Supplementary Information

Baptiste Chevalier[1,*], Shimpei Yamaguchi[1], Wojciech Roga[1], and Masahiro Takeoka[1,2]

[1]Department of Electronics and Electrical Engineering, Keio University, Yokohama 223-8522, Japan
[2]Advanced ICT Research Institute, National Institute of Information and Communications Technology (NICT), Koganei, Tokyo 184-8795, Japan
[*]chevalier.baptiste@keio.jp

ABSTRACT

In this Supplementary Information, we present supplementary properties of the mathematical objects used in the main text. We also provide a detailed discussion and intuition regarding the MCCO algorithm applied to a simplified interpretable example, and provide proofs of less important quantities which appear in the main text. These materials are not crucial to the main text; however, they provide additional insight into the mechanism of the algorithm and the meaning and relevance of the functions used in numerical simulations.
In Supplementary Information 1, we discuss some notable properties of the transform from the small set of rules to the large scale compressible functions. Such a function is introduced in the Methods and used in numerical simulations. It can be interpreted as a large scale code that embeds the rules. We prove properties such as injectivity, properties of the norm, and distances.
In Supplementary Information 2, we discuss the coherence we mentioned in the Theoretical analysis. We give a formal expression for the coherence of structured matrices that were used in the numerical simulations of the paper.
In Supplementary Information 3, we discuss the effect of thresholding on the variance of the objective function and its importance in the MCCO algorithm.
Finally, in Supplementary Information 4, we discuss a simple example that explains working of MCCO algorithm. The example shows how the solution of Problem II can be reduced to the solution of Problem I. This specific example uses a structured matrix acting on nearest neighbor pairs of bits and we interpret sketch functions as correlations that can be extracted from the samples.

The notations used in this document are the same as in the main text, refer to the paper for definitions.

1 Properties of the F_N transform

1.1 Properties

The proofs follow the list of statements.

Theorem 1. Given a string a (a rule) of size k:

1. $\|F_N(a)\| \geq 0$.
2. $F_k = id$ (if $n = k$, the transform is the identity), i.e., $F_k(a) = \mathbf{a}$.
3. F_N can be written as a $2^N \times 2^k$ rectangular matrix.

The proofs of Theorem 1 are trivial.

Theorem 2. [Injectivity]

$$F_N(a) = F_N(b) \iff a = b. \tag{1}$$

Definition 1. [ℓ_1-Norm of the transform]

$$\|F_N(a)\|_1 = \sum_{x\in\{0,1\}^N} \sum_{i=1}^{N-k+1} \delta_{h(x_i,a),0}. \tag{2}$$

Theorem 3. [ℓ_1-Norm Recursive Expression]

$$\|F_{N+1}(a)\|_1 = 2\|F_N(a)\|_1 + 2^{N-k+1}. \tag{3}$$

Theorem 4. [ℓ_1-Norm Explicit Expression]

$$\|F_{k+l}(a)\|_1 = 2^l(l+1). \tag{4}$$

Notice that all elements a have the same norm!

Theorem 5. [Rules separations]
For any $a, b \neq a \in \{0,1\}^k$ and for any $d \geq 0$, there exists N such that

$$\|F_N(a) - F_N(b)\|_2 \geq d.$$

Theorem 6. [Moments arbitrary distance]
Let $\Phi : \mathbb{R}^{2^N} \to \mathbb{R}^M$ be a map computing M generalized moments of the function π whose density is $f_{\mathscr{R}}$. The sketch map $\Phi \in \mathbb{R}^{2^N \times M}$ is chosen as a typical compressive sensing map (with RIP or random projections). For any $a, b \neq a \in \{0,1\}^k$ and for any $D \geq 0$, there exists N such that:

$$\|\Phi(F_N(a)) - \Phi(F_N(b))\|_2 \geq D. \tag{5}$$

1.2 Proofs or the above properties

Lemma 1. prerequisite for Theorem 2

$$a \neq 0^{(k)} \implies F_N(a)[i_a] = 1. \tag{6}$$

Proof. The proof is by contradiction.
$a = a_1 \cdots a_k$ matches with $i_a = 0^{(N-k)} a_1 \cdots a_k$ at the last k bits. If we assume $F_N(a)[i_a] > 1$, then, a matches with i_a at somewhere else, say $a_1 \cdots a_k = 0^{(l)} a_1 \cdots a_{k-l}, (l \in \mathbb{Z}^+, 1 \leq l \leq k-1)$. Then we get $a = 0^{(k)}$ by solving

$$\begin{aligned}
a_1 &= 0, \\
&\vdots \\
a_l &= 0, \\
a_{l+1} &= a_1, \\
&\vdots \\
a_k &= a_{k-l}.
\end{aligned}$$

