Covering Perfect Hash Families

Charles J. Colbourn\textsuperscript{1}
joint with Erin Lanus and Kaushik Sarkar

\textsuperscript{1}School of Computing, Informatics, and Decision Systems Engineering
Arizona State University

May 2017
Covering Array. Definition

- Let $N$, $k$, $t$, and $v$ be positive integers.
- Let $C$ be an $N \times k$ array with entries from an alphabet $\Sigma$ of size $v$; we typically take $\Sigma = \{0, \ldots, v - 1\}$.
- When $(\nu_1, \ldots, \nu_t)$ is a $t$-tuple with $\nu_i \in \Sigma$ for $1 \leq i \leq t$, $(c_1, \ldots, c_t)$ is a tuple of $t$ column indices $(c_i \in \{1, \ldots, k\})$, and $c_i \neq c_j$ whenever $\nu_i \neq \nu_j$, the $t$-tuple $\{(c_i, \nu_i) : 1 \leq i \leq t\}$ is a $t$-way interaction.
- The array covers the $t$-way interaction $\{(c_i, \nu_i) : 1 \leq i \leq t\}$ if, in at least one row $\rho$ of $C$, the entry in row $\rho$ and column $c_i$ is $\nu_i$ for $1 \leq i \leq t$.
- Array $C$ is a covering array $\text{CA}(N; t, k, v)$ of strength $t$ when every $t$-way interaction is covered.
- $\text{CAN}(t, k, v)$ is the minimum $N$ for which a $\text{CA}(N; t, k, v)$ exists.
Covering Array. Definition

- Let $N$, $k$, $t$, and $v$ be positive integers.
- Let $C$ be an $N \times k$ array with entries from an alphabet $\Sigma$ of size $v$; we typically take $\Sigma = \{0, \ldots, v - 1\}$.
- When $(\nu_1, \ldots, \nu_t)$ is a $t$-tuple with $\nu_i \in \Sigma$ for $1 \leq i \leq t$, $(c_1, \ldots, c_t)$ is a tuple of $t$ column indices ($c_i \in \{1, \ldots, k\}$), and $c_i \neq c_j$ whenever $\nu_i \neq \nu_j$, the $t$-tuple $\{(c_i, \nu_i) : 1 \leq i \leq t\}$ is a $t$-way interaction.
- The array covers the $t$-way interaction $\{(c_i, \nu_i) : 1 \leq i \leq t\}$ if, in at least one row $\rho$ of $C$, the entry in row $\rho$ and column $c_i$ is $\nu_i$ for $1 \leq i \leq t$.
- Array $C$ is a covering array $\text{CA}(N; t, k, v)$ of strength $t$ when every $t$-way interaction is covered.
- $\text{CAN}(t, k, v)$ is the minimum $N$ for which a $\text{CA}(N; t, k, v)$ exists.
Covering Array. Definition

- Let $N, k, t,$ and $v$ be positive integers.
- Let $C$ be an $N \times k$ array with entries from an alphabet $\Sigma$ of size $v$; we typically take $\Sigma = \{0, \ldots, v - 1\}$.
- When $(\nu_1, \ldots, \nu_t)$ is a $t$-tuple with $\nu_i \in \Sigma$ for $1 \leq i \leq t$, $(c_1, \ldots, c_t)$ is a tuple of $t$ column indices ($c_i \in \{1, \ldots, k\}$), and $c_i \neq c_j$ whenever $\nu_i \neq \nu_j$, the $t$-tuple $\{(c_i, \nu_i) : 1 \leq i \leq t\}$ is a $t$-way interaction.
- The array covers the $t$-way interaction $\{(c_i, \nu_i) : 1 \leq i \leq t\}$ if, in at least one row $\rho$ of $C$, the entry in row $\rho$ and column $c_i$ is $\nu_i$ for $1 \leq i \leq t$.
- Array $C$ is a covering array $CA(N; t, k, v)$ of strength $t$ when every $t$-way interaction is covered.
- $CAN(t, k, v)$ is the minimum $N$ for which a $CA(N; t, k, v)$ exists.
## Covering Array

**CA(13;3,10,2)**

<p>| | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Covering Perfect Hash Families

Charles J. Colbourn
joint with Erin Lanus and Kaushik Sarkar

### Covering Arrays

Covering Perfect Hash Families
1. How precisely can we determine $\text{CAN}(t, k, v)$?

