Nearly Optimal Constructions of PIR and Batch Codes

Hilal Asi and Eitan Yaakobi
Technion - Israel Institute of Technology
2017
Outline

• Definitions
  - PIR Codes
  - Batch Codes

• Contribution

• The Array Construction
  - PIR Codes from the Array Construction
  - Batch Codes from the Array Construction

• Multiplicity Codes
  - PIR Codes from Multiplicity Codes
  - Batch Codes from Multiplicity Codes
Definition – Batch Code

Definition. A linear code $C$ will be called a $k$-batch code, if for every multiset request $\{i_0, i_1, \ldots, i_{k-1}\}$ of information bits there exist $k$ mutually disjoint sets $R_{i_0}, R_{i_1}, \ldots, R_{i_{k-1}}$ such that $x_{ij}$ can be recovered by $R_{ij}$ for every $j \in [k]$.

Examples:

$k = 1 \rightarrow E(x_1x_2 \ldots x_n) = x_1x_2 \ldots x_n$

$k = 2 \rightarrow E(x_1x_2 \ldots x_n) = x_1x_2 \ldots x_n x_{n+1}, \ x_{n+1} = x_1 + x_2 + \ldots + x_n$

- $\{i_0, i_1\}$ can be recovered by $\{x_{i_0}\}$ and $\{x_{i_1}\}$.

- $\{i_0, i_0\}$ can be recovered by $\{x_{i_0}\}$ and $\{x_i: i \neq i_0\}$. 
Batch Codes

• **Main question:** Given $n$ and $k$, what is the **optimal length** $N$ such that a $k$-batch code of dimension $n$ and length $N$ exists?

• We denote this value by $B(n,k)$.

• $B(n,k = 1) = n$

• $B(n,k = 2) = n + 1$
Definition – PIR Code

• Same as batch codes but queries of a single symbol
  
  - For every $i \in [n]$, there exist $k$ mutually disjoint recovery sets that can recover $x_i$.

• **Main question:** Given $n$ and $k$, what is the optimal length $N$ such that a $k$-PIR code of dimension $n$ and length $N$ exists?

• We denote this value by $P(n,k)$.

  • $P(n,k) \leq B(n,k)$
  
  • $P(n,k = 1) = n$
  
  • $P(n,k = 2) = n + 1$
Motivation

• Applications to Distributed Data Storage
• Cryptographic applications (PIR schemes)
• Codes with efficient encoding\decoding
Previous Work (PIR Codes)

• **Fazely, Vardy, and Yaakobi15:**
  - For any fixed $k$: $P(n, k) = n + \Theta(\sqrt{n})$ (RaoVardy16-Wooters16)

• **Lin and Costello04:** several constructions of availability codes (one-step majority logic decodable codes)
  - $P(n, k = \sqrt{n}) = n + O\left(\frac{\log 3}{2}\right)$
  - For $\epsilon > 0$, $P(n, k = n^\epsilon) = n + O\left(n^{0.5+\epsilon}\right)$

• For $k \leq n^{1/2}$ there are *asymptotically* optimal constructions
  $$\lim_{n \to \infty} P(n, k)/n = 1$$

• Define the *redundancy* of these codes
  $$r_p(n, k) = P(n, k) - n$$

• We will be interested in $k$ which is a function of $n$: $r_p(n, k = n^\epsilon) = O(n^\delta)$
Previous Work (PIR Codes)

\[ r_p(n, k = n^\epsilon) = O(n^{\delta}) \]

- For \( 0 < \epsilon < 0.5 \), \( k = n^\epsilon \) there are asymptotically optimal constructions

\[ \lim_{n \to \infty} P(n, k)/n = 1 \]

Figure 1: Asymptotic results for binary PIR codes
Contribution (PIR Codes)

\[ r_p(n, k = n^\epsilon) = O(n^\delta) \]

- For \( 0 < \epsilon < 1 \), \( k = n^\epsilon \) there are asymptotically optimal constructions

\[ \lim_{n \to \infty} P(n,k)/n = 1 \]

Figure 2: Asymptotic results for binary PIR codes
Contribution (PIR Codes)

\[ r_P(n, k = n^\epsilon) = O(n^\delta) \]

- For \( 0 < \epsilon < 1 \), \( k = n^\epsilon \) there are asymptotically optimal constructions
  \[ \lim_{n \to \infty} P(n, k) / n = 1. \]

- For \( 1 \leq \epsilon \), \( k = n^\epsilon \) there are nearly optimal constructions
  \[ \lim_{n \to \infty} P(n, k) = O(kn^\tau), \tau > 0. \]

- This is almost optimal since \( r_P(n, k) = \Omega(k) \).

