The Zero Cubes Free and Cubes Unique Multidimensional Constraints

Sagi Marcovich
Technion - Israel Institute of Technology
Haifa 3200003, Israel
sagimar@cs.technion.ac.il

Eitan Yaakobi
Technion - Israel Institute of Technology
Haifa 3200003, Israel
yaakobi@cs.technion.ac.il

***Abstract*—This paper studies two families of constraints for two-dimensional and multidimensional arrays. The first family requires that a multidimensional array will not contain a cube of zeros of some fixed size and the second constraint imposes that there will not be two identical cubes of a given size in the array. These constraints are natural extensions of their one-dimensional counterpart that have been rigorously studied recently. For both of these constraints we present conditions of the size of the cube for which the asymptotic rate of the set of valid arrays approaches 1 as well as conditions for the redundancy to be at most a single symbol. For the first family, we present an efficient encoding algorithm that uses a single symbol to encode arbitrary information into a valid array and for the second family we present a similar encoder for the two-dimensional case. The results in the paper are also extended to similar constraints where the sub-array is not necessarily a cube, but a box of arbitrary dimensions and only its volume is bounded.**

I. Introduction

Coding for two-dimensional and multidimensional arrays is a topic which attracted significant attention in the last three decades due to its various applications in different areas. This includes optical storage such as page-oriented optical memories [8], [16] and holographic storage [7]. Other applications in robotics are robot localization [19], camera localization [20], projected touchscreens [2], just to name a few, and there are several more in structured light; see e.g. [9], [15], [18], [17]. Examples of coding schemes for these applications include error-correction codes [6], constrained codes [21], pseudo random arrays and perfect maps [11], [5], codes for self locating patterns [1], and more.

This paper takes one more step in advancing the theory of coding for multidimensional and studies two special constraint families for two-dimensional and multidimensional arrays. In the first constraint, it is said that a d-dimensional array is *zero L-cubes free* if it does not contain any zero cube of volume L^d. In the second constraint, we say that W is *L-cubes unique* if it does not contain any two identical cubes of volume L^d. Only little is known on these families of codes and the goal of this paper is to rigorously study them for all values of L and d and in particular for $d = 2$, as well as to construct efficient encoding and decoding algorithms for these constraints.

The zero L-cubes free constraint was studied for the one-dimensional case in [10]. It was shown that if $L = \log_q(n) - f(n)$ where $f(n)$ satisfies that $n - 2(\log_q(n) - f(n)) = \Theta(n)$ then the redundancy of the sequences that satisfy the constraint is $\Theta(q^{f(n)})$. An encoding scheme for the binary case that uses a single redundancy bit and avoids zero-runs of length $L = \lceil \log(n) \rceil + 1$ was also proposed. The L-cubes unique constraint was studied in [3], [4] and it was shown that for values of L satisfying $L^d = \left\lfloor ad \log_q(n) \right\rfloor$ with $a > 1$, the asymptotic rate of these arrays approaches 1. For the one-dimensional case, two encoding schemes were proposed. The first one uses a single redundancy symbol and supports $L = 2\lceil \log(n) \rceil + 2$, while the second works for substrings of length $L = \lceil a \log(n) \rceil$ where $1 < a \leqslant 2$ and its asymptotic rate approaches 1.

In this paper, it is shown that for the zero L-cubes free constraint, if $L = \omega(1)$ then the asymptotic rate of the set of arrays that satisfy the constraint approaches 1 and for $L \geqslant \sqrt[d]{d \log_q(n) + \log_q(\frac{q}{q-1})}$, its redundancy is at most a single symbol. Then, an efficient algorithm for encoding L-cubes free arrays that uses a single redundancy symbol is presented for $L = \left\lceil \sqrt[d]{\left\lceil d \log_q(n) \right\rceil + 1} \right\rceil$. Note that the difference between these two values of L is at most $1 + \sqrt[d]{2}$. For the L-cubes unique constraint, it is shown that if $L \geqslant \sqrt[d]{2d \log_q(n) + \log_q(\frac{q}{q-1})}$, then the redundancy of this set of arrays is at most a single symbol. For the binary two-dimensional case, an encoding algorithm that uses a single redundancy bit is proposed which supports $L = 2\left\lceil \sqrt{\lceil 3 \log(n) \rceil + 2} \right\rceil$.

The rest of the paper is organized as follows. In Section II, the constraints that will be studied in the paper are formally defined. In Section III, we study the zero cubes free constraint and in Section IV we address the cubes unique constraint. Due to the lack of space, some of the proofs in the paper are omitted. These proofs and several more results can be found in the full version of this work in [14].