This contradicts with $a \neq 0^{(k)}$. □

Proof. of Theorem 2
The proof that $a = b \implies F_N(a) = F_N(b)$: is trivial from the definition of F_N. Thus we only need to prove $F_N(a) = F_N(b) \implies a = b$. To do so, we use the contraposition that is, we want to prove that $a \neq b \Longrightarrow F_N(a) \neq F_N(b)$.
Set

$$\begin{aligned}
&a = a_1 \cdots a_k, (a_i \in \{0,1\}, i = 1, \cdots k), \\
&b = b_1 \cdots b_k, (b \in \{0,1\}, i = 1, \cdots k), \\
&0^{(m)} = \underbrace{0 \cdots 0}_{m}, \\
&i_a = 0^{(N-k)} a_1 \cdots a_k, \\
&i_b = 0^{(N-k)} b_1 \cdots b_k,
\end{aligned}$$

and $F_N(a)[i]$ where $i \in \{0,1\}^n$ denotes the i-th element of the vector $F_N(a)$.

Case i) When $a = 0^{(k)}$
$F_N(a)[0^{(N)}] = N - k + 1$, by the definition of F_N. $F_N(b)[0^{(N)}] = 0$, since $b \neq a = 0^{(k)}$. Thus, $F_N(a) \neq F_N(b)$.

Case ii) When $a \neq 0^{(k)}$
Let $b \neq 0^{(k)}$, since the case of $b = 0^{(k)}$ can be proved completely the same as the case of $a = 0^{(k)}$. We prove by contradiction. Assume $F_N(a) = F_N(b)$. This gives

$$F_N(a)[i_a] = F_N(b)[i_a], \tag{7}$$
$$F_N(a)[i_b] = F_N(b)[i_b]. \tag{8}$$

From Lemma 1 and equation (7), $F_N(b)[i_a] = 1$. Thus, for an integer $k_1,\ 1 \leq k_1 \leq k-1$,

$$b_1 \cdots b_k = 0^{(k_1)} a_1 \cdots a_{k-k_1} \tag{9}$$

since $0 \neq b \neq a$.
From Lemma 1 and equation (8), $F_N(a)[i_b] = 1$. Thus, for an integer $k_2,\ 1 \leq k_2 \leq k-1$,

$$\begin{aligned} a_1 \cdots a_k &= 0^{(k_2)} b_1 \cdots b_{k-k_2} \\ &= \begin{cases} 0^{(k_2)} 0^{(k_1)} a_1 \cdots a_{k-k_1-k_2} & \text{if } k-k_2 \geq k_1+1 \\ 0^{(k_2)} 0^{(k-k_2)} & \text{if } k-k_2 \leq k_1 \end{cases} \end{aligned}$$

by equation (9) and $0 \neq a \neq b$. By considering the same way in the proof of Lemma 1, a will be $0^{(k)}$ in both cases. This contradicts with $a \neq 0^{(k)}$. Therefore, $F_N(a) \neq F_N(b)$. □

Lemma 2. prerequisite for Theorem 3

$$\begin{aligned} F_{N+1}(a) &= \begin{pmatrix} F_N(a) \\ F_N(a) \end{pmatrix} + \mathbf{a} \otimes \mathbf{1}_{\mathbf{2^{N-k+1}}} \\ &= \mathbf{1}_2 \otimes F_N(a) + \mathbf{a} \otimes \mathbf{1}_{\mathbf{2^{N-k+1}}}, \end{aligned}$$

where $\otimes$ is the tensor product, $\mathbf{1}_n$ denotes n length column vector with all elements 1, and $\mathbf{a}$ is the 2^k-length column vector of which a-th element is 1 and the rest is 0.

Proof. An $(N+1)$-bit string $x^{(N+1)} \in \{0,1\}^{N+1}$ can be made by putting 0 or 1 in front of an N-bit string $x^{(N)} \in \{0,1\}^N$. We can divide $\{0,1\}^N$ into two sets: a set X which consists of N-bit strings which start with $(k-1)$ bits matching the $(k-1)$ last bits of a, i.e. $a_2 \cdots a_k$ and its complementary set $\bar{X}$ made of the remaining N-bit strings. Put a_1 in front of $x^{(N)} \in X$ and generate $(N+1)$-bit string $a_1 x^{(N)}$. Then the number of k-bit strings in $a_1 x^{(N)}$ that matches with a increases by one from the number of k-bit strings in $x^{(N)}$ that matches with a, because $x^{(N)} \in X$ starts with $a_2 \cdots a_k$. In other cases, i.e.,