Figure 3: Asymptotic results for binary PIR codes
Previous Work (Batch Codes)

- **Vardy and Yaakobi16:**
  - For any fixed $k$: $B(n, k) = n + O(\sqrt{n} \log(n))$

- **Dimakis, Gál, Rawat, and Song14:**
  - For $1/5 \leq \epsilon \leq 7/32$: $B(n, k = n^\epsilon) = n + O(n^{4\epsilon})$
  - For $7/32 \leq \epsilon \leq 1/4$: $B(n, k = n^\epsilon) = n + O(n^{7/8})$
  - $B(n, k = n^{1/3}) = O(n)$

- **Lipmaa and Skachek15:** Constructions of linear batch codes

- **Buzaglo, Yaakobi, Cassuto, and Siegel16:** $B(n, n) = O(n^{1.5})$

  - For $k \leq n^{1/4}$ there are *asymptotically* optimal constructions
    \[
    \lim_{{n \to \infty}} B(n, k)/n = 1
    \]

- Define the *redundancy* of these codes
  \[
  r_B(n, k) = B(n, k) - n
  \]
Previous Work (Batch Codes)

- For $0 < \epsilon < 0.25$, $k = n^\epsilon$ there are asymptotically optimal constructions

$$\lim_{n \to \infty} \frac{B(n,k)}{n} = 1.$$

Figure 4: Asymptotic results for binary batch codes
Contribution (Batch Codes)

- For $0 < \epsilon < 0.5$, $k = n^\epsilon$ there are \textit{asymptotically} optimal constructions
  \[ \lim_{n \to \infty} B(n, k)/n = 1. \]

- Construction of \textit{optimal} $k=5$-batch code.

- Construction of \textit{optimal non-binary} $k$-batch codes for \textit{fixed} $k$.

- Construction of $(r, k)$-batch codes with rate $\frac{r}{r+k}$.

![Figure 5: Asymptotic results for binary batch codes](image)
The Array Construction - Intuition

- 3-PIR code by Ishai et al.
  - Treats the input as 2-dimensional array
  - Adds one sum parity bit for every column row

<table>
<thead>
<tr>
<th>(x_0)</th>
<th>(x_1)</th>
<th>(\ldots)</th>
<th>(x_{s-1})</th>
<th>(r_0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x_s)</td>
<td>(x_{s+1})</td>
<td>(\ldots)</td>
<td>(x_{2s-1})</td>
<td>(r_1)</td>
</tr>
<tr>
<td>(\ldots)</td>
<td>(\ldots)</td>
<td>(\ldots)</td>
<td>(\ldots)</td>
<td>(\ldots)</td>
</tr>
<tr>
<td>(x_{s^2-s})</td>
<td>(x_{s^2-s+1})</td>
<td>(\ldots)</td>
<td>(x_{s^2-1})</td>
<td>(r_{s-1})</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(c_0)</th>
<th>(c_1)</th>
<th>(\ldots)</th>
<th>(c_{s-1})</th>
</tr>
</thead>
</table>
The Array Construction - Intuition

• Previous generalizations increased the dimension
• Consider diagonals with different slopes
  - Every slope can add one recovery set
  - There are \( s \) different slopes

<table>
<thead>
<tr>
<th></th>
<th>( X_0 )</th>
<th>( X_1 )</th>
<th>( X_2 )</th>
<th>( X_3 )</th>
<th>( X_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_5 )</td>
<td>( X_6 )</td>
<td>( X_7 )</td>
<td>( X_8 )</td>
<td>( X_9 )</td>
<td></td>
</tr>
<tr>
<td>( X_{10} )</td>
<td>( X_{11} )</td>
<td>( X_{12} )</td>
<td>( X_{13} )</td>
<td>( X_{14} )</td>
<td></td>
</tr>
</tbody>
</table>
The Array Construction - Intuition

• Previous generalizations increased the dimension
• Consider diagonals with different slopes
  - Every slope can add one recovery set
  - There are $s$ different slopes
The Array Construction - Intuition

- Previous generalizations increased the dimension
- Consider diagonals with different slopes
  - Every slope can add one recovery set
  - There are $s$ different slopes

