Almost Optimal Construction of Functional Batch Codes Using Hadamard Codes

Lev Yohananov
Dept. of Computer Science
Technion-Israel Institute of Technology
Haifa 3200003, Israel
Email: levyohananov@campus.technion.ac.il

Eitan Yaakobi
Dept. of Computer Science
Technion-Israel Institute of Technology
Haifa 3200003, Israel
Email: yaakobi@cs.technion.ac.il

***Abstract*—A *functional k-batch* code of dimension s consists of n servers storing linear combinations of s linearly independent information bits. Any multiset request of size k of linear combinations (or requests) of the information bits can be recovered by k disjoint subsets of the servers. The goal under this paradigm is to find the minimum number of servers for given values of s and k. A recent conjecture states that for any $k = 2^{s-1}$ requests the optimal solution requires $2^s - 1$ servers. This conjecture is verified for $s \leqslant 5$ but previous work could only show that codes with $n = 2^s - 1$ servers can support a solution for $k = 2^{s-2} + 2^{s-4} + \left\lfloor \frac{2^{s/2}}{\sqrt{24}} \right\rfloor$ requests. This paper reduces this gap and shows the existence of codes for $k = \lfloor \frac{2}{3} 2^{s-1} \rfloor$ requests with the same number of servers. Another construction in the paper provides a code with $n = 2^{s+1} - 2$ servers and $k = 2^s$ requests, which is an optimal result. These constructions are mainly based on Hadamard codes and equivalently provide constructions for *parallel Random I/O (RIO)* codes.**

I. Introduction

Motivated by several applications for load-balancing in storage and cryptographic protocols, *batch codes* were first proposed by Ishai et al. [7]. A batch code encodes a length-s string $\boldsymbol{x}$ into n strings, where each string corresponds to a *server*, such that each batch request of k different bits (and more generally symbols) from $\boldsymbol{x}$ can be decoded by reading at most t bits from every server. This decoding process corresponds to the case of a single-user. There is an extended variant for batch codes [7] which is intended for a multi-user application instead of a single-user setting, known as the *multiset batch codes*. Such codes have k different users and each requests a single data item. Thus, the k requests can be represented as a *multiset* of the bits since the requests of different users may be the same, and each server can be accessed by at most one user.

A special case of multiset batch codes, referred as *primitive batch codes*, is when each server contains only one bit. The goal of this model is to find, for given s and k, the smallest n such that a primitive batch code exists. This problem was considered in several papers; see e.g. [1], [2], [7], [8], [13]. By setting the requests to be a multiset of linear combinations of the s information bits, a batch code is generalized into a *functional batch code* [17]. Again, given s and k, the goal is to find the smallest n for which a functional k-batch code exists.

Mathematically speaking, an $FB\text{-}(n,s,k)$ functional k-batch code (and in short $FB\text{-}(n,s,k)$ code) of dimension s consists of n servers storing linear combinations of s linearly independent information bits. Any multiset of size k of linear combinations from the linearly independent information bits, can be recovered by k disjoint subsets of servers. If all the k linear combinations are the same, then the servers form an $FP\text{-}(n,s,k)$ *functional k-Private Information Retrieval (PIR)* code (and in short $FP\text{-}(n,s,k)$ code). Clearly, an $FP\text{-}(n,s,k)$ code is a special case of an $FB\text{-}(n,s,k)$ code. It was shown [17] that functional k-batch codes are equivalent to the so-called linear *parallel random I/O (RIO) codes*, where RIO codes were introduced by Sharon and Alrod [10], and their parallel variation was studied in [11], [12]. Therefore, all the results of this paper hold also for parallel RIO codes.

The value $FP(s,k), FB(s,k)$ is defined to be the minimum number of servers required for the existence of an $FP\text{-}(n,s,k), FB\text{-}(n,s,k)$ code, respectively. Several upper and lower bounds can be found in [17] on these values. It was also conjectured in [17] that for $k = 2^{s-1}$, the length of an optimal functional batch codes is $2^s - 1$, that is, $FB(s, k = 2^{s-1}) = 2^s - 1$. Indeed, in [15] this conjecture was proven for $s = 3, 4$, and in [17], by using a computer search, it was verified also for $s = 5$. However, the best known result for $s > 5$ only provides a construction of $FB\text{-}(2^s - 1, s, 2^{s-2} + 2^{s-4} + \left\lfloor \frac{2^{s/2}}{\sqrt{24}} \right\rfloor)$ codes [17]. This paper significantly improves this result and reduces the gap between the conjecture statement and the best known construction. In particular, a construction of $FB\text{-}(2^s - 1, s, \lfloor \frac{2}{3} \cdot 2^{s-1} \rfloor)$ codes is given. This construction can be extended for general $FB\text{-}(2^s + \lceil (3\alpha - 2) \cdot 2^{s-2} \rceil - 1, s, \lfloor \alpha \cdot 2^{s-1} \rfloor)$ codes for all $2/3 \leqslant \alpha \leqslant 1$. Another result that can be found in [17] states that $FP(s, 2^s) \leqslant 2^{s+1} - 2$. In this case, the lower bound is the same, i.e., this result is optimal [5]. In the extended version of this work [16], it is shown that this optimality holds not only for functional PIR codes but also for the more challenging case of functional batch codes, that is, $FB(s, 2^s) = 2^{s+1} - 2$. All the results in the paper are achieved using a generator matrix G of a Hadamard codes [3] of length 2^s and dimension s, where the matrix's columns correspond to the servers of the $FB\text{-}(n,s,k)$ code.

The rest of the paper is organized as follows. In Section II, we formally define functional k-batch codes and summarize the main results of the paper. In Section III, we show a construction of $FB\text{-}(2^s - 1, s, \lfloor \frac{2}{3} \cdot 2^{s-1} \rfloor)$. Due to the lack of space, most of the proofs in the paper are omitted, however they can be found in the extended version of this work [16].

II. Definitions

For a positive integer n define $[n] = \{0, 1, \ldots, n-1\}$. All vectors and matrices in the paper are over $\mathbb{F}_2$. We follow the definition of functional batch codes as it was first defined in [17].

Definition 1. *A **functional k-batch code** of length n and dimension s consists of n servers and s information bits $x_0, x_1, \ldots, x_{s-1}$ and is represented by a linear code with an $s \times n$ generator matrix $G = [\boldsymbol{g}_0, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_{n-1}]$ in $\mathbb{F}_2^{s \times n}$ in which the vector $\boldsymbol{g}_j$ has ones in some positions $i_0, i_1, \ldots, i_{\ell-1}$ if and only if the j-th server stores the linear combination $x_{i_0} + x_{i_1} + \cdots + x_{i_{\ell-1}}$. Using this matrix representation, a functional k-batch code is an $s \times n$ generator matrix G, such*

978-1-5386-8209-8/21/$31.00 ©2021 IEEE

that for any k request vectors $\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_{k-1} \in \mathbb{F}_2^s$ (not necessarily distinct), there are k pairwise disjoint subsets of columns in G, denoted by $R_0, R_1, \ldots, R_{k-1}$, such that the sum of the column vectors whose indices are in R_j is equal to the ***request vector*** *$\boldsymbol{v}_j$. The set of all recovery sets $R_i, i \in [k]$, is called a* ***solution*** *for the k request vectors.*

A functional k-batch code of length n and dimension s over $\mathbb{F}_2^s$ is denoted by FB-(n, s, k). Every request of k vectors will be stored as columns in a matrix M which is called the ***request matrix*** *or simply the* ***request****.*

A **functional k-PIR code** [17] of length n and dimension s, denoted by FP-(n, s, k), is a special case of FB-(n, s, k) in which all the request vectors are identical. We first review some preliminary results on the parameters of FB-(n, s, k) and FP-(n, s, k) codes which are relevant to our work. For that, another definition is presented.

