Constructions of Batch Codes with Near-Optimal Redundancy

Alexander Vardy[*†] and **Eitan Yaakobi**[‡]

[*]Department of Electrical and Computer Engineering, University of California San Diego, La Jolla, CA 92093, USA
[†]School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore 637371
[‡]Computer Science Department, Technion – Israel Institute of Technology, Haifa 32000, Israel
avardy@ucsd.edu, **yaakobi@cs.technion.ac.il**

***Abstract*—*Batch codes*, first studied by Ishai *et al.*, are a coding scheme to encode n information bits into m buckets, in a way that every batch request of k bits can be decoded while at most one bit is read from each bucket. In this work we study the class of *multiset primitive batch codes*, in which every bucket stores a single bit and bits can be requested multiple times. We simply refer to these codes as batch codes. The main problem under this paradigm is to optimize the number of encoded bits, which is the number of buckets, for given n and k, and we denote this value by $B(n,k)$. Since there are several asymptotically optimal constructions of these codes, we are motivated to evaluate their optimality by their *redundancy*. Thus we define the *optimal redundancy* of batch codes to be $r_B(n,k) \triangleq B(n,k) - n$. Our main result in this paper claims that for any fixed k, $r_B(n,k) = O(\sqrt{n}\log(n))$.**

I. Introduction

A *batch code* encodes n information bits into m buckets, such that every batch of k bits can be decoded by reading at most one (and more generally t) bits from each bucket. This class of codes was first introduced by Ishai *et al.* in the previous decade [8]. There are several families of batch codes which are classified according to several properties: whether the bits can be encoded, the number of bits each bucket stores, and lastly whether the batch of k bits may contain several requests of the same bit. In *combinatorial batch codes*, the bits stored in the buckets are simply copies of the information bits (i.e., not coded). Since the first work of [8], this class of batch codes was mostly studied; see e.g. [1]–[3], [17]. On the other hand, *computational batch codes*, which is the class of codes studied in this paper, allow to encode the bits stored in the buckets. There is only a limited number of constructions of computational batch codes and the current works which we are aware of are [8], [10], [16], [19]. Recently, bounds on these codes were presented in [23]. A batch code in which every bucket contains a single bit is called a *primitive batch code* and if the k bits may contain multiple requests of the same bit, then it is called a *multiset batch code*. In this work we refer to *batch codes* as computational primitive multiset batch codes. These codes will be denoted by $(m,n,k)_B$ batch codes, where n is the number of information bits, k is the batch size, and m is the number of buckets (or encoded bits since every bucket stores a single bit). The main goal of this paper is to study constructions of batch codes.

Batch codes were originally motivated by different applications such as load-balancing in storage and cryptographic protocols [8]. Hence they have received renewed interest recently due to their applicability to codes for distributed storage. *Locally repairable codes* (*LRCs*) are a class of codes in which a failure of a single node can be recovered by accessing at most some r other nodes [6], [13], [18]. In addition to symbol locality, another important property of codes is their symbol *availability*, which is defined to be the number of mutually disjoint recovering sets for every symbol [12], [15], [21]. High availability is a particularly attractive property for so-called *hot data* in a distributed storage system.

Another family of these codes imposes only the availability but *not* the locality constraint, and hence it is required that every symbol has some t recovering sets while their size is not constrained. It was recently observed in [7] that the this class of codes was already studied a while ago by Massey [11] and later by Lin and others [9] for applications of fast decoding, and were called *one-step majority-logic decodable codes*. Under this setup every symbol is required to have several mutually disjoint recovering sets, and it is decoded according to the majority of the values given by all of its recovering sets. Another application of codes with availability was also brought in [5] for *private information retrieval* (*PIR*) protocols [4], [22], where a similar family of codes was studied, called *k-server PIR codes*, in which n bits are encoded to m bits such that every information bit has k mutually disjoint recovering sets. In this work, we refer to this class of codes as *PIR codes*. Batch codes can be seen as a generalization of PIR codes. Instead of requiring that every bit has k mutually disjoint recovering sets, this property holds for every multiset of k bits. Thus, every batch code is in particular a PIR code with the same parameters. Lastly, we note that a special class of batch codes, called *switch codes*, in which $k = n$ was studied in [19], [20].