<table>
<thead>
<tr>
<th></th>
<th>$x_0$</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_{10}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$\rho_{0,1}$</th>
<th>$\rho_{0,1}$</th>
<th>$\rho_{0,2}$</th>
<th>$\rho_{0,3}$</th>
<th>$\rho_{0,4}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\rho_{1,1}$</td>
<td></td>
<td>$\rho_{1,1}$</td>
<td>$\rho_{1,2}$</td>
<td>$\rho_{1,3}$</td>
<td>$\rho_{1,4}$</td>
</tr>
</tbody>
</table>
The Array Construction - Intuition

- Previous generalizations increased the dimension
- Consider diagonals with different slopes
  - Every slope can add one recovery set
  - There are $s$ different slopes
The Array Construction - Intuition

- One can construct 4 recovering sets for every bit

<table>
<thead>
<tr>
<th>X0</th>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
</tr>
</thead>
<tbody>
<tr>
<td>X5</td>
<td>X6</td>
<td>X7</td>
<td>X8</td>
<td>X9</td>
</tr>
<tr>
<td>X10</td>
<td>X11</td>
<td>X12</td>
<td>X13</td>
<td>X14</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ρ₀,₁</th>
<th>ρ₀,₁</th>
<th>ρ₀,₂</th>
<th>ρ₀,₃</th>
<th>ρ₀,₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>ρ₁,₁</td>
<td>ρ₁,₁</td>
<td>ρ₁,₂</td>
<td>ρ₁,₃</td>
<td>ρ₁,₄</td>
</tr>
<tr>
<td>ρ₂,₁</td>
<td>ρ₂,₁</td>
<td>ρ₂,₂</td>
<td>ρ₂,₃</td>
<td>ρ₂,₄</td>
</tr>
</tbody>
</table>
The Array Construction - Intuition

- One can construct 4 recovering sets for every bit

<table>
<thead>
<tr>
<th>X0</th>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
</tr>
</thead>
<tbody>
<tr>
<td>X5</td>
<td>X6</td>
<td>X7</td>
<td>X8</td>
<td>X9</td>
</tr>
<tr>
<td>X10</td>
<td>X11</td>
<td>X12</td>
<td>X13</td>
<td>X14</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(\rho_{0,1})</th>
<th>(\rho_{0,1})</th>
<th>(\rho_{0,2})</th>
<th>(\rho_{0,3})</th>
<th>(\rho_{0,4})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\rho_{1,1})</td>
<td>(\rho_{1,1})</td>
<td>(\rho_{1,2})</td>
<td>(\rho_{1,3})</td>
<td>(\rho_{1,4})</td>
</tr>
<tr>
<td>(\rho_{2,1})</td>
<td>(\rho_{2,1})</td>
<td>(\rho_{2,2})</td>
<td>(\rho_{2,3})</td>
<td>(\rho_{2,4})</td>
</tr>
</tbody>
</table>
The Array Construction - Intuition

• One can construct 4 recovering sets for every bit

<table>
<thead>
<tr>
<th>X0</th>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
</tr>
</thead>
<tbody>
<tr>
<td>X5</td>
<td>X6</td>
<td>X7</td>
<td>X8</td>
<td>X9</td>
</tr>
<tr>
<td>X10</td>
<td>X11</td>
<td>X12</td>
<td>X13</td>
<td>X14</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\rho_{0,1}$</th>
<th>$\rho_{0,1}$</th>
<th>$\rho_{0,2}$</th>
<th>$\rho_{0,3}$</th>
<th>$\rho_{0,4}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\rho_{1,1}$</td>
<td>$\rho_{1,1}$</td>
<td>$\rho_{1,2}$</td>
<td>$\rho_{1,3}$</td>
<td>$\rho_{1,4}$</td>
</tr>
<tr>
<td>$\rho_{2,1}$</td>
<td>$\rho_{2,1}$</td>
<td>$\rho_{2,2}$</td>
<td>$\rho_{2,3}$</td>
<td>$\rho_{2,4}$</td>
</tr>
</tbody>
</table>
The Array Construction - Intuition

- One can construct 4 recovering sets for every bit

<table>
<thead>
<tr>
<th>X0</th>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
</tr>
</thead>
<tbody>
<tr>
<td>X5</td>
<td>X6</td>
<td>X7</td>
<td>X8</td>
<td>X9</td>
</tr>
<tr>
<td>X10</td>
<td>X11</td>
<td>X12</td>
<td>X13</td>
<td>X14</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ρ_{0,1}</th>
<th>ρ_{0,1}</th>
<th>ρ_{0,2}</th>
<th>ρ_{0,3}</th>
<th>ρ_{0,4}</th>
</tr>
</thead>
<tbody>
<tr>
<td>ρ_{1,1}</td>
<td>ρ_{1,1}</td>
<td>ρ_{1,2}</td>
<td>ρ_{1,3}</td>
<td>ρ_{1,4}</td>
</tr>
<tr>
<td>ρ_{2,1}</td>
<td>ρ_{2,1}</td>
<td>ρ_{2,2}</td>
<td>ρ_{2,3}</td>
<td>ρ_{2,4}</td>
</tr>
</tbody>
</table>
The Array Construction - Problem