Definition 2. *Denote by $FB(s, k), FP(s, k)$ the minimum length n of any FB-(n, s, k), FP-(n, s, k) code, respectively.*

Most of the following results on $FB(s, k)$ and $FP(s, k)$ can be found in [17], while the result in (c) was verified for $s = 3, 4$ in [15].

Theorem 3. *For positive integers s and t, the following properties hold:*

(a) $FP(s, 2^{s-1}) = 2^s - 1$.
(b) $FP(st, 2^s) \leqslant 2t(2^s - 1)$.
(c) For $s \leqslant 5$ it holds that $FB(s, 2^{s-1}) = 2^s - 1$.
(d) An FB-$(2^s - 1, s, 2^{s-2} + 2^{s-4} + \left\lfloor \frac{2^{s/2}}{\sqrt{24}} \right\rfloor)$ *code exists.*
(e) For a fixed k it holds that $\lim_{s\to\infty} \frac{FB(s,k)}{s} \geqslant \frac{k}{\log(k+1)}$.

Note that the result from Theorem 3(d) improves upon the result of FB-$(2^s - 1, s, 2^{s-2} + 2^{s-4} + 1)$ functional batch codes which was derived from a WOM codes construction by Godlewski [6]. This is the best known result with respect to the number of queries when the number of information bits is s and the number of encoded bits is $2^s - 1$.

The goal of this paper is to improve some of the results summarized in Theorem 3. The result in (c) holds for $s \leqslant 5$, and it was conjectured in [17] that it holds for all positive values of s.

Conjecture 1. *[17] For all $s > 5$, $FB(s, 2^{s-1}) = 2^s - 1$.*

The reader can notice the gap between Conjecture 1 and the result in Theorem $3(d)$. More precisely, [17] assures that an FB-$(2^s - 1, s, 2^{s-2} + 2^{s-4} + \left\lfloor \frac{2^{s/2}}{\sqrt{24}} \right\rfloor)$ code exists, and the goal is to determine whether an FB-$(2^s - 1, s, 2^{s-1})$ code exists. This paper takes one more step in establishing this conjecture. Specifically, the best known value of the number of requested bits k is improved for the case of s information bits and $2^s - 1$ encoded bits. The next theorem summarizes the contributions of this work. Note that only the proof of the first claim of Theorem 4, which is the main contribution of this work, is presented, while the complete proofs of the other two claims can be found in the extended version of the paper [16].

Theorem 4. *For a positive integer s, the following constructions exist:*

(a) A construction of FB-$(2^s - 1, s, \lfloor \frac{2}{3} \cdot 2^{s-1} \rfloor)$ *codes.*
(b) A construction of FB-$(2^s + \lceil (3\alpha - 2) \cdot 2^{s-2} \rceil - 1, s, \lfloor \alpha \cdot 2^{s-1} \rfloor)$ *codes where $2/3 \leqslant \alpha \leqslant 1$.*
(c) A construction of FB-$(2^{s+1} - 2, s, 2^s)$ *codes.*

We now explain the improvements of the results of Theorem 4. The construction in Theorem $4(a)$ improves upon the result from Theorem $3(d)$, where the supported number of requests increases from $\frac{1}{2}2^{s-1} + 2^{s-4} + \left\lfloor \frac{2^{s/2}}{\sqrt{24}} \right\rfloor$ to $\lfloor \frac{2}{3} \cdot 2^{s-1} \rfloor$. Moreover, according to the second result of Theorem $3(b)$, if $t = 1$ then $FP(s, 2^s) \leqslant 2^{s+1} - 2$. Based on the result in [5] it holds that $FP(s, 2^s) \geqslant 2^{s+1} - 2$. Therefore, $FP(s, 2^s) = 2^{s+1} - 2$. The construction in Theorem $4(c)$ extends this result to functional batch codes by showing that $FB(s, 2^s) \leqslant 2^{s+1} - 2$, and again, combining the result from [5], it is deduced that $FB(s, 2^s) = 2^{s+1} - 2$. Note that by taking $\alpha = 2/3$ in the result of Theorem $4(b)$, we immediately get the result of (a). However, for simplicity of the proof, we only show here the construction for (a) separately, while its extension appears in [16].

A special family of matrices that will be used extensively in the paper are the generator matrices of Hadamard codes [3], as defined next.

Definition 5. *A matrix $G = [\boldsymbol{g}_0, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_{2^s-1}]$ of order $s \times 2^s$ over $\mathbb{F}_2$ such that $\{\boldsymbol{g}_0, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_{2^s-1}\} = \mathbb{F}_2^s$ is called a Hadamard generator matrix and in short HG-matrix.*

We will use HG-matrices as the generator matrices of the linear codes that will provide the constructions used in establishing Theorem 4. More specifically, given a linear code defined by a generator HG-matrix G of order $s \times n$ and a request M of order $s \times k$, we will show an algorithm that finds a solution for M. This solution will be obtained by rearranging the columns of G and thereby generating a new HG-matrix G'. This solution is obtained by showing all the disjoint recovery sets for the request M, with respect to indices of columns of G'. Although such a solution is obtained with respect to G' instead of G, it can be easily adjusted to G by relabeling the indices of the columns. Thus, any HG-matrix whose column indices are partitioned to recovery sets for M provides a solution. Note that HG-matrices store the all-zero column vector. Such a vector will help us to simplify the construction of the algorithm and will be removed at the end of the algorithm. By a slight abuse of notation, throughout the paper -1 is referred to $n - 1$. We now show another important definition.

Definition 6. *Let $M = [\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_{n/2-1}]$ be a request of order $s \times n/2$, where $n = 2^s$. The matrix M has a* ***Hadamard solution*** *if there exists an HG-matrix $G = [\boldsymbol{g}_0, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_{n-1}]$ of order $s \times n$ such that for all $i \in [n/2]$, $\boldsymbol{v}_i = \boldsymbol{g}_{2i-1} + \boldsymbol{g}_{2i}$. In this case, we say that G is a Hadamard solution for M.*

III. A Construction of FB-$(2^s - 1, s, \lfloor \frac{2}{3} \cdot 2^{s-1} \rfloor)$ Codes

In this section a construction of FB-$(2^s - 1, s, \lfloor \frac{2}{3} \cdot 2^{s-1} \rfloor)$ codes is presented. Let the request M be denoted by $M = [\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_{\lfloor \frac{2}{3} \cdot 2^{s-1} \rfloor - 1}]$. Let $\boldsymbol{e} = (0, 0, \ldots, 0, 1) \in \mathbb{F}_2^s$ be the unit vector with 1 at its last index. The solution for the request M will be derived by using two algorithms as will be presented in this section. We start with several definitions and tools that will be used in the first algorithm.

