UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Kristiina Oksner

Study of batch codes with small recovery sets

Master's Thesis (30 ECTS)

Supervisor(s): Assoc. Prof. Vitaly Skachek
Dr. Ago-Erik Riet
Prof. Henk D. L. Hollmann

Tartu 2025

Study of batch codes with small recovery sets

Abstract:
In 2004, batch codes were introduced for load-balancing in distributed data storage by Y. Ishai, E. Kushilevitz, R. Ostrovsky and A. Sahai.

Assume that we are given a $k \times n$ binary matrix $\mathbf{G}$ such that the matrix consists of n vectors of length k in $\mathbb{F}_2$. For every vector $\mathbf{v} \in \mathbb{F}_2^k$, a recovery set with respect to $\mathbf{G}$ is a (minimal) collection of columns of $\mathbf{G}$ with a sum $\mathbf{v}$. We wish to recover t nonzero vectors using disjoint recovery sets. Note that the recovery set can be of any size. This situation is called a functional t-batch code scheme, which was first introduced by Y. Zhang, T. Etzion and E. Yaakobi in 2019. They also introduced a lower bound on the parameter n. In other words, they showed how many columns the matrix $\mathbf{G}$ should have so that any t vectors could be recovered using the columns of matrix $\mathbf{G}$.

In this thesis, we investigate what happens to the parameter n if we assume that the size of the recovery set of each vector is at most r. We showcase different possible modifications to Zhang, Etzion and Yaakobi proof of the lower bound. We derive two new lower bounds on parameter n if the size of the recovery set is at most r and compare them to each other and the Zhang, Etzion and Yaakobi bound for specific values r, t and k.

Keywords:
Coding theory, batch code, distributed data storage, load-balancing, functional batch code.

CERCS: P170 Computer science, numerical analysis, systems, control.

Väikeste tagastushulkadega partiikoodide uurimine

Lühikokkuvõte:

Aastal 2004 defineeriti Y. Ishai, E. Kushilevitz'i, R. Ostrovsky ja A. Sahai poolt hajusfailisüsteemide koormusjaotuseks partiikoodid.

Olgu antud $k \times n$ kahendmaatriks nii, et see maatriks koosneb n vektorist pikkusega k üle korpuse $\mathbb{F}_2$. Iga vektori $\mathbf{v} \in \mathbb{F}_2^k$ jaoks on tagastushulk maatriksi $\mathbf{G}$ suhtes (minimaalne) maatriksi $\mathbf{G}$ veergude kogum, mille summa on $\mathbf{v}$. Me tahame tagastada t nullvektorist erinevat vektorit, kasutades ühisosata tagastushulki. Paneme tähele, et tagastushulga suurus ei ole piiratud. Sellist olukorda nimetatakse funktionaalseks t-partiikoodi skeemiks, mis defineeriti Y. Zhang'i, T. Etzion'i ja E. Yaakobi poolt aastal 2019. Lisaks tõestasid nad parameetri n alamtõket. Teiste sõnadega nad näitasid kui palju veerge on vähemalt vaja maatriksil $\mathbf{G}$, et oleks võimalik tagastada mistahes t vektorit, kasutades maatriksi $\mathbf{G}$ veerge.

Lõputöös uuritakse, mis juhtub parameetriga n, kui oletada, et iga vektori tagastushulga suurus on ülimalt r. Me näitame erinevaid viise, kuidas Zhang'i, Etzion'i ja

Yaakobi alamtõket kohandada saab. Me saame parameetri n kaks uut alamtõket, kui tagastushulga suurus on ülimalt r, ja võrdleme saadud tõkkeid üksteisega ja Zhang'i, Etzion'i ja Yaakobi tõkkega kindlate r, t ja k väärtuste puhul.

Võtmesõnad:
Kodeerimisteooria, partiikood, hajus andmetalletus, koormusjaotus, funktsionaalne partiikood.

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria).

Acknowledgments

I would like to express my deepest gratitude to my supervisors, Assoc. Prof. Vitaly Skachek, Dr. Ago-Erik Riet and Prof. Henk D.L. Hollmann for their invaluable guidance and support throughout the research process.

I also thank the Estonian National Culture Foundation for their financial support through the Monika Oit Foundation Fellowship. Their support played a significant role in enabling me to complete this thesis and focus on my research during a critical stage of my academic journey.

Lastly, I would like to thank my family and friends who have been a huge support through my Master's studies.

Contents

1 Introduction

Batch codes were first introduced in [4] for load balancing in the multi-server distributed data storage systems.

Consider a data storage system with multiple servers. Assume that one server stores bit x_1 and the other server stores bit x_2. Also assume that two users cannot be served by the same server at the same time. When multiple users try to access the same server, that creates uneven load on the system. Batch codes help us to solve this issue without replicating the data. In our example, it is enough to have 3 servers: one stores bit x_1, the second server stores bit x_2. and the third server stores bit $x_1 + x_2$. This way, if two users want to access the same bit, for example x_1, then it is possible: indeed, the first user reads x_1 directly, and the second user reads x_2 and $x_1 + x_2$ and by a simple subtraction it can obtain x_1.

Batch codes are also potentially useful in network switches. Consider the following example with switches from [10]. Two bits of the input data are written on two servers, and two users attempt to read a bit of data in every clock cycle: in the first clock cycle bits A and B are written and read, in the second it is C and D and in the third clock cycle E and F. This situation is depicted in Figure 1 (a). However, if the users try to read the same data, then only one request can be satisfied because each server is able to deliver one bit per clock cycle. One simple solution to this problem is replication. This means having each server duplicated and in total having four servers, as can be seen in Figure 1 (b).

There exists another solution that does not require us to double the number of servers. In that approach, termed "batch code", a third server is added. For example, if the first two servers store A and B, respectively, then the third server will store A + B. In that way, two users can read bit A during one clock cycle. The system stores bits A, B and A + B, which can be viewed as encoding of the vector (A,B) into (A,B, A + B) by multiplying it by the matrix

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

with 3 columns or *length* 3.

Therefore, we would be employing three servers as shown in Figure 1 (c). This way, if the first and the second user request bit A, then the first user can read A from the first server and the second user reads B from the second server, and A + B from the third server and computes A. Using a batch code can reduce the storage overhead with additional cost of communication and computations.

Note that if users request bits that are on different servers, then they can read them from different servers in the same clock cycle. For example, if the first user requests bit A and the second user requests bit B, then the first user can read A from the first server and the second user can read B from the second server.

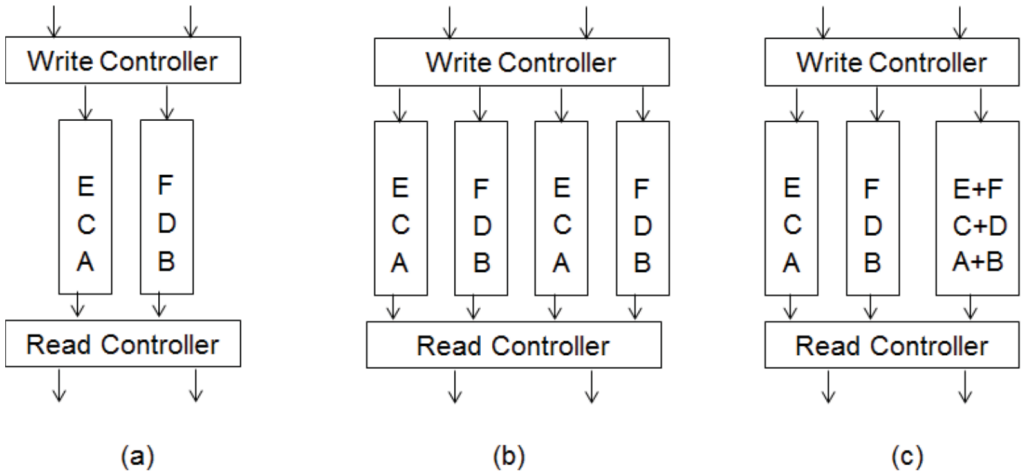


Figure 1. A switch for two ports with (a) direct input and output, (b) replication and (c) batch-encoded data

Functional batch codes, a broader definition of batch codes, were introduced in [12]. Unlike batch codes, where only information symbols could be retrieved, functional batch codes can serve any linear combination of information symbol. In [12], a lower bound on the length of a functional batch code was derived.

In a batch and a functional batch code, the number of columns of the generator matrix that can be used for serving a user's request is unlimited. We investigate such functional batch codes, where the number of columns used from the generator matrix is small. In particular, we investigate how the lower bound on the length of the code changes if we restrict the size of the recovery set.