The main goal in constructing batch codes is to minimize the value of m, given n and k. For example, for $k=1$, the minimum value of m is n, and for $k=2$, its value is $n+1$, since the simple parity is a $(n+1,n,2)_B$ batch code. For given n and k, we denote by $B(n,k)$ to be the smallest value of m such that there exists an $(m,n,k)_B$ batch code. Hence, $B(n,1)=n$ and $B(n,2)=n+1$. In particular, we will be interested in studying the asymptotic behavior of $B(n,k)$ when k is either fixed or a function of n. Based on the Subcube code construction from [8], it is possible to derive that for any fixed k, $B(n,k)$ is at most $n + O\left(n^{1-\frac{1}{\lceil \log k \rceil}}\right)$, and thus this construction is asymptotically optimal, i.e. $\lim_{n\to\infty} B(n,k)/n = 1$. This motivates us to investigate the *redundancy* of batch codes and thus study the value $r_B(n,k) \triangleq B(n,k) - n$ it order to evaluate how fast the code rate approaches 1. Our main result in the paper claims that for any fixed k,

$$r_B(n,k) = O(\sqrt{n}\log(n)). \tag{1}$$

978-1-5090-1806-2/16/$31.00 ©2016 IEEE

We note that a lower bound on the redundancy of PIR codes, which is also a lower bound on the redundancy of batch codes, states that for any fixed k, $r_B(n,k) = \Omega(\sqrt{n})$ [14].

The rest of the paper is organized as follows. In Section II, we formally define the codes studied in this paper and present several useful properties. In Section III, we show that PIR codes for $k = 3,4$ are also batch codes with the same parameters, and give a construction which confirms that (1) holds for $k = 5,6,7$. Lastly, in Section IV, we show how to extend the last construction so that (1) holds for any fixed k. We discuss and present more results in case k is not fixed. Due to the lack of space some proofs in the paper are omitted.

II. Definitions and Preliminaries

In this section we formally define the codes we study in the paper. A linear code over $GF(q)$ of length n and dimension k will be denoted by $[n,k]_q$ or by $[n,k,d]_q$ where d specifies the minimum distance of the code. In case the code is binary we will omit the field notation. For a positive integer n the notation $[n]$ will refer to the set $\{1,\dots,n\}$.

Batch codes were first studied by Ishai *et al.* in [8]. The basic problem refers to the encoding of a length-n message $\boldsymbol{x}$ into an m-tuple of strings, called buckets, such that each batch of k bits from $\boldsymbol{x}$ can be decoded by reading at most some t symbols from each bucket. Formally, these codes are defined as follows [8].

Definition 1. *An (n,N,k,m,t) **batch code** over Σ is defined by an encoding map $\mathcal{E} : \Sigma^n \to (\Sigma^*)^m$ (each output is called a bucket) and a decoding map $\mathcal{D}$ such that:*

1) *The total length of all m buckets is N.*
2) *For any $\boldsymbol{x} \in \Sigma^n$ and $\{i_1,\dots,i_k\} \subseteq [n]$,*

$$\mathcal{D}(\mathcal{E}(\boldsymbol{x}), i_1,\dots,i_k) = (x_{i_1},\dots,x_{i_k}),$$

and $\mathcal{D}$ probes at most t symbols from each bucket in $\mathcal{E}(\boldsymbol{x})$ (whose positions are determined by $i_1,\dots,i_k$).

*An (n,N,k,m,t) **multiset batch code** is an (n,N,k,m,t) batch code which also satisfies the following property: For any multiset $i_1,\dots,i_k \in [n]$ there is a partition of the buckets into k subsets $S_1,\dots,S_k \subseteq [m]$ such that each symbol x_{i_j}, $j \in [k]$, can be recovered by reading at most t symbols from each bucket in S_j.*

*An (n,N,k,m) (multiset) batch code is an $(n,N,k,m,1)$ (multiset) batch code and a **primitive (multiset) batch code** is an (n,N,k,m) (multiset) batch code in which each bucket contains a single symbol, that is $N = m$.*

In another class of codes, called *combinatorial batch codes*, it is required to have the same properties of batch codes, however the symbols cannot be encoded. Several works have considered codes under this setup; see e.g. [1]–[3], [17]. All codes studied in this work are binary, that is $\Sigma = \{0,1\}$, however all the results in the paper can be extended to the non-binary case as well.

Example 1. It is clear to see that for all $k \leqslant n$ there exists a primitive (n,n,k,n) batch code and this is an optimal construction. The simple parity code $[n+1,n,2]$ is a primitive multiset batch code with parameters $(n,n+1,2,n+1)$. $\square$

There are several constructions of batch codes, studied first in [8] and recently in [10], [16], [19]. Since it is not easy to compare between the parameters of these code constructions, we first seek to provide a simple figure of merit which we believe to provide a fair comparison between all the constructions and will allow to evaluate the optimality of each construction. As a result of the multiple parameters in the definition of batch codes, we will fix some of them and optimize the remaining one. For simplicity we assume that the number of bits stored in each bucket is the same, and we denote this value by ℓ, so $N = \ell m$. Next, we fix the values of n,k,t and ℓ and will then seek to optimize the value of m (and thus also N). In particular, we will have that t and ℓ are fixed, where $t \leqslant \ell$, and then study the growth of m as a function of n and k. Hence, we denote by $D_{t,\ell}(n,k)$ the smallest m such that an (n,N,k,m,t) batch code exists, where every bucket stores exactly ℓ bits. Similarly we let $B_{t,\ell}(n,k)$ denote the smallest m such that an (n,N,k,m,t) multiset batch code exists.