- putting $1-a_1$ in front of $x^{(n)} \in X$ and
- putting a_1 or $1-{}^{a}_{1}$ in front of $x^{(n)} \in \bar{X}$,

the number of k-bit strings that matches with a does not change. Writing this explicitly,

$$F_{N+1}(a)[y x^{(N)}] = \begin{cases} F_N(a)[x^{(N)}] + 1 & \text{if } y = a_1 \wedge x^{(N)} \in X, \\ F_N(a)[x^{(N)}] & \text{else.} \end{cases}$$

Therefore, we can calculate $F_{N+1}(a)$ by the following procedure;

1. Concatenate $F_N(a)$ and $F_N(a)$,
2. Add 1 to the elements of which indexes start from k-bit string a.

The 2^{N+1} length vector indicating the position of $(N+1)$-bit strings which start from a can be written as $\mathbf{a} \otimes \mathbf{1}_{\mathbf{2^{N-k+1}}}$, since there are 2^{N+1-k} strings like that. Thus, the equation in Lemma 2 holds. □

Proof. of Theorem 3 By taking the ℓ_1-norm on each side of the equation from Lemma 2, it can easily be shown. (both vectors on the right hand side of the equation are vectors with non-negative elements.) □

Proof. of Theorem 4
By substituting N in Theorem 3 to $k+l$,

$$\|F_{k+l+1}(a)\|_1 = 2\|F_{k+l}(a)\|_1 + 2^{l+1}.$$

Set $\|F_{k+l}(a)\|_1 = s_l$. Then,

$$\begin{aligned} s_{l+1} &= 2s_l + 2^{l+1}, \\ \frac{s_{l+1}}{2^{l+1}} &= \frac{s_l}{2^l} + 1. \end{aligned}$$

Thus,

$$\begin{aligned} \frac{s_l}{2^l} &= \frac{s_0}{2^0} + l, \\ \frac{s_l}{2^l} &= 1 + l, \quad (\because s_0 = \|F_k(a)\|_1 = \|\mathbf{a}\|_\mathbf{1} = \mathbf{1}) \\ \|F_{k+l}(a)\|_1 &= s_l \\ &= 2^l(l+1). \end{aligned}$$

□

Lemma 3. prerequisite for Theorem 5
When $a \neq b$,

$$\begin{aligned} &\|F_{k+l}(a) - F_{k+l}(b)\|_2^2 \\ \geq &\begin{cases} 2^{l+1}\left(\frac{3}{5}l - \frac{1}{9} - \frac{\frac{4}{3}k + \frac{2}{3}l + \frac{20}{9}}{2^{k-2}}\right) & (k : \text{even}) \\ 2^{l+1}\left(\frac{3}{5}l - \frac{1}{9} - \frac{\frac{5}{3}k + \frac{1}{3}l + \frac{16}{9}}{2^{k-2}}\right) & (k : \text{odd}). \end{cases} \end{aligned}$$

Proof. of Lemma 3

$$\begin{aligned} &\|F_N(a) - F_N(b)\|_2^2 \\ =&F_N(a)\cdot F_N(a) + F_N(b)\cdot F_N(b) - 2F_N(a)\cdot F_N(b) \end{aligned}$$

Thus, we will analyze $F_N(a)\cdot F_N(b)$. From

$$F_{k+l}(a) = \sum_{i=1}^{l+1}\left(\mathbf{1}_2^{\otimes(i-1)} \otimes |a\rangle \otimes \mathbf{1}_2^{\otimes(l+1-i)}\right),$$

the inner product becomes

$$\begin{aligned} &F_{k+l}(a)\cdot F_{k+l}(b) \\ =&\sum_{i=1}^{l+1}\left(\mathbf{1}_2^{\otimes(i-1)} \otimes |a\rangle \otimes \mathbf{1}_2^{\otimes(l+1-i)}\right)^\mathsf{T} \\ &\quad \sum_{i'=1}^{l+1}\left(\mathbf{1}_2^{\otimes(i'-1)} \otimes |b\rangle \otimes \mathbf{1}_2^{\otimes(l+1-i')}\right) \\ =&\sum_{i=i'}\left(\mathbf{1}_2^{\otimes(i-1)} \otimes |a\rangle \otimes \mathbf{1}_2^{\otimes(l+1-i)}\right)^\mathsf{T} \\ &\quad \left(\mathbf{1}_2^{\otimes(i'-1)} \otimes |b\rangle \otimes \mathbf{1}_2^{\otimes(l+1-i')}\right) \\ +&\sum_{i\neq i'}\left(\mathbf{1}_2^{\otimes(i-1)} \otimes |a\rangle \otimes \mathbf{1}_2^{\otimes(l+1-i)}\right)^\mathsf{T} \\ &\quad \left(\mathbf{1}_2^{\otimes(i'-1)} \otimes |b\rangle \otimes \mathbf{1}_2^{\otimes(l+1-i')}\right). \end{aligned}$$