We begin by describing the background that is needed for subsequent chapters. In the second chapter, we introduce about batch codes and their application to private information retrieval. In particular, we define the batch codes with small recovery sets and provide necessary examples. In the third chapter, we derive bounds on the length of the generator matrix for batch codes with small recovery sets and compare the derived bounds with the result in the literature.

2 Preliminaries

This chapter provides some definitions, lemmas, theorems and the main tools employed in the work.

2.1 Combinatorial counting and trees

In this subsection, we present different definitions and theorems related to combinatorial counting and trees. The definitions and theorems presented here can be found in [7].

Definition 1 (Binomial coefficient)**.** Let $n \geq k$ be nonnegative integers. The *binomial coefficient* $\binom{n}{k}$ is a function of the variables n, k defined by the formula

$$\binom{n}{k} := \frac{n(n-1)(n-2)\dots(n-k+1)}{k(k-1)\dots 1} = \frac{\Pi_{i=0}^{k-1}(n-i)}{k!}.$$

The above expression is equal to

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}.$$

Theorem 1 (Binomial theorem)**.** For any nonnegative integer n, we have

$$(x+y)^n = \sum_{i=0}^{n} \binom{n}{i} x^i y^{n-i}.$$

Now, we present the definition of the multinomial coefficient.

Definition 2 (Multinomial coefficient)**.** Given that $k_1 + \ldots + k_m = n$, the expression $\frac{n!}{k_1!\ldots k_m!}$ is called a *multinomial coefficient* and is usually written as $\binom{n}{k_1,\ldots,k_m}$.

Theorem 2 (Multinomial theorem)**.** For arbitrary real numbers $x_1, x_2, \ldots, x_m$ and any natural number $n \geq 1$, the following identity holds:

$$(x_1 + x_2 + \ldots + x_m)^n = \sum_{\substack{k_1+\ldots+k_m=n \\ k_1,\ldots,k_m \geq 0}} \binom{n}{k_1, \ldots, k_m} x_1^{k_1} x_2^{k_2} \ldots x_m^{k_m}.$$

Given a power series

$$S(x) = S_0 + S_1 x + \ldots + S_n x^n + \ldots,$$

we denote by $C_{x^n}(S(x))$ the coefficient S_n of term x^n in this sum.

Lastly, let us define trees.

Definition 3 (Graph)**.** A (simple) *graph* G is an ordered pair (V, E), where V is some set and E is a set of 2-element subsets of V. The elements of the set V are called vertices of the graph G and the elements of E are called edges of G.

Definition 4 (Path)**.** A *path* of length l from u to v in G is a finite sequence of edges

$$\{u, u_1\}, \{u_1, u_2\}, \{u_2, u_3\}, \ldots, \{u_{l-2}, u_{l-1}\}, \{u_{l-1}, v\}, \tag{1}$$

where $u, u_1, u_2, u_3, \ldots, u_{l-1}, v \in V$ are distinct.

Definition 5 (Cycle)**.** A path as in (1) is called a *cycle* if $u, u_1, u_2, u_3, \ldots, u_{l-1} \in V$ are distinct but $u = v$.

Definition 6 (Connected graph)**.** A graph G is *connected* if for every two distinct vertices u and v in G there is a path from u to v in G.

Definition 7 (Tree)**.** A *tree* is a connected graph containing no cycle.

Definition 8 (Rooted tree)**.** A *rooted tree* is a pair (T, r), where T is a tree and r is a distinguished vertex of T called the root. If $\{x, y\}$ is an edge and the vertex x lies on the unique path from y to the root, we say that x is the father of y (in the considered rooted tree) and y is a child of x.

2.2 Vector spaces

In this subsection, we provide the main algebraic definitions used in the following chapters. Those definitions can be found in [1].

Definition 9 (Group)**.** A *group* $(G, +)$ consists of a non-empty set G and a binary operation $+$ on G, which satisfies the following:

- associativity: $(a + b) + c = a + (b + c)$, for every $a, b, c \in G$;
- identity element: there exists an element $0 \in G$ such that $a + 0 = a = 0 + a$, for every $a \in G$, the element 0 is called the identity element;
- inverse: for every $a \in G$, there exists an element $-a \in G$, called the inverse of a, such that $a + (-a) = 0 = (-a) + a$.

Definition 10 (Abelian group)**.** The group G is called *abelian* if it satisfies the commutativity, or $a + b = b + a$, for every $a, b \in G$.

Definition 11 (Field)**.** A *field* $(\mathbb{F}, +, \cdot)$ is a set of elements $\mathbb{F}$ with two binary operations $+$ and $\cdot$, which are called addition and multiplication, respectively, such that they satisfy the following:

$+$	0	1
0	0	1
1	1	0

Table 1. Addition

$\cdot$	0	1
0	0	0
1	0	1

Table 2. Multiplication

- $(\mathbb{F}, +)$ is an abelian group with identity element denoted as 0;
- distributivity: $a \cdot (b + c) = a \cdot b + a \cdot c$, for every $a, b, c \in \mathbb{F}$;
- $(\mathbb{F} \setminus \{0\}, \cdot)$ is an abelian group with identity element denoted as 1.

Example 1 ($\mathbb{F}_2$)**.** $\mathbb{F}_2$ is a field with two elements 0 and 1, where 1 is the inverse element of itself with respect to addition, which is equal to 1. The multiplication and addition in $\mathbb{F}_2$ is as in Tables 1 and 2. Note that multiplication and addition in $\mathbb{F}_2$ are multiplication and addition modulo 2.

Definition 12 (Finite field)**.** A *finite field* is a field with finitely many elements. Every finite field has a size equal to a *prime power*. In fact, for every size q of the form $q = p^r$, with p a prime, there is a *unique* field of size q, denoted by $\mathbb{F}_q$. The prime p is called the *characteristic* of $\mathbb{F}_q$. The importance of the characteristic is illustrated by the *Freshman's Dream*, stating that for all $x, y \in \mathbb{F}_q$, we have $(x + y)^p = x^p + y^p$.

Definition 13 (Vector space)**.** A *vector space* or *linear space* over a field $\mathbb{F}$ is a set V, the elements of which are called *vectors*, with two operations:

- addition of vectors, which is denoted by $+ : V \times V \to V$, such that $(V, +)$ is an abelian group;
- multiplication of an element of $\mathbb{F}$ by a vector, which is denoted by $\cdot : \mathbb{F} \times V \to V$, satisfies the following:
  - $(kl)\mathbf{v} = k(l\mathbf{v})$, for every $k, l \in \mathbb{F}$ and every $\mathbf{v} \in V$;
  - $k(\mathbf{v}_1 + \mathbf{v}_2) = k\mathbf{v}_1 + k\mathbf{v}_2$, for every $k \in \mathbb{F}$ and every $\mathbf{v}_1, \mathbf{v}_2 \in V$;
  - $(k + l)\mathbf{v} = k\mathbf{v} + l\mathbf{v}$, for every $\mathbf{v} \in V$ and every $k, l \in \mathbb{F}$;
  - $1\mathbf{v} = \mathbf{v}$, for every $\mathbf{v} \in V$ and where 1 is the identity element of $\mathbb{F}$, with respect to multiplication.

Definition 14 (Linear independence)**.** A set of vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ of a vector space V over a field $\mathbb{F}$ is called *linearly independent* if for any $k_1, k_2, ..., k_n \in \mathbb{F}$ that satisfy

$$k_1\mathbf{v}_1 + k_2\mathbf{v}_2 + \ldots + k_n\mathbf{v}_n = 0$$

it follows that

$$k_1 = k_2 = ... = k_n = 0.$$

Definition 15 (Linear combination)**.** Let V be a vector space over field $\mathbb{F}$ and $\mathbf{v}_1, \mathbf{v}_2, .., \mathbf{v}_n \in V$. Any expression

$$k_1\mathbf{v}_1 + k_2\mathbf{v}_2 + ... + k_n\mathbf{v}_n,$$

where $k_1, k_2, ..., k_n \in \mathbb{F}$, is called a *linear combination* of vectors $\mathbf{v}_1, \mathbf{v}_2, .., \mathbf{v}_n$. A linear combination can also refer to the element, which is defined by this expression.

Definition 16 (Span of a set of vectors)**.** The *span* of a set of vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ is the collection of all linear combinations of vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$. Note that this is a vector space, a subspace of V.

Definition 17 (Spanning set)**.** A finite set M of vectors of a vector space V is called a *spanning set* if every vector of V can be represented as a linear combination of the vectors in M.