- **Problem:** Diagonals with different slopes might intersect at more than one point 😞

<table>
<thead>
<tr>
<th></th>
<th>X0</th>
<th>X1</th>
<th>X2</th>
<th>X3</th>
</tr>
</thead>
<tbody>
<tr>
<td>X4</td>
<td>X5</td>
<td>X6</td>
<td></td>
<td>X7</td>
</tr>
<tr>
<td>X8</td>
<td>X9</td>
<td>X10</td>
<td></td>
<td>X11</td>
</tr>
<tr>
<td>X12</td>
<td>X13</td>
<td>X14</td>
<td></td>
<td>X15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>X0</th>
<th>X1</th>
<th>X2</th>
<th>X3</th>
</tr>
</thead>
<tbody>
<tr>
<td>X4</td>
<td>X5</td>
<td>X6</td>
<td></td>
<td>X7</td>
</tr>
<tr>
<td>X8</td>
<td>X9</td>
<td>X10</td>
<td></td>
<td>X11</td>
</tr>
<tr>
<td>X12</td>
<td>X13</td>
<td>X14</td>
<td></td>
<td>X15</td>
</tr>
</tbody>
</table>
The Array Construction – Attempt 1

- Reduce the number of rows such that slopes do not cause a cyclic jump
- **Condition:** \( \text{slope} \cdot (#\text{rows} - 1) < #\text{columns} \)
- \( k \)-PIR **Redundancy** \( O(k^{1.5}\sqrt{n}) \).
The Array Construction – Attempt 2

• Removing the intersection without reducing the number of rows

• **Improved condition**: `#columns` is prime
The Array Construction – Attempt 2

- Removing the intersection without reducing the number of rows

- **Improved condition**: \(\#\text{columns} \) is prime

- \( k \)-PIR Redundancy \( O(k\sqrt{n}) \).
The Array Construction – Attempt 2

- Removing the intersection without reducing the number of rows

- **Improved condition**: \(\#\text{columns} \) is prime

*Denote by* \( C(r, p, S) \)
PIR Codes from The Array Construction

**Theorem.** Let $n = p^2$, $p$ is prime, $S \subseteq [p]$, $|S| = k \leq \sqrt{n}$. Then $C(r, p, S)$ is $k$-PIR code of dimension $n$ and redundancy $k\sqrt{n}$.

**Result:** $r_p(n, k) = O(k\sqrt{n})$ for $k \leq \sqrt{n}$.
Batch Codes from The Array Construction

Lemma. [Dimakis et.al. 14] Let $C$ be a $k$-PIR code. Assume that for every distinct indices $i, j \in [n]$, it holds that each recovering set of the $i$th bit intersects with at most one recovering set of the $j$th bit. Then, the code $C$ is a $k$-batch code.

Proof. Let $R = \{i_0, i_1, ..., i_{k-1}\}$ be the requested bits.

<table>
<thead>
<tr>
<th>$S^i_0$</th>
<th>$S^i_1$</th>
<th>$S^i_2$</th>
<th>$\cdots$</th>
<th>$S^i_{k-1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S^i_0$</td>
<td>$S^i_1$</td>
<td>$S^i_2$</td>
<td>$\cdots$</td>
<td>$S^i_{k-1}$</td>
</tr>
<tr>
<td>$S^i_0$</td>
<td>$S^i_1$</td>
<td>$S^i_2$</td>
<td>$\cdots$</td>
<td>$S^i_{k-1}$</td>
</tr>
<tr>
<td>$S^i_0$</td>
<td>$S^i_1$</td>
<td>$S^i_2$</td>
<td>$\cdots$</td>
<td>$S^i_{k-1}$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\ddots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$S^i_{k-1}$</td>
<td>$S^i_{k-1}$</td>
<td>$S^i_{k-1}$</td>
<td>$\cdots$</td>
<td>$S^i_{k-1}$</td>
</tr>
</tbody>
</table>
Batch Codes from The Array Construction

Lemma. [Dimakis et.al. 14] Let $C$ be a $k$-PIR code. Assume that for every distinct indices $i, j \in [n]$, it holds that each recovering set of the $i$th bit intersects with at most one recovering set of the $j$th bit. Then, the code $C$ is a $k$-batch code.

