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

Kristiina Oksner
Joint work with Henk D.L. Hollmann, Ago-Erik Riet, Vitaly Skachek

08.06.2026

Online shop

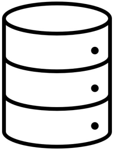
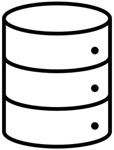
HAVE A NICE LIFE
GOOD LUCK
new item
Get special discount 15% off for min. purchase 100pcs
SHOP NOW

Online shop

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

The problem

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

x_1 x_2

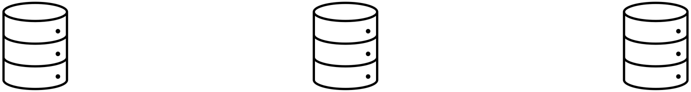
Simple solution

- Add two servers:

- 100% storage increase

Batch code solution

- Add one server:



x_1 $\quad$ x_2 $\quad$ $x_1 + x_2$

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

Generator matrix

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

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

Recovery sets

Bit x_i is represented by the i-th unit vector $e_i = \begin{pmatrix} 0 & 0 & \dots & 1 & \dots & 0 \end{pmatrix}$, where the i-th symbol of the vector is 1 and other symbols are 0.

Recovery set for a vector

Let $\boldsymbol{G} = (\mathbf{g}^{(1)} \mid \mathbf{g}^{(2)} \mid \cdots \mid \mathbf{g}^{(n)})$ be a $k \times n$ matrix over $\mathbb{F}_2$. A *recovery set* for a (nonzero) vector $\mathbf{v} \in \mathbb{F}_2^k \setminus \{\mathbf{0}\}$ is a subset $R \subseteq [n]$ of column indices such that $\mathbf{v}$ is contained in the span $\langle \mathbf{g}^{(j)} \rangle_{j \in R}$ of the columns of $\boldsymbol{G}$ with indices in R. We say R is a *minimal recovery set* for $\mathbf{v}$ if R does not contain a proper subset that is also a recovery set for $\mathbf{v}$.

Recovery sets

Bit x_i is represented by the i-th unit vector $e_i = \begin{pmatrix} 0 & 0 & \dots & 1 & \dots & 0 \end{pmatrix}$, where the i-th symbol of the vector is 1 and other symbols are 0.

Recovery set for a vector

Let $\boldsymbol{G} = (\mathbf{g}^{(1)} \mid \mathbf{g}^{(2)} \mid \cdots \mid \mathbf{g}^{(n)})$ be a $k \times n$ matrix over $\mathbb{F}_2$. A *recovery set* for a (nonzero) vector $\mathbf{v} \in \mathbb{F}_2^k \setminus \{\mathbf{0}\}$ is a subset $R \subseteq [n]$ of column indices such that $\mathbf{v}$ is contained in the span $\langle \mathbf{g}^{(j)} \rangle_{j \in R}$ of the columns of $\boldsymbol{G}$ with indices in R. We say R is a *minimal recovery set* for $\mathbf{v}$ if R does not contain a proper subset that is also a recovery set for $\mathbf{v}$.

Theorem 1

It is always possible to recover a linear combination $v_1 x_1 + \cdots + v_k x_k$ of the data $\mathbf{x}$ from the codeword symbols c_r for $r \in R$ of the corresponding codeword $\mathbf{c} = \boldsymbol{G}\mathbf{x}$ if and only if the vector $\mathbf{v} = (v_1, \dots, v_k)$ is contained in the linear span $\langle \mathbf{g}^{(r)} \mid r \in R \rangle$ of the columns in $\boldsymbol{G}$ in the positions of R.

Batch codes

Batch code

Let $\mathcal{C}$ be an $[n, k, t]$ (functional) batch code. We say that $\mathcal{C}$ has *locality* $r \in \mathbb{N}^+$, if for any batch $(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \cdots, \boldsymbol{\alpha}_t)$ there exist mutually disjoint recovery sets $R_1, R_2, \cdots, R_t$ such that $|R_\ell| \le r$ for all ℓ. Such a code will also be denoted as an $[n, k, t, r]$ (functional) batch code.

Batch codes

Batch code

Let $\mathcal{C}$ be an $[n, k, t]$ (functional) batch code. We say that $\mathcal{C}$ has *locality* $r \in \mathbb{N}^+$, if for any batch $(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \cdots, \boldsymbol{\alpha}_t)$ there exist mutually disjoint recovery sets $R_1, R_2, \cdots, R_t$ such that $|R_\ell| \leq r$ for all ℓ. Such a code will also be denoted as an $[n, k, t, r]$ (functional) batch code.