Definition 18 (Basis)**.** A *basis* of a vector space is a linearly independent spanning set.

Definition 19 (Dimension)**.** The *dimension* of vector space V is the cardinality of a basis of this vector space. The symbol $\dim(V)$ is used for the dimension of vector space V.

2.3 Codes

We provide the main coding theory definitions used further in the work. The definitions can be found, for example, in [8].

Definition 20 (Code)**.** An (n, M) *code* over a finite alphabet $\mathbb{F}$ is a nonempty subset C of size M of $\mathbb{F}^n$, where n is called the length of the code and M is the code size.

Definition 21 (Codewords)**.** *Codewords* are the elements of the code.

Definition 22 (Hamming distance)**.** Let $\mathbb{F}$ be an alphabet. The *Hamming distance* is defined as the number of coordinates in which vectors $\mathbf{x} \in \mathbb{F}^n$ and $\mathbf{y} \in \mathbb{F}^n$ differ. The Hamming distance between $\mathbf{x}$ and $\mathbf{y}$ is denoted by $\mathrm{d}(\mathbf{x}, \mathbf{y})$. Formally, if $\mathbf{x} = x_1 \ldots x_n$ and $\mathbf{y} = y_1 \ldots y_n$, then $\mathrm{d}(\mathbf{x}, \mathbf{y}) = |\{i \,:\, x_i \neq y_i\}|$.

Definition 23 (Hamming weight)**.** Let $\mathbb{F}$ be an abelian group. The *Hamming weight* of $\mathbf{v} \in \mathbb{F}^n$ is the number of nonzero entries in $\mathbf{v}$. We denote the Hamming weight by $w(\mathbf{v})$. Notice that for every two words $\mathbf{x}, \mathbf{y} \in \mathbb{F}^n$,

$$\mathrm{d}(\mathbf{x}, \mathbf{y}) = w(\mathbf{x} - \mathbf{y}).$$

Definition 24 (Minimum distance)**.** Let C be an (n, M) code over $\mathbb{F}$, where $M > 1$. The *minimum distance* of C is the minimum Hamming distance between two different codewords $\mathbf{c}_1, \mathbf{c}_2 \in C$ or in other words the minimum distance d is written as

$$d = \min_{\mathbf{c}_1 \neq \mathbf{c}_2} \mathrm{d}(\mathbf{c}_1, \mathbf{c}_2).$$

We can denote an (n, M) code with a minimum distance d as an (n, M, d) code. However, it is important to remember that using an (n, M, d) code implies that $M > 1$.

Definition 25 (Linear code)**.** An (n, M, d) code C over a field $\mathbb{F}$ is called *linear* if C is a linear subspace of $\mathbb{F}^n$ over $\mathbb{F}$.

Definition 26 (Dimension of a linear code)**.** The *dimension of a linear* (n, M, d) *code* C over $\mathbb{F}$ is the dimension of C as a linear subspace of $\mathbb{F}^n$.

A linear code over $\mathbb{F}$ is also denoted as an $[n, k, d]$ code, where k is the dimension of C.

Definition 27 (Generator matrix)**.** A *generator matrix* of a linear $[n, k, d]$ code over $\mathbb{F}$ is a $k \times n$ matrix, where rows form a basis of the code.

Definition 28 (Simplex code)**.** A binary $[2^k - 1, r]$ *simplex code* is a linear code of length $n = 2^k - 1$ and dimension k over $\mathbb{F}_2$ whose binary $k \times n$ generator matrix G contains each nonzero vector z of length k exactly once as a column.

Example 2. For $k = 3$, the generator matrix of the simplex code is

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

2.4 Exponential generating functions

Consider the following counting problems [3]:

1. Find for each n the number of partitions of an n-element set;
2. Find for each n the number of permutations of an n-element set;
3. Find for each n the number of subsets of an n-element set;
4. Find for each n the number of labelled trees on a set of n vertices.

In all of the listed problems, we begin for each n with a set of n elements and then we count all of the possible ways to impose a certain structure upon the given set [3]. We will not formally define a structure but note that structures can be ordered or unordered. For example, we wish to arrange the elements in some order for a permutation, or we wish to arrange the elements into those chosen to belong to the subset, and those not chosen for a subset [3]. Exponential generating functions correspond to ordered structures while ordinary generating functions correspond to unordered structures.

Exponential generating functions help us to solve counting problems [3].

Definition 29 (Exponential generating function)**.** The *exponential generating function* associated with a counting problem for structures is

$$F(x) = \sum_n f(n)\frac{x^n}{n!},$$

where $f(n)$ is the number of structures on an n-element set.

We provide a few examples of exponential generating functions with different structures.

Example 3. [3]

1. The structure of "set": $F(x) = e^x$. Note that conventionally, $\sum_{n=0}^{\infty} \frac{x^n}{n!}$ is equal to e^x as it is the Maclaurin series for e^x.
2. The "1-element set" structure: $F(x) = x$.
3. The "empty set" structure: $F(x) = 1$.
4. The "non-empty set" structure: $F(x) = e^x - 1$.
5. The "even-size set" structure: $F(x) = \cosh(x) = (e^x + e^{-x})/2$.
6. The "odd-size set" structure: $F(x) = \sinh(x) = (e^x - e^{-x})/2$.

Those structures in the example are usually used as building blocks. To be able to use them as such, we introduce the addition and multiplication principles of exponential generating functions. Let us assume that we have given names to some structures. Let them be f-structure, g-structure and h-structure. Let the exponential generating function counting f-structures be $F(x) = \sum f(n)x^n/n!$, where $f(n)$ is the number of f-structures on an n-element set. The same can be done for g and h.

Lemma 3. [3] Suppose that the set of f-structures on each set is the disjoint union of the set of g-structures and the set of h-structures. Then

$$F(x) = G(x) + H(x).$$

Definition 30 ($g \times h$ structure)**.** Let g and h denote two types of structures on finite sets. A $g \times h$ *structure* on a set A consists of

1. an ordered partition of A into disjoint subsets $A = A_1 \cup A_2$;
2. a g-structure on A_1;
3. an h-structure on A_2;

where the structures in 2 and 3 are chosen independently.

Lemma 4. [3] If $G(x)$ and $H(x)$ are the exponential generating functions for g-structures and h-structures, respectively, then the exponential generating function for $g \times h$ structures is

$$F(x) = G(x)H(x).$$

It is possible to generalize the above principle to the product of three or more generating functions as shown in [3]. Specifically, a $g_1 \times g_2 \times \ldots \times g_r$ structure on A consists of an ordered partition of A into r disjoint subsets $A = A_1 \cup A_2 \cup \ldots \cup A_r$, and independently chosen g_i structures on each subset A_i. This means that a $g_1 \times g_2 \times \ldots \times g_r$ structure is equivalent to a $g_1 \times (g_2 \times \ldots \times g_r)$ structure, and by induction on r, it can be proven that the generating function for $g_1 \times g_2 \times \ldots \times g_r$ structures is

$$F(x) = G_1(x)G_2(x) \ldots G_r(x).$$

We provide the following example [9]:

Example 4. Let us find the number of 3-permutations from the multiset

$$\{2 \cdot a, 3 \cdot b\}.$$

Let $G(x)$ and $H(x)$ be the exponential generating functions for the permutations of the multisets g and h. Then $F(x) = G(x)H(x)$ is the exponential generating function for the permutations of the combined multisets. The exponential generating function for $g = \{2 \cdot a\}$ is

$$G(x) = \frac{1}{0!} + \frac{1}{1!}x + \frac{1}{2!}x^2$$

and exponential generating function for $h = \{3 \cdot b\}$ is

$$H(x) = \frac{1}{0!} + \frac{1}{1!}x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3.$$

Therefore,

$$F(x) = G(x)H(x) = \ldots + x^3 \left(\frac{1}{0!} \cdot \frac{1}{3!} + \frac{1}{1!} \cdot \frac{1}{2!} + \frac{1}{2!} \cdot \frac{1}{1!} \right) + \ldots =$$
$$= \ldots + \frac{x^3}{3!} \left(\frac{3!}{0!3!} + \frac{3!}{1!2!} + \frac{3!}{2!1!} \right) + \ldots.$$

We get that the number of 3-permutations from the given multiset is $\frac{3!}{0!3!} + \frac{3!}{1!2!} + \frac{3!}{2!1!} = 7$.

3 Batch codes

In this section, we introduce batch codes and their application to private information retrieval. Let us begin with the description of the problem that batch codes can help to solve, which was introduced in [4].