For the first term,

$$\begin{aligned}
&\sum_{i=i'}\left(\mathbf{1}_2^{\otimes(i-1)}\otimes|a\rangle\otimes\mathbf{1}_2^{\otimes(l+1-i)}\right)^{\mathsf{T}}\\
&\qquad\left(\mathbf{1}_2^{\otimes(i'-1)}\otimes|b\rangle\otimes\mathbf{1}_2^{\otimes(l+1-i')}\right)\\
=&\sum_{i=1}^{l+1}\left(\mathbf{1}_2^{\otimes(i-1)\mathsf{T}}\mathbf{1}_2^{\otimes(i-1)}\right)\langle a|b\rangle\left(\mathbf{1}_2^{\otimes(l+1-i)\mathsf{T}}\mathbf{1}_2^{\otimes(l+1-i)}\right)\\
=&\begin{cases}0 & (a\neq b)\\ (l+1)2^l & (a=b)\end{cases},
\end{aligned}$$

since $\langle a|b\rangle=\delta_{a,b}$.
For the second term with $a\neq b$, the positions of $|a\rangle$ and $|b\rangle$ differ, so the calculation of the inner product will not be easy as the case of the first term. Here we assume that $l\geq k$ and will divide the second term into two cases; (i) when the positions of $|a\rangle$ and $|b\rangle$ do not overlap, and (ii) when they overlap. The case of $l<k$ can be regarded as only considering Case (ii).
Case i): when the positions of $|a\rangle$ and $|b\rangle$ do not overlap
It can be illustrated as $(l-k+1)$ cases shown below (there are $\mathbf{1}_2$s in the blank places):

$$\begin{aligned}
&a_1a_2\cdots a_{k-1}a_k\\
&\qquad\qquad b_1b_2\cdots b_{k-1}b_k,\\
&a_1a_2\cdots a_{k-1}a_k\\
&\qquad\qquad\underbrace{\phantom{b_1b_2\cdots b_{k-1}b_k}}_{1}b_1b_2\cdots b_{k-1}b_k,\\
&a_1a_2\cdots a_{k-1}a_k\\
&\qquad\qquad\underbrace{\phantom{b_1b_2\cdots b_{k-1}b_k}}_{2}b_1b_2\cdots b_{k-1}b_k,\\
&\qquad\qquad\vdots\\
&\qquad\qquad\vdots\\
&a_1a_2\cdots a_{k-1}a_k\\
&\qquad\qquad\underbrace{\phantom{b_1b_2\cdots b_{k-1}b_k}}_{l-k}b_1b_2\cdots b_{k-1}b_k.
\end{aligned}$$

Here I showed the case when a_k is in front of b_1. The opposite case (when b_k is in front of a_1) can be considered in the completely same way. For all of them, the inner product will be 2^{l-k}, because the inner product of $|a_i\rangle$ (or $|b_i\rangle$) and $\mathbf{1}_2$ will be 1 for all $1\leq i\leq k$ (there are $2k$ of this inner products), and the inner product of $\mathbf{1}_2$ and $\mathbf{1}_2$ will be 2 (there are $(l-k)$ of this inner product). When there is g-length gap between the positions of a_k and b_1, there are $(l-k-g+1)$ options for the position of a_1. Thus, the sum of inner products is

$$2\cdot2^{l-k}\sum_{g=0}^{l-k}(l-k-g+1)=2^{l-k}(l-k+1)(l-k+2).$$

Case ii): when the position of $|a\rangle$ overlaps with the position of $|b\rangle$

These cases will be illustrated as below; when a_1 is in front of b_1 (Case ii-1)

$$
\begin{aligned}
&a_1a_2\cdots a_{k-1}a_k\\
&\quad b_1b_2\cdots \quad b_{k-1}b_k,
\end{aligned}
$$

$$
\begin{aligned}
&a_1a_2a_3\cdots a_{k-1}a_k\\
&\qquad b_1b_2\cdots \quad b_{k-2}b_{k-1}b_k,
\end{aligned}
$$

$$
\begin{aligned}
&\vdots\\
&\vdots\\
&a_1a_2\cdots a_{k-1}a_k\\
&\qquad\qquad b_1b_2\cdots b_{k-1}b_k,
\end{aligned}
$$

and when b_1 is in front of a_1 (Case ii-2)

$$
\begin{aligned}
&\quad a_1a_2 \quad \cdots \quad a_{k-1}a_k\\
&b_1b_2\cdots \quad b_{k-1}b_k,
\end{aligned}
$$

$$
\begin{aligned}
&\quad a_1a_2a_3\cdots a_{k-2}a_{k-1}a_k\\
&b_1b_2b_3 \quad \cdots \quad b_k,\\
&\vdots\\
&\vdots\\
&\qquad\qquad a_1a_2\cdots a_{k-1}a_k\\
&b_1b_2\cdots b_{k-1}b_k.
\end{aligned}
$$