II. Definitions and Preliminaries

In this section, we formally define the notations and constraints studied in this paper. For integers $i, j \in \mathbb{N}$ such that $i \leqslant j$ we denote by $[i, j]$ the set $\{i, i+1, \ldots, j-1, j\}$. We notate by $[i]$ a shorthand for $[0, i-1]$. For a set A, let $|A|$ denote the number of elements in A. Let Σ_q denote a finite alphabet of size $|\Sigma_q| = q$. When $q = 2$, we omit the subscript q from this and from similar notations.

Let $d \in \mathbb{N}$ be an integer, let $\mathbb{N}^d$ be the d-dimensional grid, and let $\boldsymbol{v} = (v_0, v_1, \ldots, v_{d-1}) \in \mathbb{N}^d$ denote a vector of length d. For $A \subseteq \mathbb{N}^d$, a set of coordinate vectors, we denote by $\boldsymbol{v} + A$ the set

$$\{(v_0 + u_0, \ldots, v_{d-1} + u_{d-1}) \mid \boldsymbol{u} = (u_0, \ldots, u_{d-1}) \in A\},$$

and by $c \cdot A$, where $c \in \mathbb{N}$, the set

$$\{(cu_0, \ldots, cu_{d-1}) \mid \boldsymbol{u} = (u_0, \ldots, u_{d-1}) \in A\}.$$

The set $\boldsymbol{v} - A$ is defined similarly. Next, for a set $A \subseteq \mathbb{N}^d$ we denote by Σ_q^A the set of all functions from A to Σ_q. We denote

by $\bigcup_{A\subseteq\mathbb{N}^d}\Sigma_q^A$ the set of all *d-dimensional arrays*. For an integer $n\in\mathbb{N}$, we denote by $[n]^d$ the set $[n]^d=\otimes_{i=0}^{d-1}[n]$ and say that $\Sigma_q^{[n]^d}$ is the set of all *d-dimensional n-cubes*. Throughout this paper, we sometimes remove the d-dimensional prefix when using those notations if the dimension d is clear from the context. When $d=2$, we refer to d-dimensional n-cubes as n-squares. Additionally, the redundancy of a set $\mathcal{A}\subseteq\Sigma_q^{[n]^d}$ is defined as $\text{red}(\mathcal{A})=n^d-\log_q(|\mathcal{A}|)$.

Let $W\in\Sigma_q^A$ be an array and $A'\subseteq A\subseteq\mathbb{N}^d$ be sets of coordinate vectors. We denote by $W_{A'}$ the restriction of W to the coordinates in A'. When A' contains a single coordinate vector $A'=\{\boldsymbol{v}\}$ we simplify the representation and write $W_{\boldsymbol{v}}$. Next, we define a total order over $\mathbb{N}^d$.

Definition 1. *Let $\boldsymbol{u}=(u_0,\ldots,u_{d-1}),\boldsymbol{v}=(v_0,\ldots,v_{d-1})\in\mathbb{N}^d$ be two different coordinate vectors. We say that $\boldsymbol{u}<\boldsymbol{v}$ if there exists $0\leqslant s\leqslant d-1$ such that $u_s<v_s$ and for every $0\leqslant t<s$, $u_t=v_t$.*

For a finite set $A\subseteq\mathbb{N}^d$ and a vector $\boldsymbol{v}\in A$, the mapping $B_{A,q}(\boldsymbol{v})$ returns a q-ary vector of the index representation of $\boldsymbol{v}$ in A, where the vectors are ordered increasingly according to the total order presented in Definition 1. Note that the size of the mapping output is $\left\lceil\log_q(|A|)\right\rceil$. For an integer $i\in[d]$, let $\boldsymbol{e}_i\in\Sigma_2^d$ denote the i-th unit vector, i.e., a vector with *one* at its i-th bit and *zeros* elsewhere. For two vectors v,u we denote their concatenation by $v\circ u$. Additionally, we denote the bijection $MD_A:\Sigma_q^{|A|}\to\Sigma_q^A$ which transforms a sequence to its multidimensional representation under the coordinates of A, and its inverse $SD_A:\Sigma_q^A\to\Sigma_q^{|A|}$. MD_A reorders the symbols using the order of Definition 1 over the coordinates of A, i.e., the i-th symbol of the input sequence will transform to the symbol in the i-th coordinate in A. We will sometimes omit A from the notations when it is clear from the context.

Example 1. Let $d=2,n=4$, and

$$X=\begin{pmatrix}1&1&0&0\\1&0&1&1\\0&0&0&1\\0&0&0&1\end{pmatrix}\in\Sigma^{[n]^2}.$$

Then, $SD_{[n]^2}(X)=1100101100010001\in\Sigma^{16}$ (notice that $|[n]^2|=16$). Moreover, let $\boldsymbol{s}=0101101000111100\in\Sigma^{16}$, then,

$$MD_{[n]^2}(\boldsymbol{s})=\begin{pmatrix}0&1&0&1\\1&0&1&0\\0&0&1&1\\1&1&0&0\end{pmatrix}\in\Sigma^{[n]^2}.$$

Next, the main families of constraints that are studied in the paper are defined. For simplicity, we define these constraints only over arrays which are d-dimensional n-cubes.

