On the Minimum Length of Functional Batch Codes with Small Recovery Sets

Kristiina Oksner

26.04.2026

Online shop

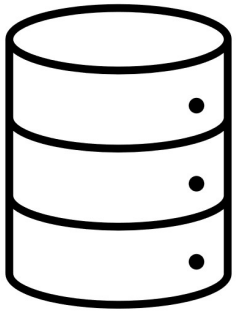

HAVE A NICE LIFE
GOOD LUCK
new
item
Get special discount
15% off for min.
purchase 100pcs
SHOP NOW

Online shop

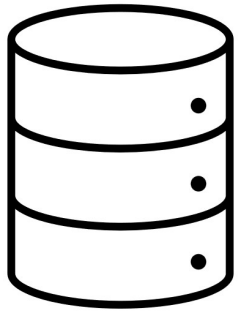

HAVE A NICE LIFE
GOOD LUCK
new
item
ial discount
off for min.
e 100pcs
SHOP NOW

The problem

- Two users want to retrieve bit x_1 in parallel
- There are two servers:



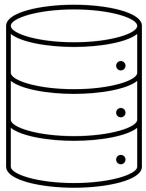
x_1



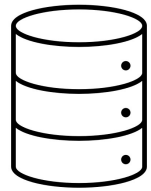
x_2

Simple solution

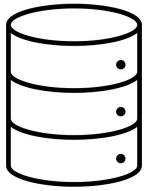
- Add two servers:



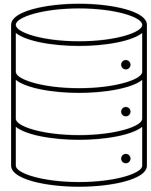
x_1



x_2



x_1

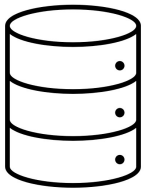


x_2

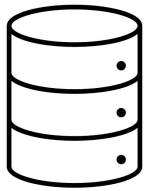
- 100% storage increase

Batch code solution

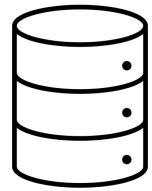
- Add one server:



x_1



x_2



$x_1 + x_2$

- $\begin{cases} x_1 = x_1 \\ x_1 = (x_1 + x_2) - x_2 \end{cases}$
- 50% storage increase

Codes

Code

An $[n, k]$ *linear code* over a finite field $\mathbb{F}$ is a linear subspace C of dimension k of $\mathbb{F}^n$, where n is called the length of the code and k is the dimension of the code. *Codewords* are the elements of the code.

Generator matrix

A *generator matrix* $\mathbf{G}$ of a linear $[n, k]$ code over $\mathbb{F}$ is a $k \times n$ matrix, where rows form a basis of the code.

Each encoded vector $\mathbf{y} \in \mathbb{F}^n$ can be written as $\mathbf{y} = \mathbf{xG}$, where $\mathbf{G}$ is a generator matrix and $\mathbf{x} \in \mathbb{F}^k$.

Batch codes

Primitive multiset linear batch code

An $[n, k]$ *primitive multiset linear batch code* is an $[n, k]$ code which supports t queries in the following sense:

- For any multiset $i_1, \ldots, i_t \in [k]$ of requests there is a partition of the symbols into t disjoint subsets $S_1, \ldots, S_t \subseteq [n]$ such that each symbol $x_{i_j}, j \in [t]$, can be recovered by reading only one symbol in S_j.

Batch codes

Primitive multiset linear batch code

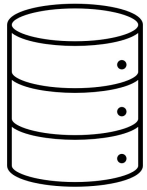
An $[n, k]$ *primitive multiset linear batch code* is an $[n, k]$ code which supports t queries in the following sense:

- For any multiset $i_1, \ldots, i_t \in [k]$ of requests there is a partition of the symbols into t disjoint subsets $S_1, \ldots, S_t \subseteq [n]$ such that each symbol $x_{i_j}, j \in [t]$, can be recovered by reading only one symbol in S_j.

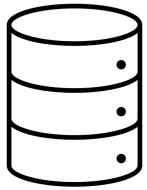
We may omit 'primitive multiset linear' codes and refer to them as batch codes or t-batch codes.