Definition 7. *Three sets $\mathcal{G},\mathcal{B},\mathcal{R}\subseteq[2^{s-1}]$ are called a **triple-set** (the good, the bad, and the redundant), and are denoted by $(\mathcal{G},\mathcal{B},\mathcal{R})$, if the following properties hold,*

$$\mathcal{G}\subseteq\left[\left\lfloor\frac{2}{3}\cdot 2^{s-1}\right\rfloor\right],\mathcal{B}=\left[\left\lfloor\frac{2}{3}\cdot 2^{s-1}\right\rfloor\right]\setminus\mathcal{G},$$
$$\mathcal{R}=[2^{s-1}]\setminus\Big(\mathcal{G}\cup\mathcal{B}\cup\{2^{s-1}-1\}\Big).$$

*Given a matrix $M=[\boldsymbol{v}_0,\boldsymbol{v}_1,\ldots,\boldsymbol{v}_{\lfloor\frac{2}{3}\cdot 2^{s-1}\rfloor-1}]$ of order $s\times\lfloor\frac{2}{3}\cdot 2^{s-1}\rfloor$, the matrix $\mathcal{M}(\mathcal{G},\mathcal{B},\mathcal{R})=[\boldsymbol{w}_0,\boldsymbol{w}_1,\ldots,\boldsymbol{w}_{2^{s-1}-1}]$ of order $s\times 2^{s-1}$ is referred as a **triple-matrix of** M if it holds that*

$$\boldsymbol{w}_t=\begin{cases}\boldsymbol{v}_t & t\in\mathcal{G}\\ \boldsymbol{v}_t+\boldsymbol{e} & t\in\mathcal{B}\\ \boldsymbol{e} & t\in\mathcal{R}\end{cases}.$$

Note that, we did not demand anything about the vector $\boldsymbol{w}_{2^{s-1}-1}$, i.e., it can be any binary vector of length s. Furthermore, by Definition 7, the set $\mathcal{B}$ uniquely defines the triple-set $(\mathcal{G},\mathcal{B},\mathcal{R})$. We proceed with the following claim.

Claim 1 *For any triple-set $(\mathcal{G},\mathcal{B},\mathcal{R})$ if $|\mathcal{B}|\leqslant\lfloor\frac{1}{3}\cdot 2^{s-1}\rfloor$ then $|\mathcal{B}|\leqslant|\mathcal{R}|$.*

As mentioned above, our strategy is to construct two algorithms. We start by describing the first one. This algorithm receives as an input the request M and outputs a set $\mathcal{B}$ and a Hadamard-solution for some triple-matrix $\mathcal{M}(\mathcal{G},\mathcal{B},\mathcal{R})$ of M. Using the matrix $\mathcal{M}(\mathcal{G},\mathcal{B},\mathcal{R})$, it will be shown how to derive the solution for M. This connection is established in the next lemma. For the rest of this section we denote $n=2^s$ and for our ease of notations both of them will be used.

Lemma 8. *If there is a Hadamard solution for $\mathcal{M}(\mathcal{G},\mathcal{B},\mathcal{R})$ such that $|\mathcal{B}|\leqslant\lfloor\frac{1}{3}\cdot 2^{s-1}\rfloor$, then there is a solution for $M=[\boldsymbol{v}_0,\boldsymbol{v}_1,\ldots,\boldsymbol{v}_{\lfloor\frac{2}{3}\cdot 2^{s-1}\rfloor-1}]$.*

Proof: Let the HG-matrix $G=[\boldsymbol{g}_0,\boldsymbol{g}_1,\ldots,\boldsymbol{g}_{n-1}]$ be a Hadamard solution for $\mathcal{M}(\mathcal{G},\mathcal{B},\mathcal{R})$. Our goal is to form all disjoint recovery sets R_t for $t\in\mathcal{G}\cup\mathcal{B}=[\lfloor\frac{2}{3}\cdot 2^{s-1}\rfloor]$ for M. Since G is a Hadamard solution for $\mathcal{M}(\mathcal{G},\mathcal{B},\mathcal{R})$, for all $t\in[2^{s-1}]$, it holds that $\boldsymbol{w}_t=\boldsymbol{g}_{2t-1}+\boldsymbol{g}_{2t}$. By definition of $\mathcal{M}(\mathcal{G},\mathcal{B},\mathcal{R})$

$$\boldsymbol{w}_t=\begin{cases}\boldsymbol{v}_t & t\in\mathcal{G}\\ \boldsymbol{v}_t+\boldsymbol{e} & t\in\mathcal{B}\\ \boldsymbol{e} & t\in\mathcal{R}\end{cases}.$$

Thus, if $t\in\mathcal{G}$ then $\boldsymbol{v}_t=\boldsymbol{w}_t=\boldsymbol{g}_{2t-1}+\boldsymbol{g}_{2t}$, and each recovery set for $\boldsymbol{v}_t$ is of the form $R_t=\{2t-1,2t\}$. If $t\in\mathcal{B}$ then $\boldsymbol{v}_t+\boldsymbol{e}=\boldsymbol{w}_t=\boldsymbol{g}_{2t-1}+\boldsymbol{g}_{2t}$, and if $t'\in\mathcal{R}$ then $\boldsymbol{e}=\boldsymbol{w}_{t'}=\boldsymbol{g}_{2t'-1}+\boldsymbol{g}_{2t'}$. Therefore, for all $t\in\mathcal{B}$ and $t'\in\mathcal{R}$, $\boldsymbol{v}_t=\boldsymbol{g}_{2t-1}+\boldsymbol{g}_{2t}+\boldsymbol{g}_{2t'-1}+\boldsymbol{g}_{2t'}$. By Claim 1, since $|\mathcal{B}|\leqslant\lfloor\frac{1}{3}\cdot 2^{s-1}\rfloor$, it holds that $|\mathcal{B}|\leqslant|\mathcal{R}|$. Thus, for all $t\in\mathcal{B}$, each recovery set R_t for $\boldsymbol{v}_t$ will have a different $t'\in\mathcal{R}$ such that $R_t=\{2t-1,2t,2t'-1,2t'\}$. ∎

In Lemma 8, it was shown that obtaining $\mathcal{M}(\mathcal{G},\mathcal{B},\mathcal{R})$ which holds $|\mathcal{B}|\leqslant\lfloor\frac{1}{3}\cdot 2^{s-1}\rfloor$ provides a solution for M. Therefore, if the first algorithm outputs a set $\mathcal{B}$ for which $|\mathcal{B}|\leqslant\lfloor\frac{1}{3}\cdot 2^{s-1}\rfloor$, then the solution for M is easily derived. Otherwise, the first algorithm outputs a set $\mathcal{B}$ such that $|\mathcal{B}|>\lfloor\frac{1}{3}\cdot 2^{s-1}\rfloor$. In this case, the second algorithm will be used in order to reduce the size of the set $\mathcal{B}$ to be at most $\lfloor\frac{1}{3}\cdot 2^{s-1}\rfloor$. For that, more definitions are required, and will be presented in the next section.