Definition 2. *Let $W\in\Sigma_q^{[n]^d}$ be a d-dimensional array. We say that W contains a **zero** L**-cube** (or **zero** L**-square** for $d=2$) at position $\boldsymbol{v}\in[n-L+1]^d$, if $W_{\boldsymbol{v}+[L]^d}=\boldsymbol{0}$. An array W satisfies the **zero** L**-cubes free constraint** if it does not contain any zero L-cube.*

Throughout the paper, we may refer to an array satisfying the constraint in Definition 2 as a *zero L-cubes free* array.

Example 2. Let $n=5,d=2$, and

$$Y=\begin{pmatrix}1&1&0&0&1\\1&0&1&1&1\\0&0&0&1&0\\0&0&0&0&1\\0&1&0&1&0\end{pmatrix}\in\Sigma^{[n]^2}.$$

Then, Y contains two zero 2-squares, at positions $(2,0)$ and $(2,1)$. For $L>2$, Y contains no zero L-squares and thus Y satisfies the zero L-squares free constraint.

For positive integers n,q,d,L, we denote by $\mathcal{C}_{d,q}(n,L)$ the set of all arrays over $\Sigma_q^{[n]^d}$ that satisfy the zero L-cubes free constraint. The authors of [10] studied the one dimensional variation of this problem and showed that if $L=\log_q(n)-f(n)$, where $f(n)$ is a function that satisfies $n-2(\log_q(n)-f(n))=\Theta(n)$, then the redundancy of $\mathcal{C}_{1,q}(n,f(n))$ is $\Theta(q^{f(n)})$. They also proposed an encoding scheme for the binary case that uses a single redundancy bit and avoids zero-runs of length $L=\lceil\log(n)\rceil+1$.

In Section III, we analyze the cardinality of $\mathcal{C}_{d,q}(n,L)$ for any d,q, and present lower bounds for L for two cases: 1) the asymptotic rate of $\mathcal{C}_{d,q}(n,L)$ is 1, and 2) the redundancy of $\mathcal{C}_{d,q}(n,L)$ is at most a single symbol. Then, we present an algorithm that encodes arrays from $\mathcal{C}_{d,q}(n,L)$ using a single redundancy symbol, where L almost achieves the lower bound that we found for this case. Moreover, we analyze this constraint for the two-dimensional case and present tight bounds for its redundancy for every n,L.

Next, the second constraint studied in the paper is defined.

Definition 3. *Let $W\in\Sigma_q^{[n]^d}$ be a d-dimensional array. We say that W contains two **identical** L**-cubes** (or **identical** L**-squares** for $d=2$) at positions $\boldsymbol{u}\neq\boldsymbol{v}\in[n-L+1]^d$, if $W_{\boldsymbol{u}+[L]^d}=W_{\boldsymbol{v}+[L]^d}$. An array W satisfies the L**-cubes unique constraint** if it does not contain any two identical L-cubes.*

Throughout the paper, we may refer to an array that satisfies the constraint in Definition 3 as an *L-cubes unique* array.

Example 3. Let $n=5,d=2$, and

$$Z=\begin{pmatrix}1&1&0&0&1\\1&0&1&1&1\\0&0&1&1&0\\0&0&1&0&1\\0&1&0&0&1\end{pmatrix}\in\Sigma^{[n]^2}.$$

Then, Z contains two identical 3-squares at positions $(0,0)$ and $(2,2)$. However, Z contains no identical 4-squares and thus Z satisfies the 4-squares unique constraint.

Denote by $\mathcal{D}_{d,q}(n,L)$ the set of all arrays over $\Sigma_q^{[n]^d}$ that satisfy the L-cubes unique constraint. In [4], the authors analyzed the cardinality of $\mathcal{D}_{d,q}(n,L)$ and proved the following.