Recall: Batch code solution

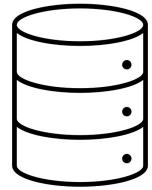
- There are three servers:



x_1



x_2



$x_1 + x_2$

- Or encoding of the vector (x_1, x_2) into $(x_1, x_2, x_1 + x_2)$ by multiplying it by the matrix

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

Recovery sets

Bit x_i is represented by the vector $\mathbf{r} = \begin{pmatrix} 0 & 0 & \dots & 1 & \dots & 0 \end{pmatrix}$, where the ith symbol of the vector is 1 and other symbols are 0.

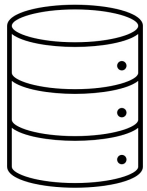
Recovery set for a vector

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

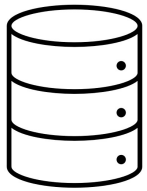
$$\boldsymbol{r} \in \langle \mathbf{g}_j \mid j \in R \rangle. \tag{1}$$

Recall: Batch code solution

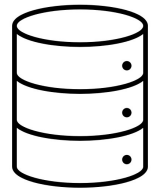
- There are three servers:



x_1



x_2



$x_1 + x_2$

- $\begin{cases} x_1 = x_1 \\ x_1 = (x_1 + x_2) - x_2 \end{cases}$

Example of batch codes

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 $\mathbf{x} = (x_1, x_2, x_3, x_4)$, $\mathbf{y} = \mathbf{xG}$.
Assume that we want to retrieve the values of (x_1, x_1, x_2, x_2):

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

This is a 4-batch code.

Example of batch codes

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 $\mathbf{x} = (x_1, x_2, x_3, x_4)$, $\mathbf{y} = \mathbf{xG}$.
Assume that we want to retrieve the values of (x_1, x_1, x_2, x_2):

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

This is a 4-batch code.

Functional batch codes

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 information bits. So the request is represented by a nonzero vector of length k.

Linear combination $\sum_{i=1}^{k} \lambda_i x_i$ of the k information bits is represented by the vector $\mathbf{r} = \begin{pmatrix} \lambda_1 & \lambda_2 & \dots & \lambda_k \end{pmatrix}$.

Lower bound on n

Theorem (Zhang, Etzion and Yaakobi 2020[1])

Suppose that the $k \times n$ binary matrix $\mathbf{G}$ is a generator matrix of a t-functional batch code. Then

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

[1] 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.

Proof

- There is a matrix $k \times n$
- Each request has its own label + label 0
- We want to assign a label to each column
- There are $(t+1)^n$ ways to do that
- There are $(2^k - 1)^t$ ways to choose request sequence
- Number of ways to assign labels $\geq$ number of ways to choose requests
- $(t+1)^n \geq (2^k - 1)^t \rightarrow \lim_{k \to \infty} \frac{n}{k} \geq \frac{t}{\log(t+1)}$

Batch codes with restricted query size

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.

Batch codes with restricted query size

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.

- Faster recovery!

Recall: Example of batch codes

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 $\mathbf{x} = (x_1, x_2, x_3, x_4)$, $\mathbf{y} = \mathbf{xG}$.
Assume that we want to retrieve the values of (x_1, x_1, x_2, x_2):

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

This is a 4-batch code.

Example: batch codes with restricted query size

- Take $r = 2$ and in this case x_2 cannot be recovered as a sum of vectors $\{y_4, y_6, y_7, y_9\}$
- We can only serve two requests if the recovery sets have a size of at most 2.

Recall: Lower bound on n

Theorem (Zhang, Etzion and Yaakobi 2020)

Suppose that the $k \times n$ binary matrix $\mathbf{G}$ is a generator matrix of a t-functional batch code. Then

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

Recall: Lower bound on n

Theorem (Zhang, Etzion and Yaakobi 2020)

Suppose that the $k \times n$ binary matrix $\mathbf{G}$ is a generator matrix of a t-functional batch code. Then

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

What happens if all recovery sets have size at most r? For example, $r = 2, 3, \ldots$?