A. Graph Definitions

In the two algorithms of the construction, we will use undirected graphs, simple paths, and simple cycles that will be defined next. These graphs will be useful to represent the HG-matrix G in some graph representation and to make some swap operations on its columns.

Definition 9. *An **undirected graph** or simply a **graph** will be denoted by $\mathbf{G}=(V,E)$, where $V=\{u_0,u_1,\ldots,u_{m-1}\}$ is its set of m nodes (vertices) and $E\subseteq\{\{u_i,u_j\}\mid u_i,u_j\in V\}$ is its edge set. A finite **simple path** of length ℓ is a sequence of distinct edges $e_0,e_1,\ldots,e_{\ell-1}$ for which there is a sequence of vertices $u_{i_0},u_{i_1},\ldots,u_{i_\ell}$ such that $e_j=\{u_{i_j},u_{i_{j+1}}\}$, $j\in[\ell]$. A **simple cycle** is a simple path in which $u_{i_0}=u_{i_\ell}$. The **degree** of a node u_i is the number of edges that are incident to the node, and will be denoted by $\deg(u_i)$.*

By a slight abuse of notation, we will use graphs in which at most 2 parallel edges are allowed between any two nodes as it will be done in the following definition.

Definition 10. *Given an HG-matrix $G=[\boldsymbol{g}_0,\boldsymbol{g}_1,\ldots,\boldsymbol{g}_{n-1}]$, and a vector $\boldsymbol{x}\in\mathbb{F}_2^s$, denote the **$\boldsymbol{x}$-type graph** $\mathbf{G}_{\boldsymbol{x}}(G)=(V,E_{\boldsymbol{x}}(G))$ of G and $\boldsymbol{x}$ such that $V=\mathbb{F}_2^s$ and a multi-set*

$$E_{\boldsymbol{x}}(G)=\Big\{\{\boldsymbol{g}_i,\boldsymbol{g}_i+\boldsymbol{x}\}|i\in[n]\Big\}\cup\Big\{\{\boldsymbol{g}_{2t-1},\boldsymbol{g}_{2t}\}|t\in[n/2]\Big\}.$$

*For all $t\in[n/2]$, we say that $\boldsymbol{g}_{2t-1}$ and $\boldsymbol{g}_{2t}$ are a **pair**. An edge $\{\boldsymbol{g}_{2t-1},\boldsymbol{g}_{2t}\}$ will be called a **pair-type edge** and will be denoted by $\{\boldsymbol{g}_{2t-1},\boldsymbol{g}_{2t}\}_{\mathbf{p}}$. An edge $\{\boldsymbol{g}_i,\boldsymbol{g}_i+\boldsymbol{x}\}$ will be called an **$\boldsymbol{x}$-type edge** and will be denoted by $\{\boldsymbol{g}_i,\boldsymbol{g}_i+\boldsymbol{x}\}_{\boldsymbol{x}}$. Note that for any $\boldsymbol{g}\in V$, it holds that $\deg(\boldsymbol{g})=2$. Thus, the graph $\mathbf{G}_{\boldsymbol{x}}(G)$ has a partition of $\ell\geqslant 1$ disjoint simple cycles, that will be denoted by $\mathbf{C}_{\boldsymbol{x}}(G)=\{C_i\}_{i=0}^{\ell-1}$.*

Note that $E_{\boldsymbol{x}}(G)$ is a multi-set since in case that $\boldsymbol{g}_{2t-1}=\boldsymbol{g}_{2t}+\boldsymbol{x}$, we have two parallel edges $\{\boldsymbol{g}_{2t-1},\boldsymbol{g}_{2t}\}_{\mathbf{p}}$ and $\{\boldsymbol{g}_{2t-1},\boldsymbol{g}_{2t}\}_{\boldsymbol{x}}$ between $\boldsymbol{g}_{2t-1}$ and $\boldsymbol{g}_{2t}$. For the following definitions assume that $G=[\boldsymbol{g}_0,\boldsymbol{g}_1,\ldots,\boldsymbol{g}_{n-1}]$ is an HG-matrix of order $s\times n$.

Definition 11. *Given an $\boldsymbol{x}$-type graph $\mathbf{G}_{\boldsymbol{x}}(G)$ such that $\boldsymbol{x}\in\mathbb{F}_2^s$, let $\boldsymbol{g}_i,\boldsymbol{g}_j$ be two vertices connected by a simple path $P_{\boldsymbol{x}}(\boldsymbol{g}_i,\boldsymbol{g}_j,G)$ of length $\ell-1$ in $\mathbf{G}_{\boldsymbol{x}}(G)$ which is denoted by*

$$\boldsymbol{g}_i=\boldsymbol{g}_{s_0}-\boldsymbol{g}_{s_1}-\cdots-\boldsymbol{g}_{s_{\ell-1}}=\boldsymbol{g}_j.$$

*The path $P_{\boldsymbol{x}}(\boldsymbol{g}_i,\boldsymbol{g}_j,G)$ will be called a **good-path** if the edges $\{\boldsymbol{g}_{s_0},\boldsymbol{g}_{s_1}\}$ and $\{\boldsymbol{g}_{s_{\ell-2}},\boldsymbol{g}_{s_{\ell-1}}\}$ are both $\boldsymbol{x}$-type edges. For all $\boldsymbol{g}_t$ and $\boldsymbol{g}_m$ on $P_{\boldsymbol{x}}(\boldsymbol{g}_i,\boldsymbol{g}_j,G)$, denote by $d_{P_{\boldsymbol{x}}}(\boldsymbol{g}_t,\boldsymbol{g}_m,G)$ the length of the simple sub-path from $\boldsymbol{g}_t$ to $\boldsymbol{g}_m$ on $P_{\boldsymbol{x}}(\boldsymbol{g}_i,\boldsymbol{g}_j,G)$. This length will be called the **sub-length from $\boldsymbol{g}_t$ to $\boldsymbol{g}_m$ in** $P_{\boldsymbol{x}}(\boldsymbol{g}_i,\boldsymbol{g}_j,G)$. When the graph G will be clear from the context we will use the notation $P_{\boldsymbol{x}}(\boldsymbol{g}_i,\boldsymbol{g}_j)$, $d_{P_{\boldsymbol{x}}}(\boldsymbol{g}_t,\boldsymbol{g}_m)$ instead of $P_{\boldsymbol{x}}(\boldsymbol{g}_i,\boldsymbol{g}_j,G)$, $d_{P_{\boldsymbol{x}}}(\boldsymbol{g}_t,\boldsymbol{g}_m,G)$, respectively.*

The next definition will be used for changing the order of the columns in G.