For both (Case ii-1) and (Case ii-2) the jth pattern represents the case when the positions of a_1 and b_1 are j-separated. For j-separated case, the sum of the inner product is

$$
\begin{cases}
(l-j+1)2^{l-j} & \text{when overlapping part of } a \text{ and } b \text{ is same,}\\
0 & \text{else.}
\end{cases}
$$

This value depends on a and b. We will calculate the maximum value of the inner product of this part. The task is to find the possible combination which gives the maximum inner product value from the below table.

j	Case ii-1	Case ii-2
1	$l2^{l-1}$	$l2^{l-1}$
2	$(l-1)2^{l-2}$	$(l-1)2^{l-2}$
$\vdots$	$\vdots$	$\vdots$
$k-1$	$(l-k+2)2^{l-k+1}$	$(l-k+2)2^{l-k+1}$

Supplementary Table S1. Maximum inner product value for each separation value j

First, choose $l2^{l-1}$ and $l2^{l-1}$. This is because

$$
\begin{aligned}
&(l-1)2^{l-2}+\cdots+(l-k+2)2^{l-k+1}\\
=&\sum_{j=2}^{k-1}(l-j+1)2^{l-j}\\
=&2^{l-1}\left[l-2-\left(\frac{1}{2}\right)^{k-2}(k+l+2)\right]\\
\leq&2^{l-1}l,
\end{aligned}
$$

so you get more than half of the possible inner product value by choosing them. If you choose these two, $a_2\cdots a_k = b_1\cdots b_{k-1}$ and $a_1\cdots a_{k-1} = b_2\cdots b_k$ have to be simultaneously held. You can easily see that this condition gives

$$
\begin{aligned}
a &= 01010101\cdots\\
b &= 10101010\cdots
\end{aligned}
$$

or the opposite. This also satisfies the conditions for j: odd. You can see it by sliding a (or b) to the right for j: odd positions. If you try to satisfy any of the condition for j: even for the above a and b, it will immediately gives you $a = b = 0000\cdots$(or $1111\cdots$).
Finally, the maximum inner product value for case (ii) is

$$
\begin{aligned}
&2\sum_{j:\text{ odd}}(l-j+1)2^{l-j}\\
=&\begin{cases}2\sum_{m=1}^{n}[l-(2m-1)+1]2^{l-(2m-1)} & (k=2n)\\ 2\sum_{m=1}^{n-1}[l-(2m-1)+1]2^{l-(2m-1)} & (k=2n-1)\end{cases}\\
=&\begin{cases}\frac{2^{l+2}}{9}\left[3l-2+\frac{6n-3l+2}{4^n}\right]\\ \frac{2^{l+2}}{9}\left[3l-2+\frac{6n-3l-4}{4^{n-1}}\right]\end{cases}\\
=&\begin{cases}\frac{2^{l+2}}{9}\left[3l-2+\frac{3k-3l+2}{2^k}\right] & (k:\text{even})\\ \frac{2^{l+2}}{9}\left[3l-2+\frac{3k-3l-1}{2^{k-1}}\right] & (k:\text{odd}).\end{cases}
\end{aligned}
$$

For the second term with $a = b$, the sum of inner product value will be the sum of

$$2^{l-k}(l-k+1)(l-k+2)$$

which is from Case (i) and

$$2\sum_{j=1}^{k-1}(l-j+1)2^{l-j} = 2^l\left[2l-2-\left(\frac{1}{2}\right)^{k-2}(k+l+2)\right]$$

which is the sum of all the values in TABLE S1 from Case (ii). We also observe that $F_N(a)\cdot F_N(a)$ does not depend on a. Thus, we can rewrite $\|F_N(a)-F_N(b)\|_2^2$ as

$$\|F_N(a)-F_N(b)\|_2^2 = 2[F_N(a)\cdot F_N(a)-F_N(a)\cdot F_N(b)].$$

Therefore we have to check the difference between $F_N(a)\cdot F_N(a)$ and $F_N(a)\cdot F_N(b)$. For the first term, the difference is $(l+1)2^l$. For the second term, the difference appears in Case (ii), and the minimum difference is

$$
\begin{aligned}
&2^l\left[2l-2-\left(\frac{1}{2}\right)^{k-2}(k+l+2)\right]\\
-&\begin{cases}\frac{2^{l+2}}{9}\left[3l-2+\frac{3k-3l+2}{2^k}\right] & (k:\text{even})\\ \frac{2^{l+2}}{9}\left[3l-2+\frac{3k-3l-1}{2^{k-1}}\right] & (k:\text{odd}).\end{cases}
\end{aligned}
$$