In this paper we focus on studying the value of $B_{1,1}(n,k)$, which for simplicity we denote by $B(n,k)$. In particular, we are interested in studying the growth of $B(n,k)$ as a function of n where k is fixed or is a function of n. For the simplicity of notations and definitions, in the rest of the paper whenever we refer to a batch code, we refer to a primitive multiset batch code and it is denoted by an $(m,n,k)_B$ *batch code*.

In [5], a similar family of codes was studied, called *k-server PIR codes*, in which n bits where encoded into m bits in a way that every information bit has k mutually disjoint recovering sets. Let us formally define these codes.

Definition 2. *A binary $[m,n]$ linear code will be called a **k-server PIR code** if for every information bit x_i, $i \in [n]$, there exist k mutually disjoint sets $R_{i,1},\dots,R_{i,k} \subseteq [m]$ such that for all $j \in [k]$, x_i is a linear function of the bits in $R_{i,j}$.*

PIR codes, as defined in [5], are required to be linear in order to be used in PIR protocols. Even though for the purpose of this work we do not need the linearity property, all constructions we present in the paper will be linear and thus this requirement does not impose another constraint. To be consistent with the notations of the paper, we denote an $[m,n]$ k-server PIR code by an $(m,n,k)_P$ *PIR code*.

The main problem studied in [5] with respect to PIR codes was to minimize the value of m given n and k. Let us denote this value by $P(n,k)$, where as for batch codes it holds that $P(n,1) = n$ and $P(n,2) = n+1$. Note that every $(n,m,k)_B$ batch code is also an $(n,m,k)_P$ so for all n,k, $B(n,k) \geqslant P(n,k)$.

In [5], it was observed by the codes from [9] that for any fixed $k \geqslant 3$ there is a construction of $(m,n,k)_P$ PIR code, where $m = n + O(\sqrt{n})$ and therefore this construction is asymptotically optimal, that is for any fixed k, $\lim_{n\to\infty} P(n,k)/n = 1$. Similarly, using the Subcube code construction from [8], it is also possible to derive that if k is fixed then $\lim_{n\to\infty} B(n,k)/n = 1$. Therefore, in order to better understand the efficiency of each construction, we evaluate PIR and batch codes by their *redundancy*. Thus, we define $r_B(n,k)$ to be the value $r_B(n,k) \triangleq B(n,k) - n$ and similarly, $r_P(n,k) \triangleq P(n,k) - n$.

Hence, $r_B(n,1) = r_P(n,1) = 0$, $r_B(n,2) = r_P(n,2) = 1$, for any fixed $k \geqslant 3$, $r_P(n,k) = \Theta(\sqrt{n})$ [14], and thus $r_B(n,k) = \Omega(\sqrt{n})$. When possible in the paper we will explicitly calculate the values of $B(n,k)$ and $P(n,k)$, however we will be mostly interested in the order of the value of $r_B(n,k)$ for fixed k or the order $r_B(n,k)$ and $r_P(n,k)$ when k is a function of n.

According to [16], there are several results which can be deduced on the behavior of the $r_B(n,k)$ when k is a function of n. In particular, $r_B(n,n^{1/3}) \leqslant n$, $r_B(n,n^{1/t}) \leqslant n^{7/8}$ for $4 \leqslant t \leqslant 32/7$, and $r_B(n,n^{1/t}) \leqslant n^{4/t}$ for $32/7 < t \leqslant 5$. By the Subcube construction from [8], it also follows that for any fixed k, $r_B(n,k) = O(n^{1-1/\lceil \log k \rceil})$. The case where $n = k$ was studied in [19] and it was shown that $B(n,n) = O(n^2/\log(n))$. For PIR codes, there are different constructions in [5], which use tools from algebraic cyclic codes, different sets, Steiner systems and more. These tools provide several codes, among them are $(2^{2m}+2^m+1, 2^{2m}+2^m-3^m, 2^m+1)_P$ and $(2^{2m}-1, 2^{2m}-3^m, 2^m)_P$ PIR codes, and thus it is possible to deduce that $r_P(n,\sqrt{n}) = O(n^{\frac{\log 3}{2}})$.

A batch or PIR code will be called *systematic* if every information bit appears systematically in the encoded bits. That is, the n information bits, given by the vector $\boldsymbol{x} \in \{0,1\}^n$, are encoded to be $\mathcal{E}(\boldsymbol{x}) = (\boldsymbol{x},\boldsymbol{p})$, where the vector $\boldsymbol{p}$ corresponds to the parity bits. Unless stated otherwise we assume that all codes studied in the paper are systematic where the information bits appear first, followed by the parity bits,. Furthermore, when it will be clear from the context, the output of the encoding map will be only the parity part, i.e. $\mathcal{E}(\boldsymbol{x}) = \boldsymbol{p}$.