Definition 12. *Let $\mathcal{H}_s$ be the set of all HG-matrices of order $s\times n$. Let $\mathcal{P}_s\subseteq\mathbb{F}_2^s\times\mathbb{F}_2^s$ be the set of all couples of column vectors $\boldsymbol{g}_m,\boldsymbol{g}_p$ of G such that there is a good-path $P_{\boldsymbol{x}}(\boldsymbol{g}_m,\boldsymbol{g}_p)$. For every two column vectors $\boldsymbol{g}_i,\boldsymbol{g}_j$ with a good-path $P_{\boldsymbol{x}}(\boldsymbol{g}_i,\boldsymbol{g}_j)$ of length $\ell-1$ in $\mathbf{G}_{\boldsymbol{x}}(G)$*

$$\boldsymbol{g}_i=\boldsymbol{g}_{s_0}-\boldsymbol{g}_{s_1}-\cdots-\boldsymbol{g}_{s_{\ell-1}}=\boldsymbol{g}_j,$$

denote the ***reordering function*** $\mathcal{F}_{\boldsymbol{x}} : \mathcal{P}_s \times \mathcal{H}_s \to \mathcal{H}_s$ *that generates an HG-matrix* $\mathcal{F}_{\boldsymbol{x}}(\boldsymbol{g}_i, \boldsymbol{g}_{j'}, G)$ *from* G *by adding* $\boldsymbol{x}$ *to every column* $\boldsymbol{g}_{s_m}, m \in [\ell]$. *We will use the notation* $\mathcal{F}_{\boldsymbol{x}}(\boldsymbol{g}_i, \boldsymbol{g}_j)$ *for shorthand.*

It can be easily verified that the matrix $\mathcal{F}_{\boldsymbol{x}}(\boldsymbol{g}_i, \boldsymbol{g}_j)$ is an HG-matrix of order $s \times n$. Another useful property on good-paths in $\boldsymbol{x}$-type graphs is stated in the next claim.

Claim 2 *If* $\boldsymbol{g}_i, \boldsymbol{g}_j$ *is a pair, then there is a good-path* $P_{\boldsymbol{x}}(\boldsymbol{g}_i, \boldsymbol{g}_j)$ *in* $\mathbf{G}_{\boldsymbol{x}}(G)$.

The next lemma shows a very important property that will be used in the construction of our first algorithm. This algorithm will have a routine of $\lfloor \frac{2}{3} \cdot 2^{s-1} \rfloor$ iterations. In iteration $t \leqslant \lfloor \frac{2}{3} \cdot 2^{s-1} \rfloor$, we will modify the order of the column vectors of G such that only the sums $\boldsymbol{g}_{2t-1} + \boldsymbol{g}_{2t}$ and $\boldsymbol{g}_{n-3} + \boldsymbol{g}_{n-2}$ will be changed by $\boldsymbol{x} \in \mathbb{F}_2^s$, and all other sums $\boldsymbol{g}_{2p-1} + \boldsymbol{g}_{2p}$ where $p \neq t, n/2-1$ will remain the same. The goal on the t-th iteration is to get that $\boldsymbol{g}_{2t-1} + \boldsymbol{g}_{2t} = \boldsymbol{v}_t + \mathbb{1}_t \boldsymbol{e}$, where $\mathbb{1}_t \in \{0,1\}$ and $\boldsymbol{e} = (0,0,\ldots,0,1) \in \mathbb{F}_2^s$.

Lemma 13. *Let* $P_{\boldsymbol{x}}(\boldsymbol{g}_{r_1}, \boldsymbol{g}_{r_2})$ *be a good-path in* $\mathbf{G}_{\boldsymbol{x}}(G)$ *where* $\boldsymbol{x} \in \mathbb{F}_2^s$ *and* $r_1, r_2 \in [n]$ *such that* $\boldsymbol{g}_{r_1}, \boldsymbol{g}_{r_2}$ *is not a pair. If* $r_1 \in \{2i-1, 2i\}$, $r_2 \in \{2j-1, 2j\}$ *(and note that* $i \neq j$*), then, the HG-matrix* $G' = \mathcal{F}_{\boldsymbol{x}}(\boldsymbol{g}_{r_1}, \boldsymbol{g}_{r_2}) = [\boldsymbol{g}'_0, \boldsymbol{g}'_1, \ldots, \boldsymbol{g}'_{n-1}]$ *satisfies the following equalities for* $p \in [n/2]$,

$$\begin{aligned} \boldsymbol{g}'_{2p-1} + \boldsymbol{g}'_{2p} &= \boldsymbol{g}_{2p-1} + \boldsymbol{g}_{2p} + \boldsymbol{x} \qquad && p \in \{i, j\}, \\ \boldsymbol{g}'_{2p-1} + \boldsymbol{g}'_{2p} &= \boldsymbol{g}_{2p-1} + \boldsymbol{g}_{2p} && p \neq i, j. \end{aligned}$$

B. The First Algorithm

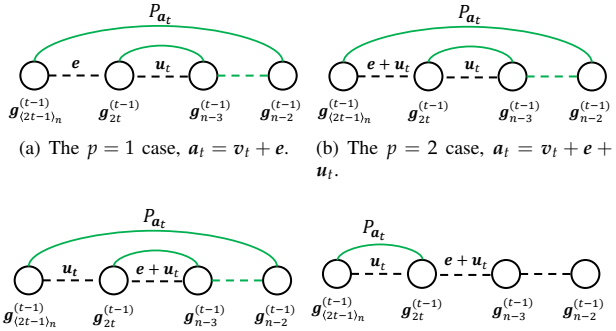
We start with the first algorithm which is referred by *FBSolution*(τ,M), where M is the request and τ will be the number of iterations in the algorithm, which is the number of columns in M. We define more variables that will be used in the routine of *FBSolution*(τ,M), and some auxiliary results. The τ iterations in the algorithm operate as follows. First, we define the initial state of the matrix $G = G^{(0)} = [\boldsymbol{g}_0^{(0)}, \boldsymbol{g}_1^{(0)}, \ldots, \boldsymbol{g}_{n-1}^{(0)}]$ such that $\boldsymbol{g}_0^{(0)} = \boldsymbol{v}_0$, and $\boldsymbol{g}_{2t-1}^{(0)} + \boldsymbol{g}_{2t}^{(0)} = \boldsymbol{e}$ for all $t \in [n/2]$. The proof of the existence of matrix $G^{(0)}$ can be found in [16].

Note that while we present this algorithm more generally, we will use it in this section for $\tau = \lfloor \frac{2}{3} \cdot 2^{s-1} \rfloor - 1$. The HG-matrix at the end of the t-th iteration will be denoted by $G^{(t)} = [\boldsymbol{g}_0^{(t)}, \boldsymbol{g}_1^{(t)}, \ldots, \boldsymbol{g}_{n-1}^{(t)}]$. We remind the reader that this algorithm will output a set $\mathcal{B}$ and an HG-matrix $G^{(\tau)}$, that will be a Hadamard solution for some matrix $\mathcal{M}(\mathcal{G}, \mathcal{B}, \mathcal{R}) = [\boldsymbol{w}_0, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_{2^{s-1}-1}]$, which is a triple-matrix of $M = [\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_{\lfloor \frac{2}{3} \cdot 2^{s-1} \rfloor - 1}]$ (see Definition 7). Also remember that $\mathcal{B}$ represents a set of "bad" indices $t \leqslant \tau$ in which $\boldsymbol{w}_t = \boldsymbol{v}_t + \boldsymbol{e}$. Now we are ready to present the *FBSolution*(τ,M) algorithm.