Theorem 4. *[4] For L that satisfies $L^d = \left\lfloor ad\log_q(n) \right\rfloor$ with $a > 1$, the asymptotic rate of $\mathcal{D}_{d,q}(n,L)$ approaches 1. Namely,*

$$\lim_{n\to\infty} \frac{\log_q(|\mathcal{D}_{d,q}(n,L)|)}{n} = 1.$$

Additionally, the authors of [3], [4] proposed two encoding schemes for the one dimensional case of the set $\mathcal{D}_1(n,L)$, which is also known as the set of *L-substring unique* sequences [12], [13]. The first scheme is applied for substrings of length $L = 2\lceil\log(n)\rceil + 2$ with a single bit of redundancy, and the second one works for substrings of length $L = \lceil a\log(n)\rceil$ for any $1 < a \leqslant 2$ and its asymptotic rate approaches 1. In Section IV, we present for all d, q a lower bound for L such that the redundancy of $\mathcal{D}_{d,q}(n,L)$ is at most 1. Then, we present an encoding scheme for the binary two-dimensional case that uses a single redundancy bit, while the value of L is far from the lower bound we found only by a factor of $\sqrt{3}$.

III. The Zero Cubes Free Constraint

First, we have the following two theorems regarding the cardinality of $\mathcal{C}_{d,q}(n,L)$.

Theorem 5. *Let $L = f(n)$ be a function of n that is not constant, i.e., $L = \omega(1)$. Then, the asymptotic rate of $\mathcal{C}_{d,q}(n,L)$ is 1. Namely,*

$$\lim_{n\to\infty} \frac{\log_q(|\mathcal{C}_{d,q}(n,L)|)}{n} = 1.$$

Theorem 6. *For an integer $L \geqslant \sqrt[d]{d\log_q(n) + \log_q(\frac{q}{q-1})}$, it holds that $|\mathcal{C}_{d,q}(n,L)| \geqslant q^{n^d-1}$. That is, $\mathrm{red}(\mathcal{C}_{d,q}(n,L)) \leqslant 1$.*

Our next goal in the paper is to provide an algorithm that encodes d-dimensional arrays over $\Sigma_q^{[n]^d}$ which satisfy the zero L-cubes constraint for

$$L = \left\lceil \sqrt[d]{\left\lceil d\log_q(n) \right\rceil + 1} \right\rceil.$$

Note that the difference between this value of L and the lower bound derived in Theorem 6 is at most $1 + \sqrt[d]{2}$. The algorithm uses a single redundancy symbol and its encoding and decoding time complexities is $O(dn^d\log(n))$.

Algorithm 1 receives a d-dimensional array $W \in \Sigma_q^{[n]^d\setminus\{(n-1)\cdot\mathbf{1}\}}$ with a single symbol missing at its corner, and outputs a cube $X \in \mathcal{C}_{d,q}(n,L)$. First, we initialize X with W and set 1 at the missing entry to mark the start of the algorithm. Then, we scan over all L-cubes in X from start to end and look for a zero L-cube. When such a cube is found, it is replaced with the non-zero cube at the position $(n-L)\cdot\mathbf{1}$ which will be referred as the *lookup-cube*. The lookup-cube is then filled with an encoding of the position of the zero cube that was found and at least one more additional zero symbol to mark the occurrence of the zero cube to the decoding process. In the case which the found cube and the lookup-cube intersect, we backup only the non-intersecting part of the lookup-cube, since we know the rest of it is zero.

In order to reconstruct $W \in \Sigma_q^{[n]^d\setminus\{(n-1)\cdot\mathbf{1}\}}$ from X, the output of Algorithm 1, we repeatedly inverse the encoding loop by recovering a position from the lookup-cube, replacing encoding its data in the lookup-cube and filling it with zeros. The decoder terminates when $X_{(n-1)\cdot\mathbf{1}} = 1$, which denotes the start of the encoding loop. The time complexity of Algorithm 1 and its decoder is $\Theta(dn^d\log(n))$.

Algorithm 1 Zero L-Cubes Free Encoding

Input: A d-dimensional array $W \in \Sigma_q^{[n]^d\setminus\{(n-1)\cdot\mathbf{1}\}}$
Output: A d-dimensional array $X \in \mathcal{C}_{d,q}(n,L)$

1: Set an array $X \in \Sigma^{[n]^d}$ with $X_{[n]^d\setminus\{(n-1)\cdot\mathbf{1}\}} = W$ and $X_{(n-1)\cdot\mathbf{1}} = 1$
2: **for** every $\boldsymbol{v} \in [n-L+1]^d$ (iterate in an increasing order) **do**
3: **if** $X_{\boldsymbol{v}+[L]^d} = \mathbf{0}$ **then**
4: **if** $A = (\boldsymbol{v}+[L]^d) \cap [n-L,n-1]^d = \emptyset$ **then**
5: Set $X_{\boldsymbol{v}+[L]^d} = X_{[n-L,n-1]^d}$
6: **else**
7: Set $\boldsymbol{y} = SD(X_{([n-L,n-1]^d\setminus A)})$
8: Set $X_{\boldsymbol{v}+[L]^d\setminus A} = MD(\boldsymbol{y})$
9: **end if**
10: Set $X_{[n-L,n-1]^d} = MD(B_{[n]^d,q}(\boldsymbol{v}+\boldsymbol{e}_d) \circ 0^{L^d - \left\lceil d\log_q(n)\right\rceil})$
11: **end if**
12: **end for**