Proof. Let $R = \{i_0, i_1, \ldots, i_{k-1}\}$ be the requested bits.

<table>
<thead>
<tr>
<th>$S^i_0$</th>
<th>$S^i_1$</th>
<th>$S^i_2$</th>
<th>$\cdots$</th>
<th>$S^i_{k-1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S^i_0$</td>
<td>$S^i_1$</td>
<td>$S^i_2$</td>
<td>$\cdots$</td>
<td>$S^i_{k-1}$</td>
</tr>
<tr>
<td>$S^i_0$</td>
<td>$S^i_1$</td>
<td>$S^i_2$</td>
<td>$\cdots$</td>
<td>$S^i_{k-1}$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\ddots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$S^i_0$</td>
<td>$S^i_1$</td>
<td>$S^i_2$</td>
<td>$\cdots$</td>
<td>$S^i_{k-1}$</td>
</tr>
</tbody>
</table>
Batch Codes from The Array Construction

**Lemma.** [Dimakis et.al. 14] Let $C$ be a $k$-PIR code. Assume that for every distinct indices $i, j \in [n]$, it holds that each recovering set of the $i$th bit intersects with at most one recovering set of the $j$th bit. Then, the code $C$ is a $k$-batch code.

**Proof.** Let $R = \{i_0, i_1, \ldots, i_{k-1}\}$ be the requested bits.
Batch Codes from The Array Construction

Lemma. [Dimakis et.al. 14] Let $C$ be a $k$-PIR code. Assume that for every distinct indices $i, j \in [n]$, it holds that each recovering set of the $i$th bit intersects with at most one recovering set of the $j$th bit. Then, the code $C$ is a $k$-batch code.

Proof. Let $R = \{i_0, i_1, ..., i_{k-1}\}$ be the requested bits.

<table>
<thead>
<tr>
<th>$S_0^{i_0}$</th>
<th>$S_1^{i_1}$</th>
<th>$S_2^{i_2}$</th>
<th>...</th>
<th>$S_{k-1}^{i_k}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[x]</td>
<td>[x]</td>
<td>[x]</td>
<td>[x]</td>
<td>[x]</td>
</tr>
<tr>
<td>[x]</td>
<td>[x]</td>
<td>[x]</td>
<td>[x]</td>
<td>[x]</td>
</tr>
<tr>
<td>[x]</td>
<td>[x]</td>
<td>[x]</td>
<td>[x]</td>
<td>[x]</td>
</tr>
<tr>
<td>[x]</td>
<td>[x]</td>
<td>[x]</td>
<td>[x]</td>
<td>[x]</td>
</tr>
<tr>
<td>[x]</td>
<td>[x]</td>
<td>[x]</td>
<td>[x]</td>
<td>[x]</td>
</tr>
</tbody>
</table>
Batch Codes from The Array Construction

Lemma. [Dimakis et.al. 14] Let $C$ be a $k$-PIR code. Assume that for every distinct indices $i, j \in [n]$, it holds that each recovering set of the $i$th bit intersects with at most one recovering set of the $j$th bit. Then, the code $C$ is a $k$-batch code.

Proof. Let $R = \{i_0, i_1, \ldots, i_{k-1}\}$ be the requested bits.

<table>
<thead>
<tr>
<th>$S_{i_0}^0$</th>
<th>$S_{i_1}^1$</th>
<th>$S_{i_2}^2$</th>
<th>$\ldots$</th>
<th>$S_{i_{k-1}}^{k-1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_0^{i_0}$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$S_1^{i_1}$</td>
</tr>
<tr>
<td>$S_1^{i_0}$</td>
<td>$S_1^{i_1}$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$S_2^{i_2}$</td>
</tr>
<tr>
<td>$S_2^{i_0}$</td>
<td>$S_2^{i_1}$</td>
<td>$S_2^{i_2}$</td>
<td>$\times$</td>
<td>$S_3^{i_3}$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$S_{k-1}^{i_0}$</td>
<td>$S_{k-1}^{i_1}$</td>
<td>$S_{k-1}^{i_2}$</td>
<td>$\ldots$</td>
<td>$S_{k-1}^{i_{k-1}}$</td>
</tr>
</tbody>
</table>
Batch Codes from The Array Construction

- Usually, $C(r, p, S)$ will not satisfy the Lemma’s condition!