As we mentioned above, the *FBSolution*(τ,M) algorithm has an external routine of τ iterations. It also has an internal routine of 3 iterations. In each internal iteration, the vector $\boldsymbol{a}_t$ is first updated. Intuitively, $\boldsymbol{a}_t$ is a vector that has to be added to the sum of the pair $\boldsymbol{g}_{2t-1}^{(t-1)}, \boldsymbol{g}_{2t}^{(t-1)}$ in order to get $\boldsymbol{v}_t$ on the t-th iteration. In each external iteration, only the sums of the pairs $\boldsymbol{g}_{2t-1}^{(t-1)}, \boldsymbol{g}_{2t}^{(t-1)}$ and $\boldsymbol{g}_{n-3}^{(t-1)}, \boldsymbol{g}_{n-2}^{(t-1)}$ are changed. In other words, we do not update the sum of the pairs on indices $2p-1$ and $2p$ for all $p \neq t$ on the t-th iteration, even though the columns could be changed. The last pair on indices $n-3$ and $n-2$ is used as a "redundancy pair", i.e., it is not important what the sum of this pair is at any step of the algorithm.

Algorithm 1 *FBSolution*(τ, M)

1: Initialize $G^{(0)}$ and $\mathcal{B}^{(0)} = \emptyset$
2: **for** $t = 1, \ldots, \tau$ **do**
3: **if** $\boldsymbol{v}_t = \boldsymbol{e}$ **then**
4: Go to Step 2
5: $\boldsymbol{u}_t \leftarrow \boldsymbol{g}_{2t}^{(t-1)} + \boldsymbol{g}_{n-3}^{(t-1)}$
6: **for** $p = 1, 2, 3$ **do**
7: **if** $p = 1$ **then**
8: $\boldsymbol{a}_t \leftarrow \boldsymbol{v}_t + \boldsymbol{e}$
9: **if** $p = 2$ **then**
10: $\boldsymbol{a}_t \leftarrow \boldsymbol{v}_t + \boldsymbol{e} + \boldsymbol{u}_t$
11: Swap the columns $\boldsymbol{g}_{2t}^{(t-1)}$ and $\boldsymbol{g}_{n-3}^{(t-1)}$ in $G^{(t-1)}$
12: **if** $p = 3$ **then**
13: $\boldsymbol{a}_t \leftarrow \boldsymbol{v}_t + \boldsymbol{u}_t$
14: Swap the columns $\boldsymbol{g}_{2t-1}^{(t-1)}$ and $\boldsymbol{g}_{n-3}^{(t-1)}$ in $G^{(t-1)}$
15: $P_{\boldsymbol{a}_t} \leftarrow$ the good-path $P_{\boldsymbol{a}_t}(\boldsymbol{g}_{2t-1}^{(t-1)}, \boldsymbol{g}_{2t}^{(t-1)}, G^{(t-1)})$
16: $\mathbf{r} \leftarrow \{\boldsymbol{g}_{n-3}^{(t-1)}, \boldsymbol{g}_{n-2}^{(t-1)}\}_{\mathbf{p}}$
17: **if** $\mathbf{r} \in P_{\boldsymbol{a}_t}$ **then**
18: $d_1 \leftarrow d_{P_{\boldsymbol{a}_t}}(\boldsymbol{g}_{2t}^{(t-1)}, \boldsymbol{g}_{n-3}^{(t-1)})$
19: $d_2 \leftarrow d_{P_{\boldsymbol{a}_t}}(\boldsymbol{g}_{2t}^{(t-1)}, \boldsymbol{g}_{n-2}^{(t-1)})$
20: **if** $d_1 < d_2$ **then**
21: $j \leftarrow n-3$
22: **else**
23: $j \leftarrow n-2$
24: $G^{(t)} \leftarrow \mathcal{F}_{\boldsymbol{a}_t}(\boldsymbol{g}_{2t}^{(t-1)}, \boldsymbol{g}_j)$
25: Go to Step 2
26: $G^{(t)} \leftarrow \mathcal{F}_{\boldsymbol{a}_t}(\boldsymbol{g}_{2t-1}^{(t-1)}, \boldsymbol{g}_{2t}^{(t-1)})$
27: Swap the columns $\boldsymbol{g}_{2t-1}^{(t)}$ and $\boldsymbol{g}_{n-3}^{(t)}$ of $G^{(t)}$
28: $\mathcal{B}^{(t)} \leftarrow \mathcal{B}^{(t-1)} \cup \{t\}$
29: Return $G^{(\tau)}$ and $\mathcal{B}^{(\tau)}$



(a) The $p = 1$ case, $\boldsymbol{a}_t = \boldsymbol{v}_t + \boldsymbol{e}$. (b) The $p = 2$ case, $\boldsymbol{a}_t = \boldsymbol{v}_t + \boldsymbol{e} + \boldsymbol{u}_t$.

(c) The $p = 3$ case, $\boldsymbol{a}_t = \boldsymbol{v}_t + \boldsymbol{u}_t$. (d) The bad case, $\boldsymbol{a}_t = \boldsymbol{v}_t + \boldsymbol{u}_t$.

Fig. 1. In this figure we illustrate three good situations in which Step 17 in the algorithm *FBSolution*(τ,M) succeeds, and one bad case in which Step 17 in the algorithm *FBSolution*(τ,M) fails. The solid green line in all figures is a sub-path of the good-path $P_{\boldsymbol{a}_t}$ (which is a path between the nodes $\boldsymbol{g}_{2t-1}^{(t-1)}, \boldsymbol{g}_{2t}^{(t-1)}$ in $\mathbf{G}_{\boldsymbol{a}_t}(G^{(t-1)})$). The dashed lines represent the edges between the signed nodes. The green dashed line is an edge on $P_{\boldsymbol{a}_t}$. Without loss of generality, it is assumed that the closest node between $\boldsymbol{g}_{n-3}^{(t-1)}$ and $\boldsymbol{g}_{n-2}^{(t-1)}$ to $\boldsymbol{g}_{2t}^{(t-1)}$ in $P_{\boldsymbol{a}_t}$ is $\boldsymbol{g}_{n-3}^{(t-1)}$. The labels of the edges represent the summation of the vectors of its incident nodes. Each of the three good cases illustrated in (a)-(c) lead to the fact that a pair $\boldsymbol{g}_{2t-1}^{(t)}, \boldsymbol{g}_{2t}^{(t)}$ will be summed up to $\boldsymbol{v}_t$. This is done by invoking Step 24. In the bad case illustrated by (d), this pair will be summed up only to $\boldsymbol{v}_t + \boldsymbol{e}$. This is done by Steps 26-27.