III. Constructions of Batch codes for $k \leqslant 7$

The main goal of this section is to give constructions of $(m,n,k)_B$ batch codes for $3 \leqslant k \leqslant 7$ and in particular analyze the value $r_B(n,k)$.

We first show a useful property of PIR codes which we will take advantage of in our constructions. Even though PIR codes are designed to answer only multiset requests which consist of a single bit, they can successfully answer other requests as long as at most a single bit is requested more than once.

Lemma 3. *An $(m,n,k)_P$ PIR code can successfully answer all multiset requests of k bits in which at most a single bit is requested more than once.*

Proof: Let $\mathcal{C}$ be an $(m,n,k)_P$ PIR code with encoding map $\mathcal{E}$ and decoding map $\mathcal{D}$. Assume the information vector $\boldsymbol{x}$ was encoded (systematically) by the encoding map $\mathcal{E}$ to the vector $(\boldsymbol{x},\boldsymbol{p})$, and let S be a multiset request of k bits. If every bit is requested exactly once then this request can be simply answered by the systematic part of $\boldsymbol{x}$. Hence, we assume that only the first bit is requested more than once, so the multiset S can be represented by $S = [i_0, i_0, \ldots, i_0, i_1, \ldots, i_h]$, where i_0 is requested $k-h$ times. Then, the decoding map $\mathcal{D}$ is invoked to get k mutually disjoint recovering sets for the i_0-th bit,

$$\mathcal{D}((\boldsymbol{x},\boldsymbol{p}), i_0) = (R_1, R_2, \ldots, R_k).$$

Since these k sets are mutually disjoint the h bits $i_1, \ldots, i_h$ can appear in at most h of them and assume without loss of generality that these are the last h sets. Therefore, the k mutually disjoint sets which recover the k bits in the multiset request S are given by the $k-h$ sets $R_1, \ldots, R_{k-h}$ to recover the i_0-th bit $k-h$ times and the sets $\{i_j\}$ for $j \in [h]$ to recover the other h bits. ■

According to the property proved in Lemma 3, we can deduce that for $k = 3$, a construction of $(m,n,3)_P$ PIR code is also a batch code with the same parameters. That it, we have the following lemma.

Lemma 4. *For all positive n, $B(n,3) = P(n,3)$.*

Proof: Assume $\mathcal{C}$ is an $(m,n,3)_P$ PIR code with encoding map $\mathcal{E}$ and decoding map $\mathcal{D}$. Note that for every multiset request of three bits at most a single bit is requested more than once and therefore according to Lemma 3 this multiset request can be answered by the decoding map $\mathcal{D}$. ■

Even though the same argument used in Lemma 4 does not hold for $k = 4$, it is possible to use the structure of PIR codes from [5] for $k = 4$, in which the union of the four recovering sets for each information bit is $[m]$. Therefore we get the following property.

Lemma 5. *For all positive n, $B(n,4) = P(n,4)$.*

Next we continue with the case $k = 5$. We cannot repeat the same arguments as for $k = 3,4$ since now we may have two bits with more than one query which cannot be resolved as for $k = 4$ case. The idea we carry here, and will adopt for other values of k, is to generate several partitions of the n information bits into two sets. For each partition, we encode each of the two sets of information bits separately and store the two parity vectors by these encoding operations. Then, in decoding, if we receive a multiset request where two bits are requested more than once, we find a partition of the n information bits where these two bits belong to different sets. Finally, we can decode each information bit separately using the information bits in its set according to the corresponding redundancy vector calculated in the encoding stage.

Before presenting this construction we define the following notations. For a length-n vector $\boldsymbol{c} \in \{0,1\}^n$, we denote by $I(\boldsymbol{c})$ to be the indicator set of the vector $\boldsymbol{c}$, that is $I(\boldsymbol{c}) = \{i : c_i = 1\}$. The vector $\overline{\boldsymbol{c}}$ is the bit-wise complement of the vector $\boldsymbol{c}$, and for a set $S \subseteq [n]$, the vector $\boldsymbol{c}_S$ is a length-$|S|$ vector whose indices are given by the set S.