Assume there is an n-symbol database and it is distributed amongst M devices. A client would like to retrieve an arbitrary subset (or *batch*) of t symbols by reading the data stored on the devices. We would like to minimize the worst-case maximal number of bits read from a device. In addition, we would like to use as little storage space as possible.

The authors of [4] used a notation of batch code to abstract the problem and defined the notion as follows:

Definition 31 (Batch code)**.** An (k, n, t, M, τ) *batch code* $\mathcal{B}$ over an alphabet Σ encodes a string $x \in \Sigma^k$ into an M-tuple of strings $\mathbf{y}_1, \ldots, \mathbf{y}_M$ over Σ (also referred to as buckets) of total length n, such that for each t-tuple (*batch* or *requests*) of distinct indices $i_1, \ldots, i_t \in [k]$, the entries $x_{i_1}, \ldots, x_{i_t}$ can be recovered by reading at most τ symbols from each bucket.

As in [4], we define the rate of the code as n/N. For simplicity, we may refer to an (k, n, t, M, τ) batch code as a t-batch code.

Definition 32 (Multiset batch code)**.** An (k, n, t, M, τ) *multiset batch code* is an (k, n, t, M, τ) batch code which also satisfies the following property: For any multiset $i_1, \ldots, i_t \in [k]$ there is a partition of the buckets into t subsets $S_1, \ldots, S_t \subseteq [M]$ such that each symbol $x_{i_j}, j \in [t]$, can be recovered by reading at most τ symbols from each bucket in S_j.

Definition 33 (Primitive batch code)**.** An (k, n, t, M) (multiset) batch code is an $(k, n, t, M, 1)$ (multiset) batch code and a *primitive (multiset) batch code* is an (k, n, t, M) (multiset) batch code in which each bucket contains a single symbol, that is $n = M$.

In the following, we consider primitive multiset batch codes with $n = M$ and $\tau = 1$.

Definition 34 (Linear batch code)**.** We say that an (k, n, t, M) primitive multiset batch code over a finite field $\mathbb{F}$ is *linear*, if every symbol in every bucket is a linear combination of original database symbols. We refer to linear batch code simply as $[n, k, t]$-batch code.

From now on, we will use linear primitive multiset batch codes with $n = M$ and $\tau = 1$. For simplicity, we may omit linear primitive multiset and refer to them as batch code or t-batch code.

Each encoded symbol $\mathbf{y}_i$, $i \in [n]$, can be written as $\mathbf{y}_i = \sum_{j=1}^{k} g_{j,i} x_j$ for some symbols $g_{j,i} \in \mathbb{F}_q$, $j \in [k]$, $i \in [n]$. This can be done due to the linearity of the code and it allows us to form matrix $\mathbf{G}$ as follows:

$$\mathbf{G} = (g_{j,i})_{j\in[k],i\in[n]}.$$

In other words, the encoding is $\mathbf{y} = \mathbf{x}G$.

The $k \times n$ binary matrix $\mathbf{G}$ has similar functions as a generator matrix for a classical linear error-correcting codes. We refer to $\mathbf{G}$ as a generator matrix of the batch code $\mathcal{B}$.

Each of t requests has its own recovery set, which is defined as follows [12]:

Definition 35 (Recovery set for a vector)**.** Let $\boldsymbol{G}$ be a $k \times n$ matrix over $\mathbb{F}_q$. A *recovery set for a (nonzero) vector* $\boldsymbol{r} \in \mathbb{F}_q^k \setminus \{\mathbf{0}\}$ is a set $R \subset [n]$ of column indices such that

$$\boldsymbol{r} \in \langle \mathbf{g}_j \mid j \in R \rangle. \tag{2}$$

We may also define a more special case of recovery sets for an information symbol [12]:

Definition 36 (Recovery set for an information symbol)**.** Let $\boldsymbol{G}$ be a $k \times n$ matrix over $\mathbb{F}_q$. A *recovery set for an information symbol* $r \in [k]$ is a set $R \subset [n]$ of column indices such that

$$\boldsymbol{e}_r \in \langle \mathbf{g}_j \mid j \in R \rangle, \tag{3}$$

where $\boldsymbol{e}_r$ denotes the rth unit vector in $\mathbb{F}_q^k$ and $\mathbf{g}_j$ denotes the jth column of $\boldsymbol{G}$.

Theorem 5. (See Theorem 1 in [6]) Let $\mathcal{B}$ be an $[n, k, t]$ batch code. It is possible to retrieve $x_{i_1}, x_{i_2}, \ldots, x_{i_t}$ simultaneously if and only if there exist t non-intersecting sets $T_1, T_2, \ldots, T_t$ of indices of columns in $\mathbf{G}$, and for T_r there exists a linear combination of columns of $\mathbf{G}$ indexed by that set, which equals to the column vector $\mathbf{e}_{i_r}^T$, for all $r \in [t]$.

We provide an example of a linear batch code, taken from [6]:

Example 5. Consider the following linear binary batch code $\mathcal{B}$ whose 4×9 generator matrix is given by

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

Let $\mathbf{x} = (x_1, x_2, x_3, x_4)$, $\mathbf{y} = \mathbf{x}\mathbf{G}$.

Assume that we want to retrieve the values of (x_1, x_1, x_2, x_2). We can retrieve (x_1, x_1, x_2, x_2) from the following set of equations:

$$\begin{cases} x_1 = y_1 \\ x_1 = y_2 + y_3 \\ x_2 = y_5 + y_8 \\ x_2 = y_4 + y_6 + y_7 + y_9 \end{cases}.$$

It is possible to verify that any 4-tuple $(x_{i_1}, x_{i_2}, x_{i_3}, x_{i_4})$, where $i_1, i_2, i_3, i_4 \in [4]$, can be retrieved by using columns indexed by some four nonintersecting sets of indices in [9]. That is why, the code $\mathcal{B}$ is a $[9, 4, 4]$ linear batch code.

Definition 37 (Functional (linear primitive multiset) batch code)**.** An $[n, k, t]$ *functional (linear primitive multiset) batch code* over $\mathbb{F}$ (or simply t-functional batch code) is a batch code with the requests being a linear combination of the k bits of information symbols.

In [12], the authors introduced a lower bound on the parameter n of a functional batch code as follows:

Theorem 6. (See Theorem 23 in [12]) Suppose that the $k \times n$ binary matrix $\mathbf{G}$ is a t-functional batch code. Then

$$\lim_{k\to\infty} \frac{n}{k} \geq \frac{t}{\log(t+1)}.$$

With each sequence $R_1, \ldots, R_t$ of mutually disjoint recovery sets, we associate a labelling of the columns of $\mathbf{G}$ with labels i, $1 \leq i \leq t$, where column j receives label i if $j \in R_i$ and receives label 0 if it does not occur in any of the recovery sets. The number of such labellings is $(t+1)^n$. On the other hand, the number of possible request sequences $\mathbf{v}_1, \ldots, \mathbf{v}_t$ with each $\mathbf{v}_i$ a distinct nonzero vector from $\mathbb{F}_2^k$ equals $(2^k - 1)^t$. We conclude that $\lim_{k\to\infty} \frac{n}{k} \geq \frac{t}{\log(t+1)}$.

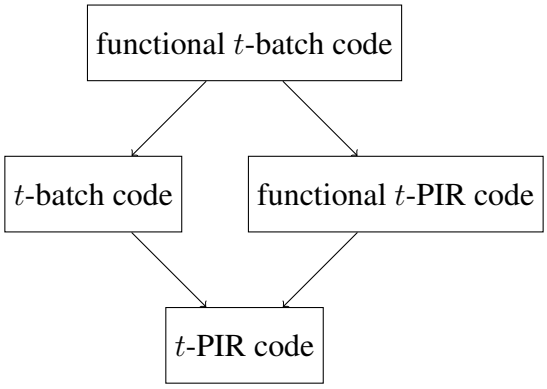
We also introduce the definition of Private Information Retrieval (PIR) codes [2]:

Definition 38 (PIR code)**.** A $k \times n$ binary matrix G has property $\mathcal{A}_t$ if for all $i \in [k]$, there exist t disjoint sets of columns of G that add up to $\boldsymbol{e}_i$. A binary linear $[n, k]$ code C is called a t-server *PIR code* (or, simply, PIR code) if there exists a generator matrix G for C with property $\mathcal{A}_t$.

In [12], a functional PIR code was introduced:

Definition 39 (Functional PIR code)**.** A *functional PIR code* is a PIR code with the requests being a linear combination of the k bits of information symbols.