Denote by $\mathbb{1}_t \in \{0,1\}$ a binary indicator for the t-th iteration such that $\mathbb{1}_t = 1$ if and only if the algorithm reaches

Step 26 in the t-th iteration. Our next goal is to prove the following important lemma with respect to the t-th iteration. Due to the lack of space we show only the proof sketch, while the complete proof can be found in [16].

Lemma 14. *In the t-th iteration, the algorithm will generate a matrix $G^{(t)} = [g_0^{(t)}, g_1^{(t)}, \ldots, g_{n-1}^{(t)}]$ such that*

$$g_{2p-1}^{(t)} + g_{2p}^{(t)} = \begin{cases} g_{2p-1}^{(t-1)} + g_{2p}^{(t-1)} & p \neq t, n/2-1 \\ v_t + \mathbb{1}_t e & p = t \end{cases}.$$

Proof sketch: First we show that in every iteration $t \leqslant \tau$, if the algorithm reaches Step 25, then

$$g_{2t-1}^{(t-1)} + g_{2t}^{(t-1)} + a_t = v_t. \tag{1}$$

We separate the proof for the three cases of $p \in \{1,2,3\}$. To better understand these cases we refer the reader to Fig. 1(a)-(c). Remember that by Step 5, $u_t = g_{2t}^{(t-1)} + g_{n-3}^{(t-1)}$.

1) If $p = 1$, then $g_{2t-1}^{(t-1)} + g_{2t}^{(t-1)} = e$. By Step 8, $a_t = v_t + e$, and therefore equality (1) holds.
2) If $p = 2$, then by Step 11, after swapping $g_{2t}^{(t-1)}$ and $g_{n-3}^{(t-1)}$, it is deduced that $g_{2t-1}^{(t-1)} + g_{2t}^{(t-1)} = e + u_t$. By Step 10, $a_t = v_t + e + u_t$, which concludes the correctness of equality (1).
3) If $p = 3$, then by Step 14, after swapping $g_{2t-1}^{(t-1)}$ and $g_{n-3}^{(t-1)}$, it is deduced that $g_{2t-1}^{(t-1)} + g_{2t}^{(t-1)} = u_t$. By Step 13, $a_t = v_t + u_t$, corresponding to (1).

Note that by Claim 2, there is always a good-path between $g_{2t-1}^{(t-1)}$ and $g_{2t}^{(t-1)}$. Now, suppose that in one of these 3 cases, there is a good-path $P_{a_t}(g_{2t-1}^{(t-1)}, g_{2t}^{(t-1)})$ in $\mathbf{G}_{a_t}(G^{(t-1)})$ which includes the edge $\{g_{n-3}^{(t-1)}, g_{n-2}^{(t-1)}\}_{\mathbf{p}}$ (with respect to Step 17). In Steps 18–23 we find the closest node between $g_{n-3}^{(t-1)}$ and $g_{n-2}^{(t-1)}$ to $g_{2t}^{(t-1)}$ on the path P_{a_t}. This node is denoted by g_j. Thus, the first and the last edges on the path $P_{a_t}(g_{2t}^{(t-1)}, g_j)$ have to be x-type edges. By definition, it is deduced that $P_{a_t}(g_{2t}^{(t-1)}, g_j)$ is a good-path. According to Step 24, $G^{(t)} = \mathcal{F}_{a_t}(g_{2t}^{(t-1)}, g_j)$. By Lemma 13 this step changes the pair summations of only the pairs $g_{2t-1}^{(t-1)}, g_{2t}^{(t-1)}$ and $g_{n-3}^{(t-1)}, g_{n-2}^{(t-1)}$. More precisely, $g_{2t-1}^{(t)} + g_{2t}^{(t)} = g_{2t-1}^{(t-1)} + g_{2t}^{(t-1)} + a_t = v_t$ and the sum of the pair $g_{n-3}^{(t)}, g_{n-2}^{(t)}$ does not matter.

Finally, if the algorithm does not succeed to find any of these good-paths, due to Steps 26–27, a matrix $G^{(t)}$ such that the pair $g_{2t-1}^{(t)}, g_{2t}^{(t)}$ will be almost correct, that is, $g_{2t-1}^{(t)} + g_{2t}^{(t)} = v_t + e$ can be obtained; see [16]. ■

At the end of the *FBSolution*(τ, M) algorithm we obtained the set $\mathcal{B}_1 = \mathcal{B}^{(\tau)}$ and the matrix $G^{(\tau)} = [g_0^{(\tau)}, g_1^{(\tau)}, \ldots, g_{n-1}^{(\tau)}]$. By Definition 7, the set $\mathcal{B}_1$ uniquely defines the triple-set $(\mathcal{G}_1, \mathcal{B}_1, \mathcal{R}_1)$. Thus, by Lemma 14, for all $t \in [n/2]$, the matrix $G^{(\tau)}$ satisfies that

$$g_{2t-1}^{(\tau)} + g_{2t}^{(\tau)} = \begin{cases} v_t & t \in \mathcal{G}_1 \\ v_t + e & t \in \mathcal{B}_1 \\ e & t \in \mathcal{R}_1 \end{cases}.$$

Let $\mathcal{M}_1 = [w_0, w_1, \ldots, w_{2^{s-1}-1}]$ be the matrix such that for all $t \in [n/2]$, $w_t = g_{2t-1}^{(\tau)} + g_{2t}^{(\tau)}$. Therefore, it is deduced that $\mathcal{M}_1 = \mathcal{M}_1(\mathcal{G}_1, \mathcal{B}_1, \mathcal{R}_1)$ is a triple-matrix of M. By definition of $\mathcal{M}_1(\mathcal{G}_1, \mathcal{B}_1, \mathcal{R}_1)$, the matrix $G^{(\tau)}$ is its Hadamard solution. If the set $\mathcal{B}_1$ satisfies $|\mathcal{B}_1| \leqslant \lfloor \frac{1}{3} \cdot 2^{s-1} \rfloor$ then by Lemma 8 there is a solution for M. Otherwise, we will make another reordering on the columns of $G^{(\tau)}$ in order to obtain a new bad set $\mathcal{B}_2$ for which $|\mathcal{B}_2| \leqslant \lfloor \frac{1}{3} \cdot 2^{s-1} \rfloor$. This will be done in the next section by showing our second algorithm.

C. The Second Algorithm

From now on we assume that $G^{(\tau)} = G = [g_0, g_1, \ldots, g_{n-1}]$ and $\mathcal{B} = \mathcal{B}_1$. Before showing the second algorithm we start with the following definition.

Definition 15. *Let $\mathbf{C}_e(G)$ be a partition of simple cycles in $\mathbf{G}_e(G)$. A pair of distinct indices $t_1, t_2 \in \mathcal{B}$ is called a **bad-indices pair** in $\mathbf{C}_e(G)$ if both edges $\{g_{2t_1-1}, g_{2t_1}\}_{\mathbf{p}}$ and $\{g_{2t_2-1}, g_{2t_2}\}_{\mathbf{p}}$ are in the same simple cycle in $\mathbf{G}_e(G)$.*

Now we show the algorithm *ClearBadCycles*$(G, \mathcal{B})$ in which the columns of the HG-matrix G are reordered, and the set $\mathcal{B}$ will be modified and its size will be decreased.