- How can we choose the set of slopes $S$ to satisfy the Lemma’s condition?
Definition. Let \( r \in \mathbb{Z}^+ \), and \( S \) a set of non-negative integers. \( S \) does not contain an \textbf{\( r \)-weighted arithmetic progression} if there do not exist \( s_1 \neq s_2 \neq s_3 \in S \) and \( 0 < x, y < r - 1 \), where \( x + y < r \), such that

\[
x s_1 + y s_2 = (x + y) s_3 \quad (*)
\]

We say that \( S \) does not contain an \textbf{\( r \)-weighted arithmetic progression modulo} \( p \) if (*) does not hold modulo \( p \).
4-weighted arithmetic progression

• \{0,1,3,4,7\} \(\times\)
  - \(1 \cdot 1 + 1 \cdot 7 = 2 \cdot 4\)

• \{1,4,8,10,22\} \(\times\)
  - \(1 \cdot 4 + 2 \cdot 10 = 3 \cdot 8\)

• \{0,1,10,11,23\} \(\checkmark\)
  - Does not contain 4-weighted arithmetic progression!
  - However, \(1 \cdot 1 + 1 \cdot 11 = 2 \cdot 0\) (mod 3)
  - It does contain 4-weighted arithmetic progression modulo 3.
Batch Codes from The Array Construction

**Main Theorem.** Let $r \leq p$, $S \subseteq [p]$ and $|S| = k$. If $p$ is prime and $S$ does not contain an $r$-weighted arithmetic progression modulo $p$ then $C(r, p, S)$ is a $k$-batch code of dimension $n = rp$.

**Goal:** For every $r$ and $p$, construct a large set $S \subseteq [p]$ that does not contain an $r$-weighted arithmetic progression (modulo $p$).
The Greedy Algorithm

Algorithm 1 The Greedy Algorithm

1: Initialize:
   \[ S \leftarrow \{0, 1\}, \ i = 2, \ num = 2 \]
2: while \( num < p \) do
3:   if \( S \cup \{num\} \) does not contain an \( r \)-arithmetic progression modulo \( p \)
4:      \[ S \leftarrow S \cup \{num\} \]
5:   i \leftarrow i + 1
6:   num \leftarrow num + 1
7: return \( S \)
The Greedy Algorithm

**Theorem.** Let \( r, p \) be positive integers, such that \( p \) is prime. Then the output of the Greedy Algorithm is a set \( S \) of size at least \( k \), where \( k \) is the largest integer such that \( p > 2k^2r^2 \).

**Proof.** (Idea)

First, we bound the number of elements that cannot join to \( S \) using the size of the set. Then we prove that if \( |S| < k \), then more elements can be added to \( S \).
Batch Codes from The Array Construction

**Theorem.** For any $n$ and $k$ such that $k = o\left(\sqrt{n}\right)$, there exists a $k$-batch code of dimension $n$ and redundancy $O(n^3k^3)$.

In particular, for $0 < \epsilon < \frac{1}{2}$, $r_B(n, k) = O(n^{2+\frac{5\epsilon}{3}})$. 
Sets without $r$-arithmetic progression for small values of $r$

- For $r = 3$, containing 3-weighted arithmetic progression is equivalent to the problem of containing 3-term arithmetic progression: i.e. set $S$ such that there does not exist $s_1, s_2, s_3 \in S$, $s_1+s_2 = 2s_3$.

- Erdös and Turan initiated the problem in 1936.

- Behrend showed that for every $0 < \alpha < 1$, and sufficiently large $p$, there exists a set $S \subseteq [p]$ without 3-term arithmetic progression, of size $\Omega(p^\alpha)$.

- His technique can be extended to any fixed value of $r$. 
Reed-Muller Codes

• Let $s, d < q$ positive integers.
• The code is defined over $F_q$.
• Let $P(x)$ be polynomial in $s$ variables, $\deg(P) \leq d$.
• $P(x_1, x_2, \ldots, x_s) \rightarrow (P(w_1, w_2, \ldots, w_s))_{(w_1, w_2, \ldots, w_s) \in F_q^s}$
• $C = \left\{(P(w))_{w \in F_q^s} : \deg(P) \leq d \right\}$
Reed-Muller Codes

**Example.** Let $s = 2, d = 3, q = 4$.

$$P(x_1, x_2) = x_1 x_2 + x_1 x_2^2 \text{ over } F = GF(4).$$

$$P(x_1, x_2) \rightarrow (w_1 w_2 + w_1 w_2^2)_{w \in F_4^2}$$
Recovering Set for Reed-Muller Codes

• Assume $s = 2$.