Approach 1

- Exponential generating function $F(x) = \sum_n f(n)x^n/n!$

Approach 1

- Exponential generating function $F(x) = \sum_n f(n)x^n/n!$
- Example:
  - Multiset $\{a, a, a, a\}$

Approach 1

- Exponential generating function $F(x) = \sum_n f(n)x^n/n!$
- Example:
  - Multiset $\{a, a, a, a\}$
  - The 0-permutation:

Approach 1

- Exponential generating function $F(x) = \sum_n f(n)x^n/n!$
- Example:
  - Multiset $\{a, a, a, a\}$
  - The 0-permutation:
  - The 1-permutation: a

Approach 1

- Exponential generating function $F(x) = \sum_n f(n)x^n/n!$
- Example:
  - Multiset $\{a, a, a, a\}$
  - The 0-permutation:
  - The 1-permutation: a
  - The 2-permutation: aa

Approach 1

- Exponential generating function $F(x) = \sum_n f(n)x^n/n!$
- Example:
  - Multiset $\{a, a, a, a\}$
  - The 0-permutation:
  - The 1-permutation: a
  - The 2-permutation: aa
  - The 3-permutation: aaa

Approach 1

- Exponential generating function $F(x) = \sum_n f(n)x^n/n!$
- Example:
  - Multiset $\{a, a, a, a\}$
  - The 0-permutation:
  - The 1-permutation: a
  - The 2-permutation: aa
  - The 3-permutation: aaa
  - The 4-permutation: $aaaa$

Approach 1

- Exponential generating function $F(x) = \sum_n f(n)x^n/n!$
- Example:
  - Multiset $\{a, a, a, a\}$
  - The 0-permutation:
  - The 1-permutation: a
  - The 2-permutation: aa
  - The 3-permutation: aaa
  - The 4-permutation: $aaaa$
  - $F(x) = \frac{1}{0!} + \frac{1}{1!}x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \frac{1}{4!}x^4$

Approach 1

- Multiset $\{\underbrace{0,\ldots,0}_{\infty \text{ times}},\underbrace{l_1,\ldots,l_1}_{r \text{ times}},\ldots,\underbrace{l_t,\ldots,l_t}_{r \text{ times}}\}$
- $F(x)$ is in form $H(x)G_1(x)\ldots G_t(x)$,where $H(x)$ is the exponential generating function for label 0 and G_i is the exponential generating function for labels l_i
- $F(x) = e^x \cdot \left(\frac{1}{1!}x + \frac{1}{2!}x^2 + \ldots + \frac{1}{r!}x^r\right)^t$
- For $r = 2$, we get $\sum_{m=t}^{n} \binom{n}{m} m! 2^{t-m} \binom{t}{m-t}$

