{s_1}, v_i \rangle^{\downarrow}$	$\langle v_{s_1}, v_j \rangle^{\downarrow}$
1	7	$\langle v_7, v_5 \rangle^{\downarrow}$	$\langle v_7, v_3 \rangle^{\downarrow}$	3	0	$\langle v_0, v_3 \rangle^{\downarrow}$	$\langle v_0, v_5 \rangle^{\downarrow}$
10	5	$\langle v_5, v_5 \rangle^{\downarrow}$	—	5	2	$\langle v_2, v_3 \rangle^{\downarrow}$	$\langle v_2, v_5 \rangle^{\downarrow}$
8	3	—	—	7	4	$\langle v_4, v_3 \rangle^{\downarrow}$	$\langle v_4, v_5 \rangle^{\downarrow}$
6	1	$\langle v_1, v_5 \rangle^{\downarrow}$	$\langle v_1, v_3 \rangle^{\downarrow}$	9	6	$\langle v_6, v_3 \rangle^{\downarrow}$	$\langle v_6, v_5 \rangle^{\downarrow}$
4	10	$\langle v_{10}, v_5 \rangle^{\downarrow}$		0	8	$\langle v_8, v_3 \rangle^{\downarrow}$	$\langle v_8, v_5 \rangle^{\downarrow}$
				2	10	$\langle v_{10}, v_3 \rangle^{\downarrow}$	

(b) Lower tringle corrected edge order

Third Loop (Black)				Forth Loop (Blue)			
s_2	s_1	$\langle v_{s_1}, v_j \rangle^{\uparrow}$	$\langle v_{s_1}, v_i \rangle^{\uparrow}$	s_2	s_1	$\langle v_{s_1}, v_i \rangle^{\uparrow}$	$\langle v_{s_1}, v_j \rangle^{\uparrow}$
2	8	$\langle v_8, v_5 \rangle^{\uparrow}$	$\langle v_8, v_3 \rangle^{\uparrow}$	4	1	$\langle v_1, v_3 \rangle^{\uparrow}$	$\langle v_1, v_5 \rangle^{\uparrow}$
0	6	$\langle v_6, v_5 \rangle^{\uparrow}$	$\langle v_6, v_3 \rangle^{\uparrow}$	6	3	$\langle v_3, v_3 \rangle^{\uparrow}$	—
9	4	$\langle v_4, v_5 \rangle^{\uparrow}$	$\langle v_4, v_3 \rangle^{\uparrow}$	8	5	—	—
7	2	$\langle v_2, v_5 \rangle^{\uparrow}$	$\langle v_2, v_3 \rangle^{\uparrow}$	10	7	$\langle v_7, v_3 \rangle^{\uparrow}$	$\langle v_7, v_5 \rangle^{\uparrow}$
5	0	$\langle v_0, v_5 \rangle^{\uparrow}$	$\langle v_0, v_3 \rangle^{\uparrow}$	1	9	$\langle v_9, v_3 \rangle^{\uparrow}$	
3	9	$\langle v_9, v_5 \rangle^{\uparrow}$					

(c) Upper tringle corrected edge order

Fig. 4. We consider the case where $n = 11$ and the failed nodes are v_3 and v_5, that is, $i = 3, j = 5$. Therefore $d = 2$ and $x = y' = 4, x' = y = 5$. We use here the lower-triangle-adjacency matrix for Loop I (red) and Loop II (green) and the upper-triangle-adjacency matrix for Loop III (black) and Loop IV (blue). Loop I starts with the edge (v_7, v_5), and ends with the edge (v_{10}, v_5) and Loop II starts with the edge (v_3, v_0) and ends with the edges (v_{10}, v_3). Similarly, Loop III starts with the edge (v_5, v_8), and ends with the edge (v_5, v_9) and Loop IV starts with the edge (v_1, v_3) and ends with the edges (v_3, v_9). Loop I, IV corrects the self loop $(v_5, v_5), (v_3, v_3)$, respectively. At the end of this algorithm, $(v_5, v_3), (v_9, v_3), (v_9, v_5)$ are the uncorrected edges for the lower-triangle-adjacency matrix and $(v_3, v_5), (v_3, v_{10}), (v_5, v_{10})$ for the upper-triangle-adjacency matrix, and are marked in gray.

REFERENCES

[1] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran. Network coding for distributed storage systems. *IEEE Transactions on Information Theory*, 56(9):4539–4551, 2010.
[2] J. J. Hopfield. Neurocomputing: Foundations of research. chapter Neural Networks and Physical Systems with Emergent Collective Computational Abilities, pages 457–464. MIT Press, 1988.
[3] R. M. Roth. Maximum-rank array codes and their application to crisscross error correction. *IEEE Transactions on Information Theory*, 37(2):328–336, 1991.
[4] E. Yaakobi and J. Bruck. On the uncertainty of information retrieval in associative memories. In *ISIT*, pages 106–110, 2012.
[5] L. Yohananov and E. Yaakobi. Codes for graph erasures. In *ISIT*, pages 844–848. IEEE, 2017.