Theorem 6. *For all positive n large, the following holds,*

$$r_B(n,5) \leqslant r_P(n,5) + 2\lceil \log(n) \rceil r_P(n/2,3).$$

Proof: For simplicity we assume in the proof that n is a power of two, while the modification for other cases will be clear from the construction. Let $\mathcal{C}_1$ be an $(m_1,n,5)_P$ PIR code with encoder $\mathcal{E}_1$ and decoder $\mathcal{D}_1$, and let $\mathcal{C}_2$ be an $(m_2,n/2,3)_P$ PIR code with encoder $\mathcal{E}_2$ and decoder $\mathcal{D}_2$. Let $r_1 = m_1 - n$ be the redundancy of $\mathcal{C}_1$ and $r_2 = m_2 - n/2$ the redundancy of $\mathcal{C}_2$. Let G be a $\log(n) \times n$ matrix formed by all $\log(n)$-length binary vectors as its columns and let $\mathcal{B} = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{\log(n)}\}$ be the $\log(n)$ rows of the matrix G. Note that G can be seen as the generator matrix of the $[n, \log(n), n/2]$ Hadamard code and has the property that for every two different column indices $i_1, i_2 \in [n]$ there exists a row $\boldsymbol{u}_\ell, \ell \in [\log(n)]$ such that $u_{i,i_1} \neq u_{i,i_2}$.

We construct an $(m,n,5)_B$ batch code of length $m = n + r_1 + 2(\log(n))r_2$ with encoder $\mathcal{E}$ and decoder $\mathcal{D}$ as follows. Let $\boldsymbol{x}$ be a length-n input binary vector, then it is encoded by the first encoder $\mathcal{E}_1$ to receive r_1 redundancy bits $\boldsymbol{p}_0 = \mathcal{E}_1(\boldsymbol{x})$. For each $i \in [\log(n)]$, the vectors $\boldsymbol{x}_{I(\boldsymbol{u}_i)}, \boldsymbol{x}_{I(\overline{\boldsymbol{u}_i})}$ are encoded by the encoder $\mathcal{E}_2$ to receive two parity vectors of r_2 bits each $\boldsymbol{p}_i, \boldsymbol{q}_i$, respectively,

$$\boldsymbol{p}_i = \mathcal{E}_2(\boldsymbol{x}_{I(\boldsymbol{u}_i)}), \quad \boldsymbol{q}_i = \mathcal{E}_2(\boldsymbol{x}_{I(\overline{\boldsymbol{u}_i})}).$$

Together, the vector $\boldsymbol{x}$ is encoded to the vector

$$(\boldsymbol{x}, \boldsymbol{p}_0, \boldsymbol{p}_1, \boldsymbol{q}_1, \boldsymbol{p}_2, \boldsymbol{q}_2, \ldots, \boldsymbol{p}_{\log(n)}, \boldsymbol{q}_{\log(n)}).$$

Let us now show that this code construction provides an $(m,n,5)_B$ batch code. The decoder $\mathcal{D}$ receives a multiset request S of some five bits $i_1, \ldots, i_5$ (which can be identical). The case where there are no two bits with more than one request is solved according to Lemma 3 using $(\boldsymbol{x}, \boldsymbol{p}_0)$. Otherwise, there are two cases: $\{i_1, i_1, i_1, i_2, i_2\}$ or $\{i_1, i_1, i_2, i_2, i_3\}$. In both cases we first find a vector $\boldsymbol{u}_\ell \in \mathcal{B}$ such that $u_{\ell,i_1} = 1$ and $u_{\ell,i_2} = 0$. Then, in the first case, we decode twice using the decoding map $\mathcal{D}_2$ as follows

$$\mathcal{D}_2\big((\boldsymbol{x}_{I(\boldsymbol{u}_\ell)}, \boldsymbol{p}_\ell), (i_1, i_1, i_1)\big), \quad \mathcal{D}_2\big((\boldsymbol{x}_{I(\overline{\boldsymbol{u}_\ell})}, \boldsymbol{q}_\ell), (i_2, i_2)\big),$$

to get five mutually disjoint recovering sets for the five bits in the multiset request S. Similarly, in the second case we apply the decoders $\mathcal{D}_2\big((\boldsymbol{x}_{I(\boldsymbol{u}_\ell)}, \boldsymbol{p}_\ell), (i_1, i_1, i_3)\big)$ and $\mathcal{D}_2\big((\boldsymbol{x}_{I(\overline{\boldsymbol{u}_\ell})}, \boldsymbol{q}_\ell), (i_2, i_2)\big)$. ■

We can repeat the same ideas and arguments as in the last proof and get the following result.

Theorem 7. *For all n, the following holds,*

1) $r_B(n,6) \leqslant r_P(n,6) + 2\lceil\log(n)\rceil r_P(n/2, 4)$.
2) $r_B(n,7) \leqslant r_P(n,7) + 2\lceil\log(n)\rceil r_P(n/2, 5)$.

The next corollary summarizes the results in the section.

Corollary 8.

1) *For $k \in \{3,4\}$, $r_B(n,k) = O(\sqrt{n})$.*
2) *For $k \in \{5,6,7\}$, $r_B(n,k) = O(\sqrt{n}\log n)$.*

IV. Constructions for large fixed k

Our goal in this section is to extend the results we derived in Section III for larger values of k. Using this extension, we will be able to show that for any fixed $k \geqslant 8$, $r_B(n,k) = O\left(\sqrt{n}\log(n)\right)$. Note that the main idea in using Hadamard codes in Theorem 6 was to have partitions of the n bits into $\log(n)$ partitions such that every two bits can be split by at least one partition. That is, we found $\log(n)$ sets $T_1, T_2, \ldots, T_{\log(n)}$ such that for every pair of different indices $i, j \in [n]$ there exists a set T_h for some $h \in [\log(n)]$ such that either i or j belongs to T_h. In order to build upon this approach we extend this property and construct codes that satisfy this extension.