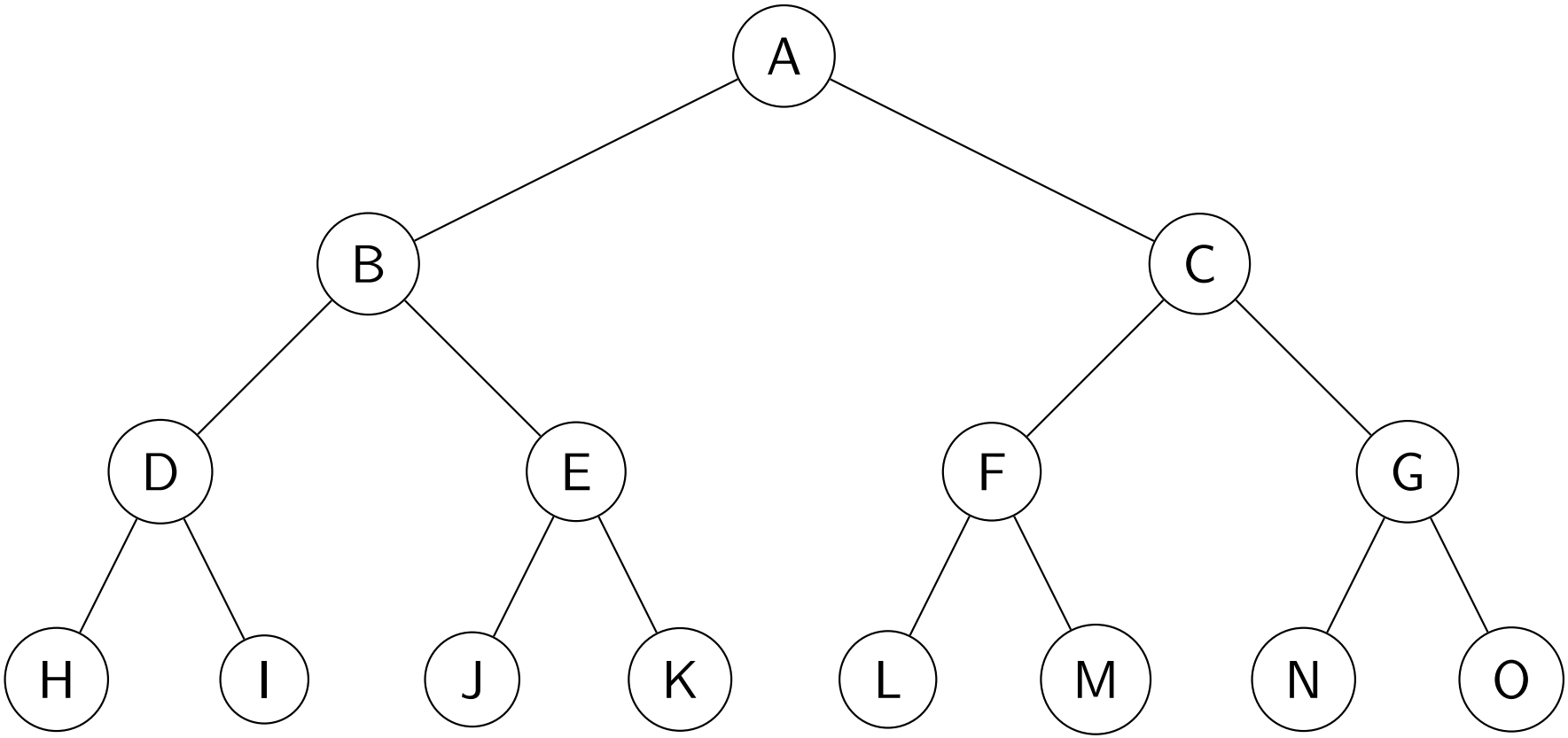
Numerical results 1

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

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.

Approach 2

- Fix $r = 2$
- For $r = 2$ and $t = 3$, we have the following tree:



Approach 2

- 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)
- $nF(n-1,t-1) + \binom{n}{2}F(n-2,t-1) \leq \left(\frac{n(n+1)}{2}\right)F(n-1,t-1)$
- $(n+1)\frac{n^2}{2} \cdot \frac{(n-1)^2}{2} \cdot \ldots \cdot \frac{(n-t+1)^2}{2} \cdot (n-t) \leq \frac{n^{2t}}{2^t}$ for $t \geq 2$

Theorem

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

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

Approach 2

Lemma

Assume that is an $[n, k, t, r]$ functional batch code over $\mathbb{F}$, for $t \geq 1$ and $r \geq 3$. Let $2r \leq n$. Then, there are at most

$$\left(\frac{n^r}{(r-1)!}\right)^t$$

possible labellings of the indices in $[n]$.

Theorem

Fix some integers $t \geq 1$ and $r \geq 3$. For any functional linear $[n, k, t, r]$ batch code C, we have

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

Approach 3

Theorem

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

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

Numerical results 2

The lower bound on n			
k	ZEY bound	Approach 2	Approach 3
10	20	46	32
15	30	256	182
20	39	1449	1024
25	49	8192	5793
30	59	46341	32768
35	68	262144	185364
40	78	1482911	1048576
45	88	8388608	5931642
50	97	47453133	33554432

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

Thank you for listening!