The relations between a (linear primitive multiset) batch code, a functional (linear primitive multiset) batch code, a PIR code and a functional PIR code can be represented as the following graph, taken from [12]:



3.1 Batch codes with restricted query size

We might be interested in the situation where we restrict the size of the recovery sets of requests. In this subsection, we define batch codes with restricted query size and provide corresponding examples.

The definition of a batch code with restricted query size comes from [11]:

Definition 40 (Batch code with restricted query size)**.** A *batch code with restricted query size* r is a batch code that can serve all its request sequences with recovery sets of size at most r.

Example 6. Let us look at Example 5 once again. Take $r = 2$ and in this case x_2 cannot be recovered as a sum of vectors $\{y_4, y_6, y_7, y_9\}$ because the size of the recovery set exceeds the allowed size 2. As can be seen there are no other ways to recover x_2 the second time in the sequence of requests x_1, x_1, x_2, x_2 meaning the batch code $\mathcal{B}$ is no longer a $[9, 4, 4]$ linear batch code. It can be verified that now the code is a $[9, 4, 2]$ linear batch code.

It can also be seen that a doubled simplex code is an example of a functional batch code with the size of the recovery sets being at most 2. For that, we state the following theorem, which was proved in [5]:

Theorem 7. Let m be the largest integer such that G contains as columns at least $2m$ copies of each nonzero vector of $\mathbb{F}_2^k$. Then G is a $m2^k$-functional batch code, where moreover all recovery sets have size at most 2.

Example 7. By taking $m = 1$ in Theorem 7, we get that the doubled simplex code, that has 2 copies of each nonzero vector of $\mathbb{F}_2^k$, is a 2^k functional batch code, where the size of each recovery set is at most 2.

3.2 Application to private information retrieval

As mentioned before, in a situation where a client wants to retrieve an arbitrary subset of data by reading the data stored in a database, with batch codes we try to minimize the worst-case maximal number of bits read from a device and we try to use as little storage space as possible.

In addition, as shown in [4], batch codes help us to reduce the time complexity of private information retrieval (PIR). Consider the $\binom{n}{t}$-PIR problem, where the user wants to obtain t bits from an n-bit string x. The simple solution to this problem is picking an arbitrary PIR protocol $\mathcal{P}$ and applying it t times. This would mean that the time complexity of the servers is at least $t \cdot n$. With batch codes we are able to get a general reduction from $\binom{n}{t}$-PIR to standard $\binom{n}{1}$-PIR. Let us fix a (k, N, m, M, t) batch code which will be used for encoding database x. Assume that the user wants to retrieve indices $i_1, \ldots, i_t$. The user applies the code's batch-decoding procedure to the set but not directly. Instead of reading one bit from each bucket, it applies the PIR protocol $\mathcal{P}$ on each bucket. This way, it is possible to retrieve the needed bit from the bucket while keeping the identity of this bit private. Let $C(n)$ be communication complexity and $T(n)$ be time complexity of the PIR protocol $\mathcal{P}$ and let $N_1, \ldots, N_m$ be the sizes of the buckets created by the batch code. Then, we get that the communication complexity of this solution is $\sum_{i=1}^{m} C(N_i)$ and its time complexity is $\sum_{i=1}^{m} C(T_i)$. This reduction can be applied to both information-theoretic and computational PIR protocols as it does not introduce any error or compromise privacy.

4 Bounds on the length of a batch code

This thesis aims to give an analogous bound as in Theorem 6 for batch codes with small recovery sets. Recall that the bound in Theorem 6 was found by comparing the number of ways to assign labels to the generator matrix to the number of possibilities for requests. Now labels i, where $1 \leq i \leq t$, have r copies and label 0 still has an unlimited number of copies as we do not restrict the number of unused columns. This means that we have to find the number of different labellings on the matrix, where labels from 1 to t have a limited number of copies.

In the proof of Theorem 6, it was assumed that the requests are distinct. Hence, the number of ways to choose the sequence of t requests is $\binom{2^k-1}{t}t!$ in the proof of Theorem 6. However, we allow requests to be repeated. In this case, there are $(2^k-1)^t$ ways to choose them, which we will be using in the derivations of our bounds. The use of the estimate $(2^k-1)^t$ allows for a slightly more accurate estimate on the number of the request sequences, when compared with the analogous proof in [12].

In this section, we derive lower bounds on the length of the generator matrix of a batch code with small recovery sets.

4.1 Approach 1

Our problem setup is essentially a setup for counting problems of exponential generating functions. As in Example 4 in section 2.4, we are given a multiset with an unlimited number of labels 0 and at most r labels i, where $1 \leq i \leq t$, and we want to find the number of n-permutations on this multiset. This means that in the exponential generating function $F(x) = \sum_n f(n)x^n/n!$, we are interested in finding $f(n)$, where n is the length of the generator matrix $\mathbf{G}$ of a batch code. In this case, the value $f(n)$ is the number of labels of matrix $\mathbf{G}$ given an unlimited number of labels 0 and at most r copies of labels i.

As mentioned before, the problem is finding the number of n-permutations on a multiset. From the multiplication principle of exponential generating functions it follows that the exponential generating function $F(x)$ is in form $H(x)G_1(x)\dots G_t(x)$, where $H(x)$ is the exponential generating function for label 0 and G_i is the exponential generating function for labels i.

Let us find the exponential generating functions for the labels. The functions $h_i(j)$ and $g_i(j)$ are 1 if the set of size j out of the corresponding label is possible and 0 otherwise. We can create a set of any size of label 0 (as we have infinite copies of label 0), so we get that the exponential generating function for label 0 is

$$H(x) = \frac{1}{0!} + \frac{1}{1!}x + \frac{1}{2!}x^2 + \ldots = \sum_{j=0}^{\infty} \frac{x^j}{j!} = e^x.$$

The equality $\sum_{j=0}^{\infty} \frac{x^j}{j!} = e^x$ is true for all $x \in \mathbb{R}$.

Function $g_i(j)$ is equal to 1 only for $1 \leq j \leq r$ because we have at most r copies of label i. It follows that the exponential generating function for label i is

$$G_i(x) = \frac{1}{1!}x + \frac{1}{2!}x^2 + \ldots + \frac{1}{r!}x^r.$$

Therefore the exponential generating function for the given multiset is

$$F(x) = H(x)G_1(x)\ldots G_t(x) = H(x)\left(G_1(x)\right)^t = e^x \cdot \left(\frac{1}{1!}x + \frac{1}{2!}x^2 + \ldots + \frac{1}{r!}x^r\right)^t. \tag{4}$$

Recall that to count how many labelling of the matrix there are, we would need to find $f(n)$.

Let us look at the problem from a different angle. Instead of assigning label 0 to all of the unused columns and counting labellings of the coulmns of matrix with $t+1$ labels, we can assume that we have t labels and that those labels take up all of the columns. The number of columns that the labels use up is from t to n. Let $T_{r,n,t}$ be the number of labellings of the matrix with t labels, such that each label is repeated at most r times and n being the length of the generator matrix $\mathbf{G}$. We get that the number of labellings of the matrix with t labels, such that each label is repeated at most r times is

$$T_{r,n,t} = \sum_{m=t}^{n} \binom{n}{m} m! C_{x^n}\left(\Pi_{i=1}^{r} G_i\right) = \sum_{m=t}^{n} \binom{n}{m} m! C_{x^n}\left(\left(\frac{1}{1!}x + \frac{1}{2!}x^2 + \ldots + \frac{1}{r!}x^r\right)^t\right).$$

Consider a special case $r = 2$:

$$\begin{aligned} & n! \cdot C_{x^n}\left(\left(x + \frac{x^2}{2!}\right)^t\right) = \\ & = n! 2^{-t} C_{x^{n-t}}\left((2+x)^t\right) = \\ & = n! 2^{-t} 2^{2t-n} \binom{t}{n-t} = \\ & = n! 2^{t-n} \binom{t}{n-t}. \end{aligned}$$

Therefore,

$$T_{2,n,t} = \sum_{m=t}^{n} \binom{n}{m} m! 2^{t-m} \binom{t}{m-t}. \tag{5}$$

We further expand the idea for $r = 3$:

$$\begin{aligned}
&n!C_{x^n}\left(\left(x+\frac{x^2}{2!}+\frac{x^3}{3!}\right)^t\right) = \\
&= n!6^{-t}C_{x^{n-t}}\left((6+3x+x^2)^t\right) \leq \\
&\leq n!6^{-t}C_{x^{n-t}}\left((6+2\cdot\sqrt{6}x+x^2)^t\right) = \\
&= n!6^{-t}C_{x^{n-t}}\left((x+\sqrt{6})^{2t}\right) = \\
&= n!6^{-t}\sqrt{6}^{3t-n}\binom{2t}{n-t} = \\
&= n!6^{\frac{3t-n}{2}-t}\binom{2t}{n-t} \\
&= n!6^{\frac{t-n}{2}}\binom{2t}{n-t}.
\end{aligned}$$

As a consequence,

$$T_{3,n,t} \leq \sum_{m=t}^{n}\binom{n}{m}m!6^{\frac{t-m}{2}}\binom{2t}{m-t}. \tag{6}$$

The idea can be used for larger r, which we will not cover in this work.