We will omit 'linear' codes and refer to them as following:

- t-PIR code: up to t requests for the same data symbol;
- t-batch code: up to t requests for (not necessarily distinct) data symbols;
- t-functional batch code: up to t requests for (not necessarily distinct) linear combinations of data symbols.

Recall: Batch code solution

- There are three servers:

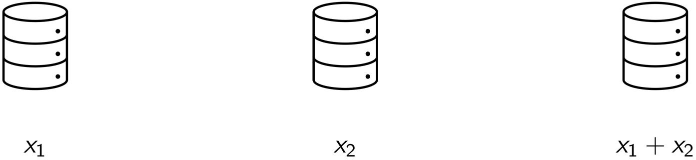
x_1 $\qquad$ x_2 $\qquad$ $x_1 + x_2$

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

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

Recall: Batch code solution

- There are three servers:



- $\begin{cases} x_1 = x_1 \\ x_1 = (x_1 + x_2) - x_2 \end{cases}$
- This is an example of a $[n{=}3, k{=}2, t{=}2, r{=}2]$ binary functional 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 $\mathbf{x} = (x_1, x_2, x_3, x_4)$, $\mathbf{y} = \mathbf{x}\mathbf{G}$.
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 with unrestricted recovery size.

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 $\mathbf{x} = (x_1, x_2, x_3, x_4)$, $\mathbf{y} = \mathbf{x}\mathbf{G}$.
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 with unrestricted recovery size.

Example: batch codes with restricted recovery size

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

Falling factorial

Falling factorial

For $n \in \mathbb{N}$ and $m \in \mathbb{Z}$ with $0 \leq m \leq n$, we define $(n)_m := n(n-1)\cdots(n-m+1)$; in particular, $(n)_0 = 1$ and $(n)_n = n!$.

Numbers $(n)_m$ are often called *falling factorials*. We use the following simple and well-known result.

Lemma

Let $m, n \in \mathbb{N}^+$, $n \geq m$. It holds: $(n)_m \leq (n-(m-1)/2)^m \leq n^m$.

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

$$\frac{n}{k} = \frac{t}{\log(t+1)} + o(1) \text{ for } k \to \infty.$$

[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 (not used in any request)
- 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
- So $\frac{t}{\log(t+1)} = \frac{n}{\log(2^k-1)} = \frac{n}{(k+\log(1+1/2^k)}$ $\approx \frac{n}{k}$ for large k

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

$$\frac{n}{k} = \frac{t}{\log(t+1)} + o(1) \text{ for } k \to \infty.$$

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

Lower bounds on the minimal length

- Number of ways to assign limited number of labels and label 0 $\geq$ number of ways to choose requests
- Recall that label 0 stands for a position that is not used in any recovery set
- Let $\theta_{t,r}(n)$ denote the number of distinct labellings of the numbers in $[n]$ with labels from $\{0, 1, \ldots, t\}$, where each nonzero label is used at least once and at most r times
- Obviously,

$$\theta_{t,r}(n) = \sum_{\substack{i_0 \in \mathbb{N}\ ,\, i_1, i_2, \cdots, i_t \in [r]\ : \\ i_0 + i_1 + i_2 + \cdots + i_t = n}} \binom{n}{i_0, i_1, \ldots, i_t} \qquad (1)$$

Lower bounds on the minimal length

- Let $\Theta_{t,r}(x) := \sum_{n=1}^{\infty} \frac{\theta_{t,r}(n)}{n!} x^n$ denote the exponential generating function for the sequence $(\theta_{t,r}(n))_{n\in\mathbb{N}}$
- Write $G_r(x) = \frac{1}{1!}x + \frac{1}{2!}x^2 + \cdots + \frac{1}{r!}x^r$
- Note $e^x = \sum_{j=0}^{\infty} \frac{x^j}{j!}$
- By using (1), we find that

$$\Theta_{t,r}(x) = e^x \cdot (G_r(x))^t \tag{2}$$

Lower bounds on the minimal length

- Since

$$\Theta_{t,r}(x) = e^x G_r(x)^t = G_r(x)\Theta_{t-1,r}(x) = \\ = \left(x + x^2/2 + \cdots + x^r/r!\right)\Theta_{t-1,r}(x)\ ,$$