A set of sets P is called a *q-partition of* $[n]$ if P consists of q mutually disjoint subsets of $[n]$ which their union is $[n]$. That is, $P = \{U_1, \ldots, U_q\}$, where $U_i \cap U_j = \emptyset$ for every two different indices $i, j \in [q]$, and $\cup_{i=1}^q U_i = [n]$. A q-partition $P = \{U_1, \ldots, U_q\}$ *covers* a set of q indices $X = \{x_1, \ldots, x_q\} \subseteq [n]$ if the q indices are all in different subsets, i.e. for all $i \in [q]$, $|U_i \cap X| = 1$. A set of q-partitions $\mathcal{P} = \{P_1, \ldots, P_s\}$ is called *complete* if it covers every set $X \subseteq [n]$ of q indices.

Example 2. The following set of 2-partitions

$$\mathcal{P}_4 = \Big\{P_1 = \{\{1,2\},\{3,4\}\}, P_2 = \{\{1,3\},\{2,4\}\}\Big\}$$

of [4] is complete, and the set $\mathcal{P}_8$ of 2-partitions of [8] is complete as well, where

$$\begin{aligned}\mathcal{P}_8 = \Big\{&P_1 = \{\{1,2,3,4\},\{5,6,7,8\}\},\\ &P_2 = \{\{1,2,5,6\},\{3,4,7,8\}\},\\ &P_3 = \{\{1,3,5,7\},\{2,4,6,8\}\}\Big\}.\end{aligned}$$

□

It is not hard to see that for every n and $q \leqslant n$ there exists a complete set of q-partitions of $[n]$. Hence, the goal is to find a complete set of minimum cardinality. For example, using Hadamard codes it is possible to construct for all n a set of $\lceil\log(n)\rceil$ 2-partitions of $[n]$ and this construction is optimal. We denote by $S(n,q)$ the minimum cardinality of a set of q-partitions of $[n]$, which is complete. Therefore, $S(n,2) = \lceil\log(n)\rceil$.

Using the construction of complete sets of q-partitions we derive the following result.

Theorem 9. *For all positive n and fixed k, the following holds*

$$r_B(n,k) \leqslant r_P(n,k) + \left\lfloor\frac{k}{2}\right\rfloor S(n, \lfloor k/2\rfloor) r_P\left(\left\lceil\frac{n}{\lfloor k/2\rfloor}\right\rceil, k-2\right).$$

Proof: For simplicity let us assume that k is even and n is a multiple of $k/2$. The modifications for other cases will be clear from the proof. Let $\mathcal{C}_1$ be an $(m_1, n, k)_P$ PIR code with redundancy $r_1 = m_1 - n$, encoder $\mathcal{E}_1$, and decoder $\mathcal{D}_1$, and let $\mathcal{C}_2$ be an $(m_2, 2n/k, k-2)_P$ PIR code with redundancy $r_2 = m_2 - 2n/k$, encoder $\mathcal{E}_2$ and decoder $\mathcal{D}_2$. Let $\mathcal{P}$ be a complete set of $(k/2)$-partitions of size $s = S(n, k/2)$, $\mathcal{P} = \{P_1, \ldots, P_s\}$. We construct an $(m,n,k)_B$ batch code of length $m = n + r_1 + (k/2)\cdot s \cdot r_2$ with the following encoder $\mathcal{E}$ and decoder $\mathcal{D}$.

Let $\boldsymbol{x}$ be a length-n input binary vector, then it is encoded by the first encoder $\mathcal{E}_1$ to receive r_1 redundancy bits $\boldsymbol{p}_0 = \mathcal{E}_1(\boldsymbol{x})$. For each $i \in [s]$, we encode using the i-th $(k/2)$-partition P_i as follows. The $(k/2)$-partition P_i is denoted by $P_i = \{U_{i,1}, \ldots, U_{i,k/2}\}$, and for $j \in [k/2]$, we encode

$$\mathcal{E}_2(\boldsymbol{x}_{U_{i,j}}) = \boldsymbol{p}_{i,j}.$$

Together, the vector $\boldsymbol{x}$ is encoded to the vector

$$(\boldsymbol{x}, \boldsymbol{p}_0, \boldsymbol{p}_{1,1}, \ldots, \boldsymbol{p}_{1,k/2}, \ldots, \boldsymbol{p}_{s,1}, \ldots, \boldsymbol{p}_{s,k/2}).$$

The decoder $\mathcal{D}$ receives a multiset request of k bits $S = [i_1, \ldots, i_k]$. Assume the requested bits without repetitions are $i_1, \ldots, i_{k'}$, and that the first $k^* \leqslant k'$ bits were requested more than once. Note that in particular, $k^* \leqslant k/2$. We consider the following two cases:

1) $k^* = 1$: In this case there are no two bits with more than one query and so according to Lemma 3 we can apply the decoder $\mathcal{D}_1$ with the encoded vector $(\boldsymbol{x}, \boldsymbol{p}_0)$ and the multiset request S to find k mutually disjoint recovering sets for the k bits.