Add these differences and double it, and we get Lemma 3. □

Proof. of Theorem 5 We combine both previous lemmas. Lemma 3 tells us that the distance between a and b increases exponentially when N grows. Thus, for any $d \geq 0$, there exists N such that $\|F_N(a) - F_N(b)\|_2 \geq d$. □

Proof. of Theorem 6

According to the Johnson-Lindenstrauss lemma, we have:

$$\begin{aligned}(1-\delta)\|F_N(a) - F_N(b)\|_2 &\leq \|\Phi(F_N(a) - F_N(b))\|_2 \\ &\leq (1+\delta)\|F_N(a) - F_N(b)\|_2,\end{aligned}$$

where $\delta \in [0,1]$ is a constant and for a number of measurements scaling as $M = O(\frac{k}{\delta^2})$. In particular:

$$\|F_N(a) - F_N(b)\|_2 \leq \frac{1}{1-\delta}\|\Phi(F_N(a) - F_N(b))\|_2.$$

However, from Theorem 5 we know that there exists N such that:

$$d \leq \|F_N(a) - F_N(b)\|_2,$$

for any d. And so, for the same N, and because Φ is linear:

$$d(1-\delta) \leq \|\Phi(F_N(a)) - \Phi(F_N(b))\|_2.$$

Because this holds for any d, one just need to chose it in such a way that $d \geq D/(1-\delta)$ to get the desired property. □

2 Coherence of structured matrices

Theorem 7. The coherence of a structured matrix using nearest-neighbor patterns of size k acting on a space of N bits is given by

$$\mu = \frac{N-k}{N-k+1}.$$

Proof. The k-uplet can be positioned starting from $N-(k-1)$ distinct positions (e.g. $N-1$ for pairs, $N-2$ for triplets etc...).

So any column of the structured matrix has $N-k+1$ entries and the associated normalization factor is $\frac{1}{\sqrt{N-k+1}}$.

Then for each of the $N-k+1$ starting position, the first column matches with the second column except for the last position, i.e. when we look at each bit individually.

The number of match which contribute to the scalar product is then $N-k$. □

Theorem 8. The coherence of a structured matrix using all possible patterns of size k acting on a space of N bits is given by

$$\mu = \frac{\binom{k}{N-1}}{\binom{k}{N}}.$$

Proof. For full matrices we can place the measured k-uplet at $\binom{k}{N}$ starting positions.

So any column of the structured matrix has $\binom{k}{N}$ entries and the normalization factor is $\frac{1}{\sqrt{\binom{k}{N}}}$.

Then for each of the $\binom{k}{N}$ starting position, the first column matches each time with the second column except when one of the measured bit is in last position, i.e. when we look at each bit individually.

The number of case where this does not happened is exactly $\binom{k}{N-1}$. □

3 Effect of Thresholding

We can show that a clever use of thresholding usually results in lowering the variance.

We define T_t the thresholding operator which sets every value lower than t to 0, i.e.

$$T_t : x \mapsto \begin{cases} x \text{ if } x \geq t \\ 0 \text{ else} \end{cases} .$$

Theorem 9. [Threshold increases probabilities]

Let $\sigma^{(t)^2} = \text{Var}[\Phi(T_t f_{\mathscr{R}}(x))]$ be the variance of the thresholded function. Recall, σ^2 is the variance of the original function $f_{\mathscr{R}}$.

There exists a value t for the thresholding parameter for which the variance decreases:

$$\sigma^{(t)^2} \leq \sigma^2. \tag{10}$$

The following lemma is used for the proof.

Lemma 4. [Concentration Bound with Thresholding]

Let $\sigma^{(t)^2} = \text{Var}[\Phi(T_t f_{\mathscr{R}}(x))]$ be the variance of the thresholded function. The following expression holds:

$$\sigma^{(t)^2} = p(t)\sigma_1^2(t) + \mu_1^2(t)(1 - p(t)p(t), \tag{11}$$

where $p(t)$ is the probability to sample from the non-zero region of $T_t f_{\mathscr{R}}$ after thresholding, μ_1 and σ_1^2 are respectively the mean and variance of this non-zero region.

Proof. To express $\sigma^{(t)^2}$ we use the law of total variance. For a given t, let's divide the space of $\Phi(T_t f_{\mathscr{R}}(x))$ into its zero part ($G = 0$) and its non-zero part ($G = 1$). The law of total variance allows us to write the variance $\text{Var}[\Phi(T_t f_{\mathscr{R}}(x))]$ (which we will call $\text{Var}(X)$ for convenience) as follows:

$$\text{Var}(X) = \mathbb{E}[\text{Var}[X|G]] + \text{Var}[\mathbb{E}[X|G]],$$

where G is our grouping variable separating the zero part from the non-zero part. Assume

- $\mu_0 = 0, \quad \sigma_0 = 0.$
- $\mu_1, \quad \sigma_1$

are the mean and variance of respectively the zero and non-zero parts. The first term of $\text{Var}(X)$ (within group variance) turns into $\mathbb{E}[\text{Var}[X|G]] = p_1\sigma_1^2$.