Lastly, we note that Algorithm 1 works also for the one-dimensional case, which achieves the same value of L as the one achieved by the algorithm presented in [10]. However the complexity of the algorithm in [10] is $\Theta(n)$ while the complexity of Algorithm 1 for the one dimensional case is $\Theta(n\log(n))$. The correctness of Algorithm 1 appears in the full version of this work in [14]

Example 4. Let $n = 7, L = 3$, and the input array is

$$W = \begin{pmatrix} 0&0&1&0&0&0&1\\ 0&0&0&0&1&1&1\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1\\ 0&1&0&0&1&0&0\\ 0&0&0&1&0&0& \end{pmatrix}.$$

The algorithm appends *one* at the missing entry to initialize X. Then, it iterates the coordinates in an increasing order until a zero 3-square is found. In the following figures, the lookup-square and the found zero square are highlighted.

$$X = \begin{pmatrix} 0&0&1&0&0&0&1\\ 0&0&0&0&1&1&1\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1\\ 0&1&0&0&1&0&0\\ 0&0&0&1&0&0&1 \end{pmatrix}.$$

A zero 3-square found in $\boldsymbol{v} = (1,0)$. It is replaced with the lookup-square, and the latter is filled with the encoding

of $\boldsymbol{v}+\boldsymbol{e}_2=(1,0)+(0,1)=(1,1)$ using six bits, which is $B((1,1))=000100$, and appending three more zeros to have a 3-square.

$$X=\begin{pmatrix} 0&0&1&0&0&0&1\\ 0&0&1&0&1&1&1\\ 1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&1&0&0&1&0&0\\ 0&0&0&1&0&0&0 \end{pmatrix}$$

Next, a zero 3-square found in $\boldsymbol{v}=(2,3)$. It intersects with the lookup-square at positions $A=\{(4,4),(4,5)\}$. Hence, the algorithm fills only the non-intersecting part of $X_{(2,3)+[3]^2}$ with the non-intersecting portion of the lookup-square, $\boldsymbol{y}=0100000$. The lookup-square is filled with the encoding of $\boldsymbol{v}+\boldsymbol{e}_2=(2,3)+(0,1)=(2,4)$, that is, $B((2,4))=010010$, appended by zeros.

$$X=\begin{pmatrix} 0&0&1&0&0&0&1\\ 0&0&1&0&1&1&1\\ 1&0&0&0&1&0&0\\ 0&0&1&0&0&0&0\\ 0&0&0&0&0&1&0\\ 0&1&0&0&0&1&0\\ 0&0&0&1&0&0&0 \end{pmatrix}$$

The algorithm finishes iterating the entries of X without finding additional zero 3-squares. The result is indeed a zero 3-square free array.

Redundancy Analysis for the Two-Dimensional Case:

The proof of the next theorem is given in [14] by deriving lower and upper bounds for the cardinality of $\mathcal{C}_{2,q}(n,L)$ using various enumeration techniques.

Theorem 7. *There exist constants C_1, C_2 such that for any positive integer n it holds that*

$$C_2\frac{(n-2L)^2}{q^{L^2}}\leqslant red(\mathcal{C}_{2,q}(n,L))\leqslant C_1\frac{n^2}{q^{L^2}}.$$

The next corollary follows immediately.

Corollary 8. *Let n, L be integers such that $n-2L=\Theta(n)$. Then,*

$$red(\mathcal{C}_{2,q}(n,L))=\Theta\left(\frac{n^2}{q^{L^2}}\right).$$

IV. The Cubes Unique Constraint

First, we derive a lower bound for L which assures that the redundancy of $\mathcal{D}_{d,q}(n,L)$ is at most a single symbol.

Theorem 9. *For $L\geqslant\sqrt[d]{2d\log_q(n)+\log_q(\frac{q}{q-1})}$, it holds that $|\mathcal{D}_{d,q}(n,L)|\geqslant q^{n^d-1}$. That is,* $\mathrm{red}(\mathcal{D}_{d,q}(n,L))\leqslant 1$.

Next, we focus on the two-dimensional case and present a generic encoding algorithm that uses a single redundancy bit in order to encode binary n-squares that are L-squares unique, for

$$L=2\left\lceil\sqrt{\lceil 3\log(n)\rceil+2}\right\rceil.$$

Note that this value of L is far from the value derived in Theorem 9 only by roughly a factor of $\sqrt{3}$. For simplicity, we sometimes omit ceiling notations in the rest of this section.