2) $k^* > 1$: Since the set $\mathcal{P}$ is a complete set of $k/2$-partitions and $k^* \leqslant k/2$, there exists a $k/2$-partition $P_\ell = \{U_{\ell,1}, \ldots, U_{\ell,k/2}\} \in \mathcal{P}$ for some $\ell \in [s]$ which covers the set of indices $\{i_1, \ldots, i_{k^*}\}$. Therefore, for $j \in [k^*]$ we can decode the bit i_j with its repetitions by the decoder $\mathcal{D}_2$ using the information bits $\boldsymbol{x}_{U_{\ell,j}}$ and the parity part $\boldsymbol{p}_{\ell,j}$. Lastly, all the bits which are requested once will be decoded with the first bits i_1. Note that in all cases, the size of all input multiset requests to the decoding map $\mathcal{D}_2$ is at most $k-2$. ■

Lastly, in order to fully characterize the value of $r_B(n,k)$ we seek the study the value of $S(n,q)$. We accomplish this task in the following lemma.

Lemma 10. *For all n and q such that $q \leqslant n$, the following holds*

$$S(n,q) \leqslant \left\lceil \frac{\log \binom{n}{q}}{-\log\left(1 - \frac{q!}{q^q}\right)} \right\rceil.$$

Proof: First, note that it is possible to represent every q-partition of $[n]$ as a vector in $[q]^n$. We can also think in the other direction that every vector in $[q]^n$ represents a q-partition of $[n]$ while we allow some of the subsets to be empty (in case some of the symbols in $[q]$ do not appear in the vector).

For a subset $X \subseteq [n]$ of size q and a vector $\boldsymbol{v} \in [q]^n$ we say that $\boldsymbol{v}$ *covers* X if the projection of $\boldsymbol{v}$ on the q positions in X forms a permutation of $[q]$. Hence, the q-partition which is represented by the vector $\boldsymbol{v}$ also covers the set X.

For a fixed set X, if the vector $\boldsymbol{v}$ is chosen uniformly at random then the probability that $\boldsymbol{v}$ covers the set X is $q!/q^q$. Hence if we draw some s vectors independently uniformly at random from the set $[q]^n$, then the probability that none of them covers the set X is

$$\left(1 - \frac{q!}{q^q}\right)^s.$$

For each of the $\binom{n}{q}$ subsets X of size q, let Y_X be the corresponding indicator random variable Y_X as follows:

$$Y_X = \begin{cases} 0 & \text{if } X \text{ is covered by at least one of the } s \text{ vectors,} \\ 1 & \text{otherwise.} \end{cases}$$

Define $Z = \sum_{X \in \binom{[n]}{q}} Y_X$ to be the random variable which counts the number of subsets X which are *not* covered after s independent draws of vectors in $[q]^n$. By linearity of expectation we have that

$$E[Z] = \sum_{X \in \binom{[n]}{q}} E[Y_X] = \binom{n}{q}\left(1 - \frac{q!}{q^q}\right)^s.$$

Therefore, if $E[Z] \leqslant 1$ then there exists at least one realization of the s draws that achieves the value $Z = 0$, that is all subsets $X \subseteq [n]$ of size q will be covered by at least one of the vectors in this realization. Specifically, the value

$$s = \left\lceil \frac{\log \binom{n}{q}}{-\log\left(1 - \frac{q!}{q^q}\right)} \right\rceil$$

satisfies this requirement which proves the lemma's statement. ■

Combining the proof or Theorem 9 and Lemma 10 we get the following corollary.

Corollary 11. *For all n and k,*

$$r_B(n,k) \leqslant r_P(n,k) + \left\lfloor \frac{k}{2} \right\rfloor \left\lceil \frac{\log \binom{n}{\lfloor k/2 \rfloor}}{-\log\left(1 - \frac{\lfloor \frac{k}{2} \rfloor!}{\lfloor \frac{k}{2} \rfloor^{\lfloor \frac{k}{2} \rfloor}}\right)} \right\rceil r_P\left(\left\lceil \frac{n}{\lfloor \frac{k}{2} \rfloor} \right\rceil, k-2\right).$$

In particular if k is fixed then,

$$r_B(n,k) = O(\sqrt{n}\log(n)).$$

It is possible to apply the ideas of this construction also when k is not fixed, and we state here the following result.