2. When we can show $\text{CAN}(t, k, v) \leq N$, can we construct a $\text{CA}(N; t, k, v)$ efficiently and explicitly?
A Random Method

- Fix $t$ and $v$ independent of $k$.
- In an array chosen uniformly at random from $\{0, \ldots, v - 1\}^{N \times k}$, the probability that any specific $t$-way interaction is not covered is $\left(1 - \frac{1}{vt}\right)^N$.
- So the expected number of uncovered $t$-way interactions is $\binom{k}{t} v^t \left(1 - \frac{1}{vt}\right)^N$.
- When this expected number is less than 1, some array has all $t$-way interactions covered!
A Random Method

- Take logarithms of \((k\binom{k}{t}v^t(1 - \frac{1}{v^t})^N < 1\) to get

\[
\text{CAN}(t, k, v) \leq \frac{t}{\log \frac{v^t}{v^t-1}} \log k(1 + o(1))
\]

- \((\text{CAN}(t, k, v) = \Omega(\log k)\) is easy: No two columns can be identical.)
Derandomizing
The Stein-Lovász-Johnson Method

- Instead generate one row at a time at random from \( \{0, \ldots, v - 1\}^k \).
- The expected number of \( t \)-way interactions covered by this row for the first time is therefore \( \frac{1}{v^t} \) times the number of as-yet-uncovered \( t \)-way interactions.
- Stein (1974), Lovász (1975), and Johnson (1974): Select a row that covers the largest number of as-yet-uncovered \( t \)-way interactions.
- But finding such a row is NP-hard!
- Our method: Select a row that covers at least the average, efficiently.
Derandomizing
The Discrete Stein-Lovász-Johnson Method

- Iterate to produces a covering array with the number of rows no larger than the initial random approach.
- In fact, we do better: After each row is selected, the number of uncovered interactions is an integer.
Constructive Methods

- We need to construct covering arrays for applications in software and hardware testing, evaluating complex engineered systems, etc. So how are we doing in this regard?
- Let’s look at some parameter situations and compare to the discrete SLJ bound.
Computational Results

3-CAs with 3 symbols

\[ N \quad \text{vs.} \quad \log(\text{Number of Factors}) \]
Computational Results

5-CAs with 5 symbols

$N$ vs $\log($Number of Factors$)$
Computational Results

6-CAs with 7 symbols
Better asymptotics

LLL

- SLJ and Discrete SLJ do not account for the limited statistical dependence among the events of coverage of interactions.
- The (symmetric version of the) Lovász Local Lemma (LLL) yields a better bound (obtained by Godbole, Skipper, and Sunley in 1996)

\[
\text{CAN}(t, k, \nu) \leq \frac{t - 1}{\log \frac{\nu^t}{\nu^t - 1}} \log k(1 + o(1))
\]
Francetic and Stevens (2016) made the first improvement in 20 years, using an entropy compression technique

\[ \text{CAN}(t, k, \nu) \leq \frac{\nu(t - 1)}{\log \left( \frac{\nu^{t-1}}{\nu^{t-1}-1} \right)} \log k(1 + o(1)) \]
Constructive algorithms

- Applications require explicit constructions of arrays, not asymptotic bounds.
- Can we meet the bounds efficiently when $t$ and $v$ are fixed?
  - Discrete SLJ: Yes, an efficient conditional expectation method ("density") deterministically chooses a row as good as average (Bryce-C, 2007, 2009)
  - LLL: Yes if you allow expected polynomial time: Moser-Tardos (2010) give a resampling method that succeeds within a linear expected number of resamplings.
  - Francetic-Stevens: Not clear (yet), but stay tuned.
Constructive algorithms

Why are the tables so bad?

- When $v = 7$, $t = 6$, and $k = 50$ there are $1,869,524,964,300$ interactions to cover!

- Density stores coverage information for each, and the storage requirement is enormous.

- Moser-Tardos recomputes coverage for each (at a cost of $O(NT)$) for every resampling, and the number of resamplings needed is a random variable.
Constructive algorithms
Sample space reduction

- Consider covering arrays that are invariant under the action of a group on the symbols of the array, in order to make the space to search for an array much smaller.
- We consider three permutation groups acting on the symbols.
  - the cyclic group of order \( v \), which partitions the interactions on \( t \) columns into \( v^{t-1} \) orbits of length \( v \);
  - the Frobenius or affine group when \( v \) is a prime power, which partitions the interactions into \( \frac{v^{t-1}-1}{v-1} \) orbits of length \( v(v-1) \) and one orbit of length \( v \);
  - PGL when \( v+1 \) is a prime power, which partitions the interactions into orbits of length \( v(v-1)(v-2) \), \( v(v-1) \), and \( v \).
Constructive algorithms
Covering Orbits