The second term (between group variance) can be computed as well: we have $\mu = p_0\mu_0 + p_1\mu_1$ for the total mean

$$\begin{aligned} \text{Var}[\mathbb{E}[X|G]] &= p_0(\mu_0 - \mu) + p_1(\mu_1 - \mu) \\ &= p_0(\mu_0 - \mu) + p_1(\mu_1 - \mu) \\ &= p_0\mu_1^2 p_1^2 + p_1\mu_1^2(1 - p_1)^2 \\ &= \mu_1^2 p_0 p_1. \end{aligned}$$

Finally we get:

$$\sigma^{(t)} = \sqrt{\text{Var}(x)} = \sqrt{p(t)\sigma_1^2(t) + \mu_1^2(t)(1 - p(t)p(t)}.$$

□

We now prove the Theorem 9

Proof. [Threshold increases probability] (intuition behind the proof)

Let's show that the theorem holds for some nontrivial case with moderate t. First, we notice that the terms in Lemma 4 do not depend on permutations of the domain of $f_{\mathscr{R}}$, therefore for this analysis we can reorder the function according to decreasing values. Let's assume that $f_{\mathscr{R}}(x)$ can be approximated by an exponential law $f_{\mathscr{R}}(x) = \lambda e^{-\lambda x}$ (after reordering the values). Therefore, we can approximate each terms from lemma 4:

$$p(t) = \frac{-\log(t/\lambda)}{\lambda},$$

$$\begin{aligned}\mu_1(t) &= \int_0^{p(t)} x \cdot \lambda e^{-\lambda x} dx \\ &= \frac{1}{\lambda} - \frac{t}{\lambda^2} + \frac{t\log(t/\lambda)}{\lambda^2},\end{aligned}$$

$$\begin{aligned}\sigma_1(t) &= \int_0^{p(t)} x^2 \cdot \lambda e^{-\lambda x} - \mu^2 \\ &= \frac{(\lambda^2 - t^2 + 2t^2\log[t/\lambda] - t(\lambda+t)\log[t/\lambda]^2)}{\lambda^4}.\end{aligned}$$

Putting everything together in Lemma 4, we obtain an explicit expression for the value $\sigma^{(t)^2}$. For a fixed λ, the function in t increases before reaching a maximum and then decreases. One can also check $\lim_{t\to\infty} \sigma^{(t)^2} = 0$. Finally, since the function $f_{\mathscr{R}}$ is continuous, the Intermediate Value Theorem tells us that there exists a value of t for which $\sigma^{(t)^2} \leq \sigma^2$. This end the proof of Theorem 9. □

4 Intuition behind Algorithm 1

In this Supplementary Information, we provide a heuristic justification of the algorithm supported by numerical simulation and rigorous proofs of lemmas and propositions whenever possible. We focus on a specific example and explain the detailed behavior of Alogirhtm 1.

Consider a simplified situation with a single 3-bit rule $r = r_1 r_2 r_3$. We define our target problem, which is to find a binary string x_I^* that contains as many sub-string identical to r as possible. Notice that r is unknown and does not need to be known, while x_I^* is learned based on rewards given to randomly selected binary strings, where for a given string the reward is equal to the number of times the rule r appears in it.

In Algorithm 1, we solve Problem II where the rows of matrix Φ, are given by

Duplets

$$\begin{aligned}\Phi^{i,i+1}_{x_1,x_2} &= 1_1 \otimes \ldots \otimes 1_{i-1} \\ &\otimes \quad (1-x_1,x_1) \otimes (1-x_2,x_2) \otimes 1_{i+2} \otimes \ldots \otimes 1_N\end{aligned} \tag{12}$$

for the $[4(i-1)+2x_1+x_2+1]$-th row and x_1 and x_2 are binary variables.

Define variable $\mu_x^i = \Phi_x^i Y$, where

$$\Phi_x^i = 1_1 \otimes \ldots \otimes 1_{i-1} \otimes (1-x,x) \otimes 1_{i+1} \otimes \ldots \otimes 1_N$$

is a **Singulet** map and

$$Y = (1-y_1,y_1) \otimes \ldots \otimes (1-y_n,y_n),$$

is a vector whose entries are zero except at the position given by the binary variables $y_i \in \{0,1\}$. If vectors Y are randomly chosen from a distribution, the variable μ_x^i is a random variable. Assume Y are defined by arguments of the function $T_t S f_{\mathscr{R}}$—lines sampled uniformly from $f_{\mathscr{R}}$ and thresholded.