We introduce first a new type of two-dimensional arrays, denoted as *bottom semi squares*, or *semi squares* in short. For a vector $\boldsymbol{v}\in[n]^2$, the set $A=[n]^2\setminus(\boldsymbol{v}+[n]^2)$ contains coordinates of a semi square with a corner at $\boldsymbol{v}$. Hence, we say that $X\in\Sigma_q^A$ is an $(n,\boldsymbol{v})$-semi square.

Let X be an $(n,\boldsymbol{v})$-semi square for $\boldsymbol{v}\in[n]^2$, let t be an integer, and let Y be a $(t,\boldsymbol{u})$-semi square for $\boldsymbol{u}\in[t]^2$. We denote by $X\circ Y$ the concatenation of X and Y which is defined by placing Y at position $\boldsymbol{v}$ of X, and restricting the result to the coordinates in $[n]^2$. It follows that $X\circ Y$ is a $(n,\boldsymbol{v}+\boldsymbol{u})$-semi square if and only if for every $i\in[2]$, $u_i=0$ or $v_i+t\geqslant n$.

Example 5.

$$X=\begin{pmatrix} 1&1&0&0&1\\ 1&0&1&1&1\\ 0&0&0&1&0\\ 0&0&&&\\ 0&1&&& \end{pmatrix}\in\Sigma^{[5]^2\setminus((3,2)+[5]^2)}$$

is a $(5,\boldsymbol{v})$-semi square for $\boldsymbol{v}=(3,2)$, and

$$Y=\begin{pmatrix} 0&0&1\\ 1&&\\ 0&& \end{pmatrix}\in\Sigma^{[3]^2\setminus((1,1)+[3]^2)}$$

is a $(3,\boldsymbol{u})$-semi square for $\boldsymbol{u}=(1,1)$. Then, the concatenation $X\circ Y$ is a semi $(5,\boldsymbol{v}+\boldsymbol{u})$-square,

$$X\circ Y=\begin{pmatrix} 1&1&0&0&1\\ 1&0&1&1&1\\ 0&0&0&1&0\\ 0&0&0&0&1\\ 0&1&1&& \end{pmatrix}.$$

Definition 10. *Let X be an $(n,\boldsymbol{v})$-semi square for $\boldsymbol{v}\in[n]^2$, such that $\boldsymbol{v}\neq\boldsymbol{0}$. We denote by $CR(X)$ the iterative self-concatenation of X to an n-square, that is,*

$$CR(X)=X^{\left\lceil\frac{n}{v_{\min}}\right\rceil},$$

where $v_{\min}$ is the smallest entry of $\boldsymbol{v}$ that is not 0.

It can be shown by induction that after m concatenations, X^m is a $(n,m\cdot\boldsymbol{v})$-semi square, since $v_i+n\geqslant n$ for every $i\in[2]$. Thus, the self-concatenation of an $(n,\boldsymbol{v})$-semi square for every $\boldsymbol{v}\neq\boldsymbol{0}$ is defined properly, and in fact an n-square since for every $i\in[2]$, $\left\lceil\frac{n}{v_{\min}}\right\rceil\cdot v_i\geqslant n$.

Example 6. Let X,Y from Example 5. Then,

$$CR(X)=\begin{pmatrix} 1&1&0&0&1\\ 1&0&1&1&1\\ 0&0&0&1&0\\ 0&0&1&1&0\\ 0&1&1&0&1 \end{pmatrix},CR(Y)=\begin{pmatrix} 0&0&1\\ 1&0&0\\ 0&1&0 \end{pmatrix}.$$

Additionally, we define the matching *upper semi squares*, described by a coordinate vectors set of $A=[n]^2\setminus(\boldsymbol{v}-[n]^2)$ for $\boldsymbol{v}\in[n]^2$. We similarly define concatenation and self-concatenation to an n-square of upper semi squares.

Algorithm 2 receives a two-dimensional array $W\in\Sigma^{[n]^2\setminus\{(0,0)\}}$, an n-square with a single missing entry, and outputs an n-square $X\in\mathcal{D}_2(n,L)$. The algorithm consists of two main procedures, elimination and expansion.