Since $d < q$, $p(\lambda)$ is unique.

\[
p(\lambda) = P(w + \lambda v)
\]

\[
P(w) = P(w + 0) = p(0)
\]
Multiplicity Codes

- Generalize Reed-Muller Codes (*Kopparty et al. 11*).
  - Evaluate the polynomials and their derivatives.
  - Used to construct *LDC* codes.

- Improved rate over Reed-Muller codes.
The Hasse derivative

• For a vector $i = (i_1, i_2, \ldots, i_s)$ of non-negative integers, its weight $\text{wt}(i) = \sum_{j=1}^{s} i_j$.

• For a vector $i = (i_1, i_2, \ldots, i_s)$ and $x = (x_1, x_2, \ldots, x_s)$, denote $x^i = x_1^{i_1} x_2^{i_2} \cdots x_s^{i_s}$.
The Hasse derivative

• **Definition.** For a polynomial $P(x)$ and a non-negative vector $i$, the $i$-th Hasse derivative of $P(x)$, $P^{(i)}(x)$, is the coefficient of $z^i$ in the polynomial $P(x + z)$.

• $P(x + z) = \sum_i P^{(i)}(x) z^i$
The Hasse derivative

Example.

\[ P(x_1, x_2) = x_1 x_2 + x_1 x_2^2 \] over \( GF(4) \).

\[ P(x + z) = P(x_1 + z_1, x_2 + z_2) = x_1 x_2 + x_1 x_2^2 + (x_2 + x_2^2) z_1 + x_1 z_2 + z_1 z_2 + x_1 z_2^2 + z_1 z_2^2 \]

\[ P^{(0,0)}(x) = x_1 x_2 + x_1 x_2^2, \]

\[ P^{(1,0)}(x) = x_2 + x_2^2, \]

\[ P^{(0,1)}(x) = x_1, \]

\[ P^{(1,1)}(x) = 1, \]

\[ \vdots \]
Multiplicity Codes

- Let $m, s, d, q$ be parameters, $\frac{d}{m} < q$.
- Let $P(x)$ be polynomial over $F_q$ in $s$ variables, $\deg P \leq d$.
- Define $P(<m)(w) = \left( P(i)(w) \right)_{\text{wt}(i)<m}$.
- $P(x_1, x_2, \ldots, x_s) \rightarrow (P(<m)(w))_{w \in F_q^s}$.
- $C(m, d, s, q) = \left\{ (P(<m)(w))_{w \in F_q^s} : \deg P \leq d \right\}$.

Reed-Muller Codes

- $d < q$ positive integers.
- The code is defined over $F_q$.
- Let $P(x)$ be polynomial in $s$ variables, $\deg(P) \leq d$.
- $P(x_1, x_2, \ldots, x_s) \rightarrow (P(w_1, w_2, \ldots, w_s))_{(w_1, w_2, \ldots, w_s) \in F_q^s}$.
- $C = \left\{ (P(w))_{w \in F_q^s} : \deg(P) \leq d \right\}$.
**Example.** Let $q = 4, m = s = 2, d = 3$.

$P(x_1, x_2) = x_1 x_2 + x_1 x_2^2$

$P^{(<2)}(x) = \left( P^{(0,0)}(x), P^{(1,0)}(x), P^{(0,1)}(x) \right)$

$= (x_1 x_2 + x_1 x_2^2, x_2 + x_2^2, x_1)$

$P(x_1, x_2) \rightarrow (w_1 w_2 + w_1 w_2^2, w_2 + w_2^2, w_1)_{w \in F_4^2}$
Recovering Set for Multiplicity Codes

• Let $m = s = 2$.

Let $\mathbf{w} = (w_1, w_2)$

\[ P^{(<2)}(\mathbf{w}) = (P^{(0,0)}(\mathbf{w}) = ?, P^{(1,0)}(\mathbf{w}) = ?, P^{(0,1)}(\mathbf{w}) = ?) \]

Since $\frac{d}{m} < q$, $p(\lambda)$ is unique.

\[ P(\mathbf{w}) = P(\mathbf{w} + 0) = p(0) \]
Recovering Set for Multiplicity Codes

• Let $m = s = 2$.

\[
\sum_{j} c_j \lambda^j = p(\lambda) = P(w + \lambda v) = \sum_i P^{(i)}(w) v^i \lambda^{wt(i)}
\]

\[
c_1 = P^{(1,0)}(w)v^{(1,0)} + P^{(0,1)}(w)v^{(0,1)}
\]

\[
c_1 = P^{(1,0)}(w)v_1 + P^{(0,1)}(w)v_2
\]
Recovering Set for Multiplicity Codes

• Let $m = s = 2$.