In addition, the exponential generating function allows us to give another proof of the result of Theorem 6. Assume that there is an unlimited number of labels i, where $0 \leq i \leq t$. Moreover, assume that label i might not be present in a labelling of the matrix, the way as it was done in [12] and that the requests are distinct. We still want to find $f(n)$ in $F(x) = H(x)G_1(x)\ldots G_t(x)$. In the setup of Theorem 6, we get that functions $H(x)$ and $G(x)$ are the same. Namely,

$$H(x) = G_i(x) = \frac{1}{0!}+\frac{1}{1!}x+\frac{1}{2!}x^2+\ldots = \sum_{j=0}^{\infty}\frac{x^j}{j!} = e^x.$$

Hence,

$$F(x) = e^x\cdot e^{x^t} = e^x\cdot e^{xt} = e^{x+xt} = e^{x(1+t)} = \sum_{j=0}^{\infty}\frac{1}{j!}x^j(1+t)^j.$$

Let us look at the term where $j = n$. The term is $\frac{1}{n!}x^n(1+t)^n$ and because of the definition of $F(x)$, we get $f(n) = (1+t)^n$. Then, $(1+t)^n \geq (2^k-1)^t)$.

Therefore,

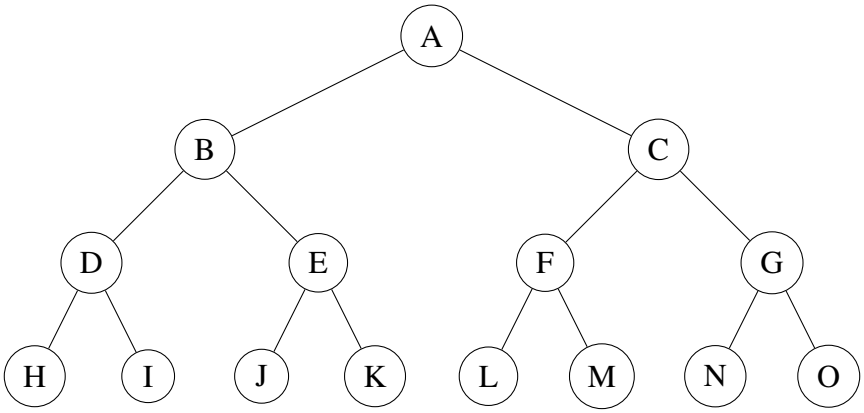
$$\lim_{k\to\infty}\frac{n}{k} \geq \frac{t}{\log(1+t)}.$$

We obtained an alternative proof for the lower bound in [12].

4.2 Approach 2

We may also represent the process of choosing columns for labels as a decision tree. For each request, there are r possibilities for the size of the recovery set, namely, $1, 2, \ldots, r$. This means that when reconstructing the request, we first need to "decide" the size of the recovery set and then we may count in how many ways we can reconstruct the request. In this subsection, we will focus on the case when $r = 2$.

Example 8. For $r = 2$ and $t = 3$, we have the following tree. The left child means that the request has a recovery set of size 1; the right child means that the request has a recovery set of size 2.



Lemma 8. For positive m, we have:

$$m + \binom{m}{2} = \binom{m+1}{2}. \tag{7}$$

Proof. We have that

$$\begin{aligned} & m + \binom{m}{2} = \\ & = m + \frac{m(m-1)}{2} = \\ & = \frac{m(m+1)}{2} = \\ & = \binom{m+1}{2}. \end{aligned}$$

□

Corollary 1. If x, y are integers with $x \geq y$, then

$$(x-y) + \binom{x-y}{2} = \binom{x-y+1}{2}$$

Proof. In Expression (7), take $m = x - y$. □

Lemma 9. For $r = 2$, there are at most

$$\frac{n^{2t}}{2^t}$$

possible labellings of the matrix for $t \geq 2$.

Proof. Assume there is a functional t-batch code of dimension k and length n that is represented by a $k \times n$ matrix $\mathbf{G}$. Given that $r = 2$, there are either n or $\binom{n}{2}$ possibilities to choose the columns for the first request, depending on the size of its recovery set, which means that in total there are $n + \binom{n}{2}$ possibilities. Let $F(n, t)$ be the number of labellings of the n columns of the generator matrix $\mathbf{G}$ with t labels (excluding label 0). Then the function F, which is non-decreasing in n, for the root is $F(n, t) = nF(n-1, t-1) + \binom{n}{2}F(n-2, t-1)$. This function can bounded from above by

$$nF(n-1,t-1) + \binom{n}{2}F(n-1,t-1) = \left(\frac{n(n+1)}{2}\right)F(n-1,t-1).$$

This means that we only need to calculate what happens if one column was chosen for the first request. We can continue the same pattern for the next $t-1$ request and we get:

$$(n+1)\frac{n^2}{2} \cdot \frac{(n-1)^2}{2} \cdot \ldots \cdot \frac{(n-t+1)^2}{2} \cdot (n-t).$$

The above expression can be bounded from above by $\frac{n^{2t}}{2^t}$ for $t \geq 2$. This follows from $(n-i+1) \leq n$ for $i = 1, \ldots, t-1$ and $(n+1)(n-t+1) \leq (n+1)(n-1) = n^2 - 1 < n$ for $t \geq 2$. □

Theorem 10. For any functional t-batch code with recovery sets of size at most 2, that is represented by a $k \times n$ matrix, we have

$$n \geq \sqrt{(2^k - 1) \cdot 2}.$$

Proof. We have the inequality $\frac{n^{2t}}{2^t} \geq (2^k - 1)^t$. We can take the t-th root and we are obtain $\frac{n^2}{2} \geq (2^k - 1)$. Then,

$$n^2 \geq (2^k - 1) \cdot 2.$$

Hence,

$$n \geq \sqrt{(2^k - 1) \cdot 2}.$$

□

4.3 Approach 3

The third approach is based on the exact counting of labellings there are of a matrix, where label i, $1 \leq i \leq t$, is used at most r times. Then we estimate the exact number of such labellings from above with the help of the multinomial theorem.

Theorem 11. For a functional t-batch code with recovery sets of size at most 2, there are

$$\sum_{j=0}^{t} \binom{n-j}{2(t-j)} \frac{(2(t-j))!}{2^{t-j}} \binom{t}{j} \binom{n}{j} j! \tag{8}$$

different labellings of a matrix that represents the batch code.

Proof. Assume there is a functional t-batch code of dimension k and length n that is represented by a $k \times n$ generator matrix $\mathbf{G}$. Let us assign a label for every column of $\mathbf{G}$ for any recovery process of request $\mathbf{v} = (\mathbf{v}_1, \ldots, \mathbf{v}_t)$. The label assigned to each column is i, where $0 \leq i \leq t$. Label i denotes the recovery set for the request $\mathbf{v}_i$ that the column belongs to. Due to the restriction of size of the recovery set, each label i, $1 \leq i \leq n$ can appear in the matrix at most 2 times. Label 0 does not have any restrictions, as it indicates that the column is not used in any recovery sets. The labelling of $\mathbf{G}$ must be different for any two different ordered t-tuples of requests $(\mathbf{v}_1, \ldots, \mathbf{v}_t)$ and $(\mathbf{u}_1, \ldots, \mathbf{u}_t)$.

Let $0 \leq j \leq t$ denote the number of vectors, whose recovery set is of size 1. There are $\binom{t}{j}$ possibilities to choose labels that appear only once in the matrix and there are $\binom{n}{j} j!$ possibilities to choose a place for those labels in the matrix. Note, that ordering is important when assigning labels to columns.

There are $t - j$ labels left and each label appears twice in the matrix. Those labels are assigned to a subset of a set of $n - j$ columns that are left. There are $\binom{n-j}{2(t-j)} \frac{(2(t-j))!}{2!^{t-j}}$ ways to assign those labels.