- we have

$$\begin{aligned} \theta_{t,r}(n) &= n! \cdot [x^n]\ \left(\left(\frac{x}{1!} + \frac{x^2}{2!} + \cdots + \frac{x^r}{r!}\right)\Theta_{t-1,r}(x)\right) = \\ &= n!\left(\frac{\theta_{t-1,r}(n-1)}{(n-1)!\cdot 1!} + \frac{\theta_{t-1,r}(n-2)}{(n-2)!\cdot 2!} + \cdots + \frac{\theta_{t-1,r}(n-r)}{(n-r)!\cdot r!}\right) = \\ &= \binom{n}{1}\theta_{t-1,r}(n-1) + \binom{n}{2}\theta_{t-1,r}(n-2) + \cdots + \binom{n}{r}\theta_{t-1,r}(n-r). \end{aligned} \tag{3}$$

Starting point $t = 1$

- Note that, $\theta_{0,r}(n) = 1$ for all $n \geq 1$
- For $t = 1$, we get

$$\theta_{1,r}(n) = \binom{n}{1} \cdot 1 + \binom{n}{2} \cdot 1 + \ldots + \binom{n}{r} \cdot 1 = \sum_{i=1}^{r} \binom{n}{i}$$

- For fixed t, we can construct a table for $\theta_{t,r}(n)$ recursively, starting with $\theta_{1,r}(n)$

Lower bounds on the minimal length

Since

$$\theta_{t-1,r}(n-1) \geq \theta_{t-1,r}(n-2) \geq \cdots \geq \theta_{t-1,r}(n-r) \ ,$$

we obtain that for $n \geq 2r-1$:

$$\theta_{t,r}(n) \leq \left(\binom{n}{1} + \cdots + \binom{n}{r}\right) \cdot \theta_{t-1,r}(n-1) \leq r \cdot \binom{n}{r} \cdot \theta_{t-1,r}(n-1) \ , \tag{4}$$

which can be useful to prove bounds on $\theta_{t,r}(n)$ by induction.

Lower bounds on the minimal length

Theorem 2

If an $[n, k, t, r]$ *binary functional batch code exists and if* $n \geq \max\{t+1, 2r-1\}$, *then*

$$n \geq (t+r)/2 - 1 + \left((2^k - 1)(r-1)!\right)^{1/r}.$$

Example

Example, $r = 2$

By using (2), for $r = 2$ we have:

$$\begin{aligned}
\theta_{t,2}(n) \ &= \ n! \cdot [x^n] \left((x + x^2/2)^t \sum_{j=0}^{\infty} \frac{x^j}{j!} \right) \ = && (5)\\
&= n! \cdot \sum_{i=0}^{\min\{t,n-t\}} \frac{\binom{t}{i}}{2^i (n-t-i)!} \leq && (6)\\
&\leq \ \frac{(n)_t}{2^t} \cdot \sum_{i=0}^{t} \binom{t}{i} 2^{t-i} (n-t)_i \ \leq && (7)\\
&\leq (n)_t \cdot (n-t+2)^t / 2^t \ . && (8)
\end{aligned}$$

Lower bounds on the minimal length

When $r = 2$, the inequality (8) implies

$$\begin{aligned}(2^k - 1) &\leq (n)_t^{1/t}(n - t + 2)/2 \leq (n - (t-1)/2)(n - (t-2))/2 \\ &\leq \frac{1}{2}(n - 3t/4 + 5/4)^2 \,. \end{aligned} \tag{9}$$

Theorem 3

For any $[n, k, t, 2]$ functional batch code, $t \geq 1$, we have

$$n \geq \sqrt{2\,(2^k - 1)} + 3t/4 - 5/4 \,.$$

Numerical results 1

k	t	Theorem 3	Exact $\theta_{t,r}(n)$
2	4	5	5
3	8	9	10
4	16	17	19
5	32	31	38
6	64	58	74
7	128	111	146

Comparison of lower and upper bounds on minimal n for $r = 2$.

Numerical results 2

k	ZEY bound, $t = 2$	$t = 2,\ r = 2$	$t = 2,\ r = 3$	$t = 3,\ r = 3$	$t = 2,\ r = 5$
5	7	7	6	6	-
6	8	9	7	8	-
7	9	13	8	9	9
8	11	17	10	10	10
9	12	24	12	13	10
10	13	33	15	15	11
11	14	47	18	18	12
12	16	65	22	23	13
13	17	92	27	28	14
14	18	129	34	34	16
15	19	183	42	43	18

Lower bounds on n from Zhang-Etzion-Yaakobi (ZEY) and Theorem 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 better bounds for larger r.
- For more details and further results, refer to our paper.

Thank You for Listening!