\[ w + \lambda v, \lambda \in F_q \]
\[ p(\lambda) = P(w + \lambda v) \]
\[ w + \lambda v', \lambda \in F_q \]

\[ c_1 = P^{(1,0)}(w)v_1 + P^{(0,1)}(w)v_2 \]
\[ c_1' = P^{(1,0)}(w)v'_1 + P^{(0,1)}(w)v'_2 \]

$w = (w_1, w_2)$
\[ P^{(<2)}(w) = (P^{(0,0)}(w)=?, P^{(1,0)}(w)=?, P^{(0,1)}(w)=?) \]
Recovering Set for Multiplicity Codes

- What happens when \( s = 3 \)?

\[
\mathbf{w} + \lambda \mathbf{v}, \lambda \in F_q
\]

\[
c_1 = p^{(1,0,0)}(\mathbf{w})v_1 + p^{(0,1,0)}(\mathbf{w})v_2 + p^{(0,0,1)}(\mathbf{w})v_3
\]

\[
c_1' = p^{(1,0,0)}(\mathbf{w})v_1' + p^{(0,1,0)}(\mathbf{w})v_2' + p^{(0,0,1)}(\mathbf{w})v_3'
\]

\[
c_1'' = p^{(1,0,0)}(\mathbf{w})v_1'' + p^{(0,1,0)}(\mathbf{w})v_2'' + p^{(0,0,1)}(\mathbf{w})v_3''
\]

\[
\mathbf{w} = (w_1, w_2, w_3)
\]

\[
P^{(<2)}(\mathbf{w}) = (p^{(0,0,0)}(\mathbf{w}), p^{(1,0,0)}(\mathbf{w}) = ?, p^{(0,1,0)}(\mathbf{w}) = ?, p^{(0,0,1)}(\mathbf{w}) = ?)
\]

\[
\text{NOT EVERY COMBINATION OF 3 LINES WILL WORK!}
\]
Recovering Set for Multiplicity Codes

• In general, \( m^{s-1} \) lines are needed to recover \( P^{(<m)}(w) \).

• These lines must be carefully chosen in order to guarantee successful recovery.

• We show how to construct \( \left\lfloor \frac{q}{m} \right\rfloor^{s-1} \) disjoint recovering sets.

**Corollary.** For all \( m, s, d, q \) such that \( \frac{d}{m} < q \), the code \( C(m, d, s, q) \) is a \( k \)-PIR code with \( k = \left\lfloor \frac{q}{m} \right\rfloor^{s-1} \).
PIR Codes from Multiplicity Codes

**Theorem.** For every positive $s \geq 2$ and $0 < \alpha < 1$, there exists a $k$-PIR code over $F_Q$ of dimension $n$ and redundancy $r$ such that

\[
Q = n^{\Theta(n^\alpha)} \\
k = \Theta \left( n^{\left(\frac{1}{s} - \frac{1}{s} \left(1 - \alpha\right)\right)} \right) \\
r = O\left( n^{1 - \frac{\alpha}{s}} \right)
\]
Binary PIR Codes from Multiplicity Codes

**Theorem.** For every positive $s \geq 2$ and $0 < \alpha < 1$, there exists a binary $k$-PIR code of dimension $n$ and redundancy $r$ such that

\[
\begin{align*}
    k &= \Theta \left( \frac{n}{\log n} \right)^{1 - \frac{1}{s}(1 - \alpha)}^{1 + \alpha} \\
    r &= O\left( n^{1 - \frac{\alpha}{s(1 + \alpha) \log(n)}} \right)
\end{align*}
\]
PIR Codes from Multiplicity Codes

\[ r_B(n, k = n^\epsilon) = O(n^{\delta}) \]

Fig. 2. Asymptotic results for binary PIR codes
PIR Codes for $k \geq n$

\[ r_B(n, k = n^\varepsilon) = O(n^\delta) \]

Figure 3: Asymptotic results for binary PIR codes
Batch Codes from Multiplicity Codes

\[ w_1 \]
\[ w_2 \]
\[ w_3 \]
Batch Codes from Multiplicity Codes

**Theorem.** Let $0 < \alpha \leq 0.5$. Then, there exist a binary \textit{k-batch code} of dimension $n$ and \textit{redundancy} $r$ such that

\[
k = \Theta \left( \left( \frac{n}{\log n} \right)^{0.5-\alpha} \right)\\
\]

\[
r = O(n^{1-\frac{\alpha}{3}} \log(n))
\]
Batch Codes Summary

Figure 5: Asymptotic results for binary batch codes
Questions?