By combining everything, we get that there are

$$\sum_{j=0}^{t} \binom{n-j}{2(t-j)} \frac{(2(t-j))!}{2^{t-j}} \binom{t}{j} \binom{n}{j} j!$$

different labellings for $\mathbf{G}$, where label i is used at most 2 times. □

Theorem 12. For a functional t-batch code with recovery sets of size at most 3, there are

$$\sum_{l=0}^{t-j} \sum_{j=0}^{t} \binom{t}{j} \binom{n}{j} j! \binom{t-j}{l} \binom{n-j}{2l} \frac{(2l)!}{2^l} \binom{n-j-2l}{3(t-j-l)} \frac{(3(t-j-l))!}{3!^{t-j-l}} \tag{9}$$

different labellings of the columns of a matrix that represents the batch code.

The proof of this theorem is similar to the proof of Theorem 11.

Theorem 13. If $\sum_{i=1}^{r} j_i \leq t$, then for a functional t-batch code with recovery sets of size at most r, there are

$$\sum_{j_{r-1}=0}^{t-\sum_{i=1}^{r-2} j_i} \dots \sum_{j_1=0}^{t} \binom{t}{j_1, \dots, j_r} \binom{n}{j_1, 2 \cdot j_2, \dots, r \cdot j_r, x} \prod_{i=1}^{r} \frac{(i \cdot j_i)!}{i!^{j_i}} \tag{10}$$

different labellings of the columns of a matrix, where $j_r = t - \sum_{i=1}^{r-1} j_i$ and $x = n - \sum_{i=1}^{r} i \cdot j_i$.

Proof. Assume there is a functional t-batch code of dimension k and length n that is represented by a $k \times n$ matrix $\mathbf{G}$. Let us assign a label for every column of $\mathbf{G}$ for any recovery process of request $\mathbf{v} = (\mathbf{v}_1, \dots, \mathbf{v}_t)$. The label assigned to each column is either 0 or some i, where $0 \leq i \leq t$ and it denotes the recovery set for $\mathbf{v}_i$ that the column belongs to. Due to the restriction of the size of the recovery set, each label can appear in the matrix at most r times. Label 0 does not have any restrictions, as it indicates that the column is not used in any recovery sets.

Let $0 \leq j_i \leq t$ denote the number of recovery sets of size i. For recovery sets of sizes from 1 to $r-1$, there are $\binom{t-\sum_{l=1}^{i-1} j_l}{j_i}$ possibilities to choose labels that appear exactly i times in the matrix and there are $\binom{n-\sum_{l=1}^{i-1} l \cdot j_l}{i \cdot j_i} \frac{(i \cdot j_i)!}{i!^{j_i}}$ possibilities to choose a place for those labels in the matrix. Note, that ordering is important when assigning labels to columns. In addition, we have to take into consideration all of the columns that already have a label.

There are $t - \sum_{i=1}^{r-1} j_i$ labels left and each label appears r times in the matrix. Those labels are assigned to $n - \sum_{i=1}^{r-1} i \cdot j_i$ columns that are left. There are $\binom{n-\sum_{i=1}^{r-1} i \cdot j_i}{r \cdot (t - \sum_{i=1}^{r-1} j_i)} \frac{(r \cdot (t - \sum_{i=1}^{r-1} j_i))!}{r^{\sum_{i=1}^{r-1} j_i}}$ ways to assign those labels.

By combining everything, we get that there are

$$\sum_{j_{r-1}=0}^{t-\sum_{i=1}^{r-2} j_i} \dots \sum_{j_1=0}^{k} \left(\binom{n - \sum_{i=1}^{r-1} i \cdot j_i}{r \cdot (t - \sum_{i=1}^{r-1} j_i)} \frac{(r \cdot (t - \sum_{i=1}^{r-1} j_i))!}{r^{\sum_{i=1}^{r-1} j_i}} \cdot \right.$$
$$\left. \cdot \prod_{i=1}^{r-1} \binom{t - \sum_{l=1}^{i-1} j_l}{j_i} \binom{n - \sum_{l=1}^{i-1} l \cdot j_l}{i \cdot j_i} \frac{(i \cdot j_i)!}{i!^{j_i}} \right)$$

different possibilities for recovering t vectors with recovery sets of size at most r. This expression is equal to Expression 10. □

Lemma 14. Based on Theorem 2, the following is true:

$$(\sigma_1 x_1 + \sigma_2 x_2 + \ldots + \sigma_m x_m)^n = \sum_{\substack{k_1 + \ldots + k_m = n \\ k_1, \ldots, k_m \geq 0}} \binom{n}{k_1, \dots, k_m} x_1^{k_1} x_2^{k_2} \dots x_m^{k_m} \sigma_1^{k_1} \dots \sigma_m^{k_m}.$$

Lemma 15. An upper bound for the Expression (10) is

$$\left(n^1 + \ldots + n^r \cdot \frac{1}{r!}\right)^t \tag{11}$$

Proof. The Expression (10) is equal to expression

$$\sum_{j_{r-1}=0}^{t-\sum_{i=1}^{r-2} j_i} \cdots \sum_{j_1=0}^{t} \binom{t}{j_1, \ldots, j_r} \frac{n!}{j_1!(2 \cdot j_2)! \ldots (r \cdot j_r)! x!} \frac{j_1!(2 \cdot j_2)! \ldots (r \cdot j_r)!}{1!^{j_1} 2!^{j_2} \ldots r!^{j_r}}.$$

This expression can be simplified to

$$\sum_{j_{r-1}=0}^{t-\sum_{i=1}^{r-2} j_i} \cdots \sum_{j_1=0}^{t} \binom{t}{j_1, \ldots, j_r} \frac{n!}{x!} \frac{1}{1!^{j_1}} \frac{1}{2!^{j_2}} \cdots \frac{1}{r!^{j_r}}.$$

Because $\frac{1}{x!} = \frac{\left(n+1-\sum_{i=1}^{r} i \cdot j_i\right) \ldots n}{n!}$, the previous expression is equal to

$$\sum_{j_{r-1}=0}^{t-\sum_{i=1}^{r-2} j_i} \cdots \sum_{j_1=0}^{t} \binom{t}{j_1, \ldots, j_r} \frac{n! n(n-1) \cdot (x+1)}{n!} \frac{1}{1!^{j_1}} \frac{1}{2!^{j_2}} \cdots \frac{1}{r!^{j_r}}.$$

Since x is close to n due to fixed and small r, then the above equation can be bounded from above by

$$\sum_{j_{r-1}=0}^{t-\sum_{i=1}^{r-2} j_i} \cdots \sum_{j_1=0}^{t} \binom{t}{j_1, \ldots, j_r} n^{n-x} \frac{1}{1!^{j_1}} \frac{1}{2!^{j_2}} \cdots \frac{1}{r!^{j_r}}.$$

Because $n^{n-x} = n^{\sum_{i=1}^{r} i \cdot j_i} = (n^1)^{j_1} \ldots (n^r)^{j_r}$, we can use Lemma 14 substituting $\sigma_i = n^i$ and we get that the previous expression is equal to

$$\left(n^1 + \ldots + n^r \cdot \frac{1}{r!}\right)^t.$$

□

Theorem 16. For any functional t-batch code with recovery sets of size at most r, that is represented by a $k \times n$ matrix, we have

$$n \geq \ln\left(2^k - 1\right).$$

Proof. The expression in Lemma 15 can be upper-bounded using Taylor's series of the exponential function e^x, which is $\sum_{i=0}^{\infty} \frac{x^i}{i!}$. We obtain the bound $e^{nt} - 1 < e^{nt}$.

Comparing the obtained bound to the number of ways to choose a sequence of t vectors, we obtain

$$e^{nt} \geq (2^k - 1)^t.$$

We can take the t-th root and the natural logarithm:

$$\begin{aligned} e^n &\geq (2^k - 1) \\ \ln(e^n) &\geq \ln(2^k - 1) \\ n &\geq \ln(2^k - 1). \end{aligned}$$

□

However, the bound in Theorem 16 does not depend on the value of r or t. By approximating the expression in Lemma 15 by Taylor's series of the exponential function e^x, we allow r to go to infinity, which is not the case we are interested in.

We further simplify the upper bound in Lemma 15 with the help of the following lemma:

Lemma 17. The expression in Lemma 15 is smaller or equal to

$$\left(\frac{r \cdot n^r}{r!}\right)^t$$

for $r < n$.