- Now we cover orbits of interactions and apply the group to recover the covering array at the end.
- We can apply the SLJ paradigm and the density methods in the same way in the cyclic and Frobenius cases (For density, see Colbourn 2013).
- We can apply LLL and the Moser-Tardos methods in the same way in the cyclic and Frobenius cases.
- This reduces time and storage for density, and time for Moser-Tardos — But what does it do to the asymptotic bounds?
Better asymptotics
Cyclic LLL

- Applying LLL with the cyclic group, we reproduce the Francetic and Stevens (2016) bound

$$\text{CAN}(t, k, \nu) \leq \frac{\nu(t - 1)}{\log \left( \frac{\nu^{t-1}}{\nu^{t-1} - 1} \right)} \log k(1 + o(1))$$

- and we get a Moser-Tardos type method that runs in expected polynomial time to meet the bound.
Better asymptotics

Frobenius LLL

- Applying LLL with the Frobenius group, we improve on the Francetic and Stevens (2016) bound

\[
\text{CAN}(t, k, v) \leq \frac{v(v-1)(t-1)}{\log \left( \frac{\nu^{t-1}}{\nu^{t-1} - v + 1} \right)} \log k(1 + o(1))
\]

- and we get a Moser-Tardos type method that runs in expected polynomial time to meet the bound.
What about PGL?

- Covering orbits of length $v$ can be done with $v$ constant rows.
- Covering orbits of length $v(v - 1)(v - 2)$ can be done with LLL (or Moser-Tardos).
- But orbits of length $v(v - 1)$ are a problem, in that their probability of being covered in a random selection is much smaller.
- So the road to higher levels of sharply $\ell$-transitive groups acting on the symbols seems blocked.
George Sherwood suggested a framework for constructing covering arrays using finite fields.

Let $q$ be a prime power, and let $\mathbb{F}_q$ be the finite field of order $q$.

Let $\mathcal{R}_{t,q} = \{r_0, \ldots, r_{q^t-1}\}$ be the set of all (row) vectors of length $t$ with entries from $\mathbb{F}_q$, and let $\mathcal{T}_{t,q}$ be the set of all column vectors of length $t$ with entries from $\mathbb{F}_q$, not all 0.

A vector $x \in \mathcal{T}_{t,q}$ is a permutation vector, so called because the multiplication of all $r_i \in \mathcal{R}_{t,q}$ with $x$ can be interpreted as $q^{t-1}$ permutations of $\mathbb{F}_q$. 
Lemma
(Sherwood-Martirosyan-Colbourn) Let \( \mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_t\} \) be a set of vectors from \( \mathcal{T}_{t,q} \). The array \( A = (a_{ij}) \) formed by setting \( a_{ij} \) to be the product of \( r_i \) and \( x_j \) is a CA\( (q^t; t, t, q) \) if and only if the \( t \times t \) matrix \( X = [\mathbf{x}_1 \cdots \mathbf{x}_t] \) is nonsingular.

Proof.
Array \( A \) contains some row \( \mathbf{b} \) at least twice exactly when \( rX = b \) has more than one solution.
Covering Perfect Hash Families Setup III

- $(0, \ldots, 0)^T$ cannot appear in a nonsingular matrix, so it is not in $T_{t,q}$.

- For any nonzero $\mu \in \mathbb{F}_q$, substituting $\mu x_i$ for $x_i$ may change the ordering of the rows of the covering array produced but does not alter the fact that it is a covering array.

- Define $\langle x \rangle = \{\mu x : \mu \in \mathbb{F}_q, \mu \neq 0\}$. When $x$ is not all $0$, we can select as the representative of $\langle x \rangle$ the unique vector whose first nonzero coordinate is the multiplicative identity element.

- Let $V_{t,q}$ be the set of representatives of the column vectors in $T_{t,q}$, and let $U_{t,q}$ be the set of vectors in $V_{t,q}$ whose first coordinate is not zero.