Algorithm 2 L-Squares Unique Encoding

Input: A two-dimensional array $W \in \Sigma^{[n]^2 \setminus \{(0,0)\}}$
Output: An n-square $X \in \mathcal{D}_2(n, L)$

Set a square $X \in \Sigma^{[n]^2}$ with $X_{0,0} = 0, X_{[n]^2 \setminus \{(0,0)\}} = W$
Denote $(i_m, j_m) = (n, 0)$, append $X_{(i_m, j_m)+[k]^2} = P_M$
First part - Elimination
while at least one of the occurences in cases 1,2,3 exists **do**
    case 1: A k-square equals to P_M exists at $(i_1, j_1) < (i_m, j_m)$
        Remove square $X_{(i_1, j_1)+[k]^2}$
        Set $\boldsymbol{v} = 101 \circ B_{[n]^2}(i_1, j_1) \circ 1^{k^2 - 2\log(n) - 3}$, insert $MD_{[k]^2}(\boldsymbol{v})$ at $X_{0,0}$
    case 2: Identical L-squares exist at $(i_1, j_1) < (i_2, j_2)$
        Remove square $X_{(i_1, j_1)+[L]^2}$
        Set $\boldsymbol{v} = 100 \circ B_{[n]^2}(i_1, j_1) \circ B_{[n]^2}(i_2, j_2) \circ 1^{3k^2 - 4\log(n) - 3}$, insert $MD_{[k]\times[3k]}(\boldsymbol{v})$ at $X_{0,0}$
    case 3: Identical (k, L)-rectangles exist at $(i_1, j_1) < (i_2, j_2)$, where

$$(i_2, j_2) \in I = (\{i_m - k\} \times [j_m - k, n-1]) \cup (\{i_m\} \times [j_m - k - 1]),$$

        Remove rectangle $X_{(i_1, j_1)+[k]\times[L]}$
        Set $\boldsymbol{v} = 11 \circ B_{[n]^2}(i_1, j_1) \circ B_I(i_2, j_2)$, insert $MD_{[k]^2}(\boldsymbol{v})$ at $X_{0,0}$
    If cases 2 or 3 were executed, decrement (i_m, j_m) by a k-square
end while
If $|X| \geqslant n^2$, return $X_{[n]^2}$
Second part - Expansion
while $|X| < n^2$ **do**
    Set indexes (i_e, j_e) to point to the next missing k-square in X
    Let $I_e = ((i_e, j_e) - [k]^2) \cap [n]^2$. Set

$$\mathcal{S} = \{X_{(i,j)+[k]^2} \mid (i, j) \notin I_e\} \cup \{CR(X_{(i,j)+[k]^2}) \mid (i, j) \in I_e\}$$

    Pick $Y \in \Sigma^{[k]^2} / \mathcal{S}$ and set $X_{(i_e, j_e)+[k]^2} = Y$
end while
Return X

First, we initialize X with W and set 0 at the missing entry to mark the start of the elimination. Then, we append to X a marker $(L/2)$-square that will mark the transition between the elimination and the expansion parts of the encoder. At the elimination part, we iteratively shorten X by an $(L/2)$-square at a time by eliminating one of the two occurrences in X: 1. two identical L-squares, 2. two identical rectangles of size $[L/2] \times [L]$ (notated for the rest of this section as $(L/2, L)$-rectangles) where one of them is at the bottom of X. Likewise, we make sure that the marker $(L/2)$-square appears only once in X. Later, at the expansion part, we enlarge X to an n-square by iteratively appending $(L/2)$-squares while making sure that no new identical L-squares are created.

For convenience, we denote for the rest of this section $k = L/2 = \left\lceil \sqrt{3\log(n) + 2} \right\rceil$, and define the *marker k-square* denoted as P_M as the following square,

$$P_M = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & \mathbf{0} & \\ 0 & & & \end{pmatrix} \in \Sigma^{[k]^2}.$$

We also explain the notion of removal and insertion of squares with granularity. We assume that $n \bmod k = 0$, and let $X \in \Sigma^{[n]^2 \setminus A_t}$ where A_t contains the coordinates of the last t k-squares in $[n]^2$, i.e., $A_t = ([i_t - k, n-1] \times [j_t, n-1]) \cup ([i_t, n-1] \times [n])$ where $(i_t, j_t) = (n - \lfloor tk/n \rfloor, n - (tk \bmod n))$. We look at X as a grid of k-squares, and allow only removals and insertions in granularity of k-square units. Removal or insertion actions on a k-square at an aligned position $(\hat{i} \cdot k, \hat{j} \cdot k)$ are performed by transforming X to a k-rows array with coordinates $[k] \times [|X|/k]$, executing the action on the $(i \cdot k + j)$-th k-square like in a one-dimensional array, and transforming back to a grid of k-squares.

However, during the elimination part of the algorithm we sometimes need to remove a k-square from an unaligned position (i, j). Such an action is done by finding the closest aligned position $(\hat{i} \cdot k, \hat{j} \cdot k)$, then replacing the data of the non-intersecting parts of $X_{(\hat{i} \cdot k, \hat{j} \cdot k)+[k]^2}$ and $X_{(i,j)+[k]^2}$, and finally removing the aligned k-square at position $(\hat{i} \cdot k, \hat{j} \cdot k)$. This is a technical procedure that can be transparent to the reader of the encoder in Algorithm 2. All of the above ensures that appended k-squares, and specifically the marker k-square, are not trimmed or modified accidentally as a result of unrelated removal of insertion actions. The correctness of the Algorithm 2 as well as the explanation for the decoding can also be found in [14].