Proof. We aim to find the biggest term of the sum $\left(\frac{n^1}{1!} + \ldots + \frac{n^r}{r!}\right)$. We compare terms i and $i+1$: $\frac{n^{i+1}}{(i+1)!} = \frac{n}{i+1} \cdot \frac{n^i}{i!} > \frac{n^i}{i!}$. Thus, the term $i+1$ is larger than the term i for $1 \leq i \leq r-1$, and therefore the largest term in the sum is the last term. We obtain that Expression (11) is smaller of equal to

$$\left(\frac{r \cdot n^r}{r!}\right)^t.$$

□

Theorem 18. For any functional t-batch code with recovery sets of size at most r, that is represented by a $k \times n$ matrix, we have

$$n \geq \sqrt[r]{(r-1)!(2^k - 1)}.$$

Proof. There are $(2^k - 1)^t$ ways to choose t requests. We compare it to the expression in Lemma 17. We have

$$\begin{aligned}\left(\frac{r \cdot n^r}{r!}\right)^t &\geq (2^k - 1)^t \\ \frac{r \cdot n^r}{r!} &\geq 2^k - 1 \\ n^r &\geq \frac{(2^k - 1) \cdot r!}{r} \\ n &\geq \sqrt[r]{(r-1)!(2^k - 1)}.\end{aligned}$$

□

4.4 Numerical results

In this section, we compare the results for a fixed r and t. Namely, we compare the bound from Theorem 6 within Section 3, proposed in [12] without a restriction on r, and the bound we presented in Theorem 18. For $r = 2$, we also include the bound from Theorem 10.

For $r = 2$ and $t = 5$, we get Table 3.

The lower bound on n			
k	Theorem 6	Theorem 10	Theorem 18
10	64	45	32
15	96	256	181
20	129	1448	1024
25	161	8192	5793
30	193	46341	32768
35	225	262144	185364
40	257	1482910	1048576
45	289	8388608	5931642
50	321	47453133	33554432

Table 3. Comparison of the lower bound on n for $r = 2$ and $t = 5$.

For $r = 3$ and $t = 5$, we have Table 4.

The bounds derived with exponential generating functions are unexplicit, meaning that we cannot calculate lower bound on n using just parameters r, k and t. This is why we calculate what is the smallest n for $r = 2$, so that Expression (5) is at least $(2^k - 1)^t$ for different k and t. We have Table 5.

We do the same for Expression (6) for $r = 3$ and obtain Table 6.

The lower bound on n		
k	Theorem 6	Theorem 18
10	64	13
15	96	40
20	129	128
25	161	406
30	193	1290
35	225	4096
40	257	13004
45	289	41285
50	321	131072

Table 4. Comparison of the lower bound for $r = 3$ and $t = 5$.

The smallest n				
k	$t = 5$	$t = 10$	$t = 15$	$t = 20$
5	12	17	22	27
10	49	54	60	65
15	260	265	270	275
20	1452	1457	1462	1467

Table 5. Smallest n so that inequality $\sum_{m=t}^{n} \binom{n}{m} m! 2^{t-m} \binom{t}{m-t} \geq (2^k - 1)^t$ holds. The significance of this table is that we would need at least n columns so that the number of all possible labellings would be bigger than the number of ways to construct requests.

The smallest n				
k	$t = 5$	$t = 10$	$t = 15$	$t = 20$
5	10	16	21	26
10	24	32	39	47
15	64	72	80	88
20	190	198	206	214

Table 6. Smallest n so that inequality $\sum_{m=t}^{n} \binom{n}{m} m! 6^{\frac{t-m}{2}} \binom{2t}{m-t} \geq (2^k - 1)^t$ holds. The significance of this table is that we would need at least n columns so that the number of all possible labellings would be bigger than the number of ways to construct requests.

5 Conclusion

In this thesis, we studied batch codes with small recovery sets. In particular, we derived lower bounds on the length n of a functional batch code. We used a similar approach that was used in the proof of Theorem 23 in [12] to obtain a lower bound on n for the case of recovery sets of unrestricted size. Namely, we compared the number of sequences of requests of length t with the number of different labellings of the columns of a generator matrix $\mathbf{G}$ of a functional batch code with labels $0, 1 \ldots, t$, where label i corresponds to the ith request in the sequence (with label 0 indicating that the column is not a part of any recovery set).

In our first approach, we considered the exponential generating functions to derive a lower bound on n. Here, we found lower bounds on n for the cases $r = 2$ and $r = 3$ that improve the lower bound from [12]. In addition, we gave an alternative proof to Theorem 23 in [12] using exponential generating functions.

In our second approach, we use decision trees to model the process of column labelling. In this way, for $r = 2$ we obtained the lower bound $n \geq \sqrt{(2^k - 1) \cdot 2}$.

In the third and final approach, we expressed the exact number all possible labellings of matrix $\mathbf{G}$, which was then estimated form above using the multinomial theorem. This allowed us to find a lower bound on n, which is $n \geq \sqrt[r]{(r-1)!(2^k - 1)}$ for general small r. Using a similar approach but with a better approximation at the crucial part of the proof, is possible to improve this bound to $n \geq \frac{t-1}{2} + \sqrt[r]{(r-1)!(2^k - 1)}$.

In Table 3, it can be seen that there is a gap between the lower bound obtained from the tree approach and the lower bound obtained in the third approach for $r = 2$. The bound obtained from the tree approach gives slightly better results. When comparing n for $k = 10$, $r = 2$ and $k = 5$ with the tree bound for n for the same parameters in Table 3, we see that the lower bound on n obtained by the third approach is 49 while the tree bound gives a lower bound of 45. While the bound from the third approach might not give the best results for $r = 2$, it applies for a general small r, unlike the bound from the tree approach, which is true only for $r = 2$.

Future work

- In this work, the bounds are derived under the assumption that the size of the restricted recovery set is small. It would be interesting to derive bounds for larger r.

- We derived two inequalities involving n in Expressions (5) and (6). Explicit bounds on n from these inequalities have not been derived in this thesis. In addition, those bounds are for $r = 2$ and $r = 3$, respectively. Finding bounds for larger r is still an open question.

References

[1] David S. Dummit and Richard M. Foote. *Abstract Algebra*. third edition, 2004. URL: https://rksmvv.ac.in/wp-content/uploads/2021/04/David_S_Dummit_Richard_M_Foote_Abstract_Algeb_230928_225848.pdf.

[2] Arman Fazeli, Alexander Vardy, and Eitan Yaakobi. Codes for distributed pir with low storage overhead. In *2015 IEEE International Symposium on Information Theory (ISIT)*, pages 2852–2856, 2015. doi:10.1109/ISIT.2015.7282977.

[3] Mark Haiman. The exponential generating function, 2010. Accessed: 2025-04-21. URL: https://math.berkeley.edu/~mhaiman/math172-spring10/exponential.pdf.

[4] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Batch codes and their applications. *Conference Proceedings of the Annual ACM Symposium on Theory of Computing*, 2004. doi:10.1145/1007352.1007396.

[5] Jüri Lember and Ago-Erik Riet. Equal requests are asymptotically hardest for data recovery. *2024 IEEE International Symposium on Information Theory (ISIT)*, page 3678–3683, 2024.

[6] Helger Lipmaa and Vitaly Skachek. Linear batch codes. In *Coding Theory and Applications*, pages 245–253. Springer International Publishing, 2015.

[7] Jiri Matoušek and Jaroslav Nešetril. *Invitation to Discrete Mathematics*. Oxford University Press Inc., second edition, 2008.

[8] Ron M. Roth. *Introduction to Coding Theory*. Cambridge University Press, 2006.

[9] Jeff Suzuki. Introduction to exponential generating functions, 2024. Accessed: 2025-05-13. URL: https://www.youtube.com/watch?v=CwKjTRxEQ7A.

[10] Zhiying Wang, Omer Shaked, Yuval Cassuto, and Jehoshua Bruck. Codes for network switches. In *2013 IEEE International Symposium on Information Theory*, pages 1057–1061, 2013. doi:10.1109/ISIT.2013.6620388.

[11] Hui Zhang and Vitaly Skachek. Bounds for batch codes with restricted query size. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 1192–1196, 2016. doi:10.1109/ISIT.2016.7541487.

[12] Yiwei Zhang, Tuvi Etzion, and Eitan Yaakobi. Bounds on the length of functional PIR and batch codes. *IEEE Transactions on Information Theory*, 66(8):4917–4934, 2020. doi:10.1109/TIT.2020.2977631.

Appendix

I. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Kristiina Oksner**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Study of batch codes with small recovery sets,

   supervised by Vitaly Skachek, Ago-Erik Riet and Henk D. L. Hollmann.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Kristiina Oksner
15/05/2025