- Then $|V_{t,q}| = \frac{q^t - 1}{q - 1} = \sum_{i=0}^{t-1} q^i$, and that $|U_{t,q}| = q^{t-1}$. 
Covering Perfect Hash Families Definition

A covering perfect hash family CPHF\((n; k, q, t)\) is an \(n \times k\) array \(C = (c_{ij})\) with entries from \(V_{t,q}\) so that, for every set \(\{\gamma_1, \ldots, \gamma_t\}\) of distinct column indices, there is at least one row index \(\rho\) of \(C\) for which \([c_{\rho\gamma_1} \cdots c_{\rho\gamma_t}]\) is nonsingular; call this a covering \(t\)-set and say that the \(t\)-set of columns is covered.

It is a Sherwood covering perfect hash family, SCPHF\((n; k, q, t)\), if in addition each entry is in \(U_{t,q}\).

Lemma

Suppose that \(C\) is a CPHF\((n; k, q, t)\). Then

1. there exists a CA\((n(q^t - 1) + 1; t, k, q)\); and
2. there exists a CA\((n(q^t - q) + q; t, k, q)\) if \(C\) is an SCPHF\((n; k, q, t)\).
Choose entries of an $n \times k$ array $A$ uniformly at random from $T_{t,q}$.

Let $T$ be a set of $t$ columns of $A$.

The probability that $A$ does not contain a covering $t$-set for $T$ can easily be computed.

The total number of $t$-sets is $(q^t - 1)^t$, and the number that are covering $t$-sets is $\prod_{i=0}^{t-1} (q^t - q^i)$.

So within one row of $A$, the probability that the columns of $T$ are not covering is

$$\phi_{t,q} := 1 - \frac{\prod_{i=0}^{t-1} (q^t - q^i)}{(q^t - 1)^t} = 1 - \prod_{i=1}^{t-1} \frac{q^t - q^i}{q^t - 1}.$$
Lemma

For all \( q \geq 3 \) and \( t \geq 3 \),

\[
\frac{1}{q} \leq \phi_{t,q} \leq \frac{q + 1}{q^2}.
\]

This leads to three better asymptotic bounds for covering arrays!

<table>
<thead>
<tr>
<th>SLJ CPHF</th>
<th>LLL CPHF</th>
<th>SLJ (pp) CPHF</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \frac{\nu^t t}{2 \log \nu - \log(\nu+1)} )</td>
<td>( \frac{\nu^t (t-1)}{2 \log \nu - \log(\nu+1)} )</td>
<td>( \frac{\nu^t (t-1)}{2 \log \nu - \log(\nu+1)} )</td>
</tr>
</tbody>
</table>
In 2006, Sherwood, Martirosyan, and Colbourn used a backtracking method to find a handful of SCPHFs leading to best covering array numbers.

In 2007, Walker and Colbourn used a tabu search approach to find about twenty more.

Here we use a conditional expectation ("density") method based on SLJ, and a column replacement method based on the Moser-Tardos deterministic LLL to get many hundreds more.

Indeed we make no effort to list them all here, but instead summarize when improvements are found.
CPHF Computations for Strength Three

Strength Three, up to 10000 columns

<table>
<thead>
<tr>
<th>$q$</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>11</th>
<th>13</th>
<th>16</th>
<th>17</th>
<th>19</th>
<th>23</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>q</td>
<td>Strength Four, up to 2500 columns</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-----------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>![Graph for q=3]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>![Graph for q=4]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>![Graph for q=5]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>![Graph for q=7]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>![Graph for q=8]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>![Graph for q=9]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>![Graph for q=11]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>![Graph for q=13]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>![Graph for q=16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>![Graph for q=17]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>![Graph for q=19]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>![Graph for q=23]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>![Graph for q=25]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CPHF Computations for Strength Five

Strength Five up to 600 columns

<table>
<thead>
<tr>
<th>q</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
</tr>
</tbody>
</table>
Covering Perfect Hash Families

Charles J. Colbourn
joint with Erin Lanus and Kaushik Sarkar

Covering Arrays

CPHF Computations for Strength Six

Strength Six up to 200 columns

$q$  3  4  5  7  8  9  11  13  16  17  19  23  25
Next Steps: CPHF Restrictions

▶ Sherwood CPHFs restrict CPHFs by ensuring that every two rows produce a common set of constant rows, or a common covering array of strength 1.

▶ Other restrictions ensure that certain pairs of rows produce a common covering array of strength $\ell < t$, forcing the presence of duplicated rows in the covering array produced, which can then be deleted.

▶ The promise of getting much better results by such restrictions is quite real, and we have computational results in this direction.
Next Steps: Linear Codes and Projective Geometry

- Covering perfect hash families are built on the points of the projective geometry PG\((t, q)\), and we believe that in this formulation it should be possible to get powerful results for specific parameters.