REFERENCES

[1] A. M. Bruckstein, T. Etzion, R. Giryes, N. Gordon, R. J. Holt, and D. Shuldiner, "Simple and robust binary self-location patterns," *IEEE Transactions on Information Theory*, vol. 58, no. 7, pp. 4884–4889, 2012.

[2] J. Dai and C. R. Chung, "Touchscreen everywhere: On transferring a normal planar surface to a touch-sensitive display," *IEEE Transactions on Cybernetics*, vol. 44, no. 8, pp. 1383–1396, 2014.

[3] O. Elishco, R. Gabrys, M. Mèdard, and E. Yaakobi, "Repeat free codes," in *Proc. of the IEEE International Symposium of Information Theory*, Paris, France, 2019, pp. 932–936.

[4] O. Elishco, R. Gabrys, E. Yaakobi, and M. Médard, "Repeat-free codes," *IEEE Transactions on Information Theory*, vol. 67, no. 9, pp. 5749–5764, 2021.

[5] T. Etzion, "Constructions for perfect maps and pseudorandom arrays," *IEEE Transactions on Information Theory*, vol. 34, no. 5, pp. 1308–1316, 1988.

[6] T. Etzion and E. Yaakobi, "Error-correction of multidimensional bursts," *IEEE Transactions on Information Theory*, vol. 55, no. 3, pp. 961–976, 2009.

[7] C. Gu, J. Hong, I. McMichael, R. Saxena, and F. Mok, "Cross-talk-limited storage capacity of volume holographic memory," *J. Opt. Soc. Am. A*, vol. 9, no. 11, pp. 1978–1983, Nov 1992.

[8] J. F. Heanue, M. C. Bashaw, and L. Hesselink, "Volume holographic storage and retrieval of digital data," *Science*, vol. 265, no. 5173, pp. 749–752, 1994.

[9] Y. C. Hsieh, "Decoding structued light patters for three-dimensional imaging systems," *Pattern Recognition*, vol. 34, pp. 343–349, 2001.

[10] M. Levy and E. Yaakobi, "Mutually uncorrelated codes for DNA storage," *IEEE Transactions on Information Theory*, vol. 65, no. 6, pp. 3671–3691, 2019.

[11] F. J. MacWilliams and N. J. A. Sloane, "Pseudo-random sequences and arrays," in *Proceedings of the IEEE*, vol. 64, no. 12, 1976, pp. 1715–1729.

[12] S. Marcovich and E. Yaakobi, "Reconstruction of strings from their substrings spectrum," in *"Proc. of the IEEE International Symposium on Information Theory, Los Angeles, USA"*, 2020, pp. 658–663.

[13] ——, "Reconstruction of strings from their substrings spectrum," *Submitted to: IEEE Transactions on Information Theory*, 2020.

[14] ——, "The zero cubes free and cubes unique multidimensional constraints," *to: IEEE Transactions on Information Theory*, 2021.

[15] R. A. Morano, C. Ozturk, R. Conn, S. Dubin, S. Zietz, and J. Nissanov, "Structured light using pseudorandom codes," *IEEE Transactions Pattern Analysis and Machine Intelligence*, vol. 20, pp. 322–327, 1998.

[16] M. A. Neifeld and M. McDonald, "Error correction for increasing the usable capacity of photorefractive memories," *Opt. Lett.*, vol. 19, no. 18, pp. 1483–1485, Sep 1994.

[17] J. Pages, J. Salvi, C. Collewet, and J. Forest, "Optimised de Bruijn patterns for one-shot shape acquisition," *Image and Vision Computing*, vol. 23, pp. 707–720, 2005.

[18] J. Salvi, S. Fernandez, T. Pribanic, and X. Llado, "State of the art in structured light patterns for surface profilometry," *Pattern Recognition*, vol. 23, pp. 2666–2680, 2010.

[19] E. R. Scheinerman, "Determining planar location via complement-free de brujin sequences using discrete optical sensors," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 883–889, 2001.

[20] I. Szentandrasi, M. Zacharias, J. Havel, A. Herout, M. Dubská, and R. Kajan, "Uniform marker fields: Camera localization by orientable de bruijn tori," *IEEE International Symposium on Mixed and Augmented Reality*, pp. 319–320, 2012.

[21] I. Tal and R. M. Roth, "Bounds on the rate of 2-d bit-stuffing encoders," *IEEE Transactions on Information Theory*, vol. 56, no. 6, pp. 2561–2567, 2010.