Theorem 12. *For n and k large enough*

$$r_B(n,k) = O\left(k^{2.5}\log(n) \cdot e^{k/2} r_P(n,k)\right).$$

In particular for $k = \log(\log(n))$ we get

$$r_B(n, \log(\log(n))) = O\left((\log(\log n))^{2.5}(\log(n))^{\frac{2+\log e}{2}} r_P(n,k)\right).$$

REFERENCES

[1] S. Bhattacharya, S. Ruj, and B. Roy, "Combinatorial batch codes: A lower bound and optimal constructions," *Advances in Mathematics of Comm.*, vol. 6, pp. 165–174, 2012.

[2] R.A. Brualdi, K.P. Kiernan, S.A. Meyer, and M.W. Schroeder, "Combinatorial batch codes and transversal matroids," *Advances in Mathematics of Comm.*, vol. 4 pp. 419–431, 2010.

[3] C. Bujtás and Z. Tuza, "Relaxations of Halls condition: Optimal batch codes with multiple queries," *Appl. Anal. Discrete Math*, vol. 6, no. 1, pp. 72–81, 2012.

[4] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," *J. ACM*, 45, 1998. Earlier version in FOCS 95.

[5] A. Fazeli, A. Vardy, and E. Yaakobi, "PIR with low storage overhead: coding instead of replication," arXiv:1505.06241, May 2015.

[6] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin, "On the locality of codeword symbols," *IEEE Trans. on Inf. Theory*, vol. 58, no. 11, pp. 6925–6934, 2012.

[7] P. Huang, E. Yaakobi, H. Uchikawa, and P.H. Siegel, "Linear locally repairable codes with availability," *Proc. IEEE Int. Symp. on Inf. Theory*, pp. 1871–1875, Hong Kong, Jun. 2015.

[8] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, "Batch codes and their applications," *Proc. of the 36-sixth Annual ACM Symposium on Theory of Computing*, pp. 262–271, Chicago, ACM Press, 2004.

[9] S. Lin and D. J. Costello, *Error Control Coding*, Prentice Hall, 2004.

[10] H. Lipmaa and V. Skachek, "Linear batch codes," *Coding Theory and Applications, CIM Series*, vol. 3. pp. 245–253, 2015.

[11] J.L. Massey, *Threshold Decoding*, MIT Press, 1963.

[12] L. Pamies-Juarez, H.D.L. Hollmann, and F. Oggier, "Locally repairable codes with multiple repair alternatives," *Proc. IEEE Int. Symp. on Inf. Theory*, pp. 892–896, Istanbul, Turkey, Jul. 2013.

[13] D.S. Papailiopoulos and A.G. Dimakis, "Locally repairable codes," *Proc. IEEE Int. Symp. on Inf. Theory*, pp. 2771–2775, Cambridge, Jul. 2012.

[14] S. Rao and A. Vardy, "Lower bound on the redundancy of PIR codes," arxiv:1605.01869v1, May 2016.

[15] A.S. Rawat, D.S. Papailiopoulos, A.G., Dimakis, and S. Vishwanath, "Locality and availability in distributed storage" *Proc. IEEE Int. Symp. on Inf. Theory*, pp. 681–685, Honolulu, HI, Jul. 2014.

[16] A.S. Rawat, Z. Song, A.G. Dimakis, and A. Gál, "Batch codes through dense graphs without short cycles" *IEEE Int. Symp. on Inf. Theory*, pp. 1477–1481, Hong Kong, June 2015.

[17] N. Silberstein and A. Gál, "Optimal combinatorial batch codes based on block designs," *Designs, Codes and Cryptography*, pp. 1–16, Sep. 2014.

[18] I. Tamo and A. Barg, "A family of optimal locally recoverable codes," *IEEE Trans. on Inf. Theory*, vol. 60, no. 8, pp. 4661–4676, 2014.

[19] Z. Wang, H.M. Kiah and Y. Cassuto, "Optimal binary switch codes with small query size," *Proc. IEEE Int. Symp. on Inf. Theory*, pp. 636–640, Hong Kong, Jun. 2015.

[20] Z. Wang, O. Shaked, Y. Cassuto and J. Bruck, "Codes for network switches," *Proc. IEEE Int. Symp. on Inf. Theory*, pp. 1057–1061, Istanbul, Turkey, Jul. 2013.

[21] A. Wang and Z. Zhang, "Repair locality with multiple erasure tolerance," *IEEE Trans. on Inf. Theory*, vol. 60, no. 11, pp. 6979–6987, 2014.

[22] S. Yekhanin, "Private information retrieval," *Comm. of the ACM*, vol. 53, no. 4, pp. 68–73, 2010.

[23] H. Zhang and V. Skachek, "Bounds for batch codes with restricted query size," arXiv:1510.08883v1, October 2015.