Algorithm 2 *ClearBadCycles*$(G, \mathcal{B})$

```
C_e(G) ← The partition of simple cycles in G_e(G)
while ∃ t_1, t_2 a bad-indices pair in C_e(G) do
    P_e ← the good-path P_e(g_{2t_1-1}, g_{2t_1})
    d_1 ← d_{P_e}(g_{2t_1}, g_{2t_2-1})
    d_2 ← d_{P_e}(g_{2t_1}, g_{2t_2})
    if d_1 < d_2 then
        j ← 2t_2 − 1
    else
        j ← 2t_2
    G ← F_e(g_{2t_1}, g_j)
    C_e(G) ← The partition of simple cycles in G_e(G)
    Remove t_1, t_2 from B
Return G and B
```

Let $G_2 = [g_0^\star, g_1^\star, \ldots, g_{n-1}^\star]$ be the HG-matrix and $\mathcal{B}_2$ be the bad set output of the *ClearBadCycles*$(G, \mathcal{B})$ algorithm. We remind the reader that $M = [v_0, v_1, \ldots, v_{\lfloor \frac{2}{3} \cdot 2^{s-1} \rfloor - 1}]$. Let $\mathcal{M}_2 = [w_0, w_1, \ldots, w_{2^{s-1}-1}]$ be the matrix such that for all $t \in [n/2]$, $w_t = g_{2t-1}^\star + g_{2t}^\star$. Since $\mathcal{B}_2$ uniquely defines the triple set $(\mathcal{G}_2, \mathcal{B}_2, \mathcal{R}_2)$, it is deduced that the matrix $\mathcal{M}_2$ is a triple-matrix $\mathcal{M}_2(\mathcal{G}_2, \mathcal{B}_2, \mathcal{R}_2)$ of M. Next, it will be shown that the algorithm *ClearBadCycles*$(G, \mathcal{B})$ will stop and $|\mathcal{B}_2|$ will be bounded from above by $\lfloor \frac{1}{3} \cdot 2^{s-1} \rfloor$ after the execution of the algorithm.

Lemma 16. *The algorithm ClearBadCycles$(G, \mathcal{B})$ outputs a set $\mathcal{B}_2$ such that $|\mathcal{B}_2| \leqslant \lfloor \frac{1}{3} \cdot 2^{s-1} \rfloor$.*

We are finally ready to prove the main result of this section.

Theorem 17. *An FB-$(2^s - 1, s, \lfloor \frac{2}{3} \cdot 2^{s-1} \rfloor)$ code exists.*

Proof: Using the result of Lemma 16 it is deduced that the algorithm *ClearBadCycles*$(G, \mathcal{B})$ outputs the set $\mathcal{B}_2$ such that it size is at most $\lfloor \frac{1}{3} \cdot 2^{s-1} \rfloor$. The HG-matrix G_2 is again a Hadamard solution for a triple-matrix $\mathcal{M}_2(\mathcal{G}_2, \mathcal{B}_2, \mathcal{R}_2)$ of M. Thus, by using Lemma 8, it is deduced that there is a solution for M. After removing the all-zero column vector from G, the proof of this theorem is immediately deduced. ■

ACKNOWLEDGMENTS

This work was partially supported by the ISF grant 1817/18 and by the Technion Hiroshi Fujiwara cyber security research center and the Israel cyber directorate.

REFERENCES

[1] H. Asi and E. Yaakobi, "Nearly optimal constructions of PIR and batch codes," *In Proceedings IEEE International Symposium on Information Theory*, pp. 151–155, Aachen, Germany, Jun. 2017.

[2] S. Buzaglo, Y. Cassuto, P. H. Siegel, and E. Yaakobi, "Consecutive switch codes," *IEEE Transactions on Information Theory*, vol. 64, no. 4, pp. 2485–2498, Apr. 2018.

[3] S. Arora and B. Barak, "Computational complexity – a modern approach," *Cambridge University Press*, Cambridge, 2009

[4] G.D. Cohen, P. Godlewski, and F. Merx, "Linear binary code for write-once memories," *IEEE Transactions on Information Theory*, vol. 32, no. 5, pp. 697–700, Oct. 1986.

[5] A. Fazeli, A. Vardy, and E. Yaakobi, "PIR with low storage overhead: Coding instead of replication," *arxiv.org/abs/1505.06241*, May 2015.

[6] P. Godlewski, "WOM-codes construits à partir des codes de Hamming," *Discrete Mathematics*, vol. 65, no. 3, pp. 237–243, Jul. 1987.

[7] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, "Batch codes and their applications," *In Proceedings 36th Annual ACM Symposium on Theory of Computing*, pp. 262–271, 2004.

[8] A. S. Rawat, Z. Song, A. G. Dimakis, and A. Gál, "Batch codes through dense graphs without short cycles," *IEEE Transactions on Information Theory*, vol. 62, no. 4, Apr. 2016.

[9] R.L. Rivest and A. Shamir, "How to reuse a write-once memory," *Information and Control*, vol. 55, no. 1–3, pp. 1–19, Dec. 1982.

[10] E. Sharon and I. Alrod, "Coding scheme for optimizing random I/O performance," *IEEE Transactions on Information Theory*, San Diego, Apr. 2013.

[11] H. Sun and S. A. Jafar, "The capacity of private computation," *arxiv.org/abs/1710.11098*, Oct. 2017.

[12] M. Vajha, V. Ramkumar, and P. V. Kumar, " Binary, shortened projective Reed Muller codes for coded private information retrieval," *In Proceedings IEEE International Symposium on Information Theory*, pp. 2648–2652. Aachen, Germany, Jun. 2017.

[13] A. Vardy and E. Yaakobi, "Constructions of batch codes with near optimal redundancy," *In Proceedings IEEE International Symposium on Information Theory*, pp. 1197–1201, 2016.

[14] E. Yaakobi, S. Kayser, P. H. Siegel, A. Vardy, and J.K. Wolf, "Codes for write-once memories," *IEEE Transactions on Information Theory*, vol. 58, no. 9, pp. 5985–5999, Sep. 2012.

[15] A. Yamawaki, H. Kamabe, and S. Lu, "Construction of parallel RIO codes using coset coding with Hamming code," *In Proceedings IEEE Information Theory Workshop*, pp. 239–243, Kaohsiung, Taiwan, Nov. 2017

[16] L. Yohanaov and E. Yaakobi, "Almost optimal construction of functional batch codes using Hadamard codes," *arxiv.org/abs/2101.06722*, Jan. 2021.

[17] Y. Zhang, T. Etzion, and E. Yaakobi, "Bounds on the length of functional PIR and batch codes," *IEEE Transactions on Information Theory*, vol. 66, no. 8, pp. 4917–4934, Aug. 2020.