Moments $\varphi^{i,i+1}_{x_1,x_2} = \langle \mu^i_{x_1} \mu^{i+1}_{x_2} \rangle$ are calculated as $\Phi^{i,i+1}_{x_1,x_2} T_t S f_{\mathscr{R}}$, where $\Phi^{i,i+1}_{x_1,x_2}$ is given in (12). Up to normalization, these moments express averages over sampled elements of $f_{\mathscr{R}}$ with weights corresponding to the value each element. Analogously, we consider moments $\varphi^{i,i+1,i+2}_{x_1,x_2,x_3} = \langle \mu^i_{x_1} \mu^{i+1}_{x_2} \mu^{i+2}_{x_3} \rangle$. These averages are related to appropriate correlation functions

$$corr^{i_1 \ldots i_n}_{x_{i_1},\ldots,x_{i_n}} = \frac{\langle (\mu^{i_1}_{x_{i_1}} - \langle \mu^{i_1}_{x_{i_1}} \rangle) \ldots (\mu^{i_n}_{x_{i_n}} - \langle \mu^{i_n}_{x_{i_n}} \rangle) \rangle}{\langle \mu^{i_1}_{x_{i_1}} \rangle \ldots \langle \mu^{i_n}_{x_{i_n}} \rangle}.$$

Therefore, loosely speaking, the moments express correlations between a given bit configuration $x_1 \ldots x_n \in \{0,1\}^n$ and the appearance of this configuration in important elements of the sample from $f_{\mathscr{R}}$. The importance is specified by values of $f_{\mathscr{R}}$ that remain after sampling and thresholding.

In Algorithm 1 we solve Problem II, which is, we find a string x_{II}^* for which the sum of correlations for neighboring variables is maximized,

$$x_{II}^* = \arg\max \sum_{i=1}^{N-1} \varphi^{i,i+1}_{x_1,x_2}. \tag{13}$$

Moreover, the way Algorithm 1 assigns the values $\varphi^{i,i+1}_{x_1,x_2}$ induces correlations between these values.

Our numerical analysis shows that solving Problem II often suffices to solve Problem I. Because x_I^* contains several times $r_1 r_2 r_3$ it is true that r_2 follows r_1 more often than in a random string (the same for the pair r_3 and r_2), and obviously, r_3 follows the pair $r_1 r_2$ more often than in random sequences. Therefore, the minimum requirement for any solver of Problem I should be to propose candidate solutions x which:

- Condition 1: possibly often contain pairs of bits $r_1 r_2$ and $r_2 r_3$, and
- Condition 2: the pairs appear in proper order, $r_2 r_3$ likely appear after $r_1 r_2$.

Notice that the solution of Problem II is a binary string x_{II}^* which is defined by the lower indices of the variables $\mu^i_{x_i}$ for which the sum of correlations $\varphi^{i,i+1}_{x_1,x_2} = \langle \mu^i_{x_1} \mu^{i+1}_{x_2} \rangle$ is maximum (as in Equation (13)). In the following proposition (proven in Supplementary Information 4.1), we claim that pairs of variables with $x_1 x_2$ matching substrings of r have larger expected moments than bits that does not match r.

Proposition 1. Let $\varphi_x^{i,i+1} = \Phi_x^{i,i+1} f_{\mathscr{R}}$,

(i) for the problem with single rule r, for any $2 \leq i \leq N-2$, if r' is a substring of r, and z is not, then

$$\varphi_{r'}^{i,i+1} \geq \varphi_{z}^{i,i+1},$$

(ii) for the problem with single rule r, for any i, there exists at least one r' substring of r such that, for any string z:

$$\varphi_{r'}^{i,i+1} \geq \varphi_{z}^{i,i+1},$$

(iii) for the problem with single rule r, for any i, there exists at least one r' substring of r such that, for any string z the expectation values $\langle \varphi_{r}^{i,i+1} \rangle = \Phi_{x}^{i,i+1} T_t S f_{\mathscr{R}}$ satisfy:

$$\langle \varphi_{r'}^{i,i+1} \rangle \geq \langle \varphi_{z}^{i,i+1} \rangle.$$

Therefore, solvers of Problem II (like Algorithm 1) promotes outputs x_{II}^{*} including substrings of r, hence Condition 1 is satisfied. Condition 2 is also expected in the output x_{II}^{*} of Problem II solvers. This comes from the consistency of the bits in $\varphi_{x_1,x_2}^{i,i+1}$ and $\varphi_{x_2,x_3}^{i+1,i+2}$ — in the correct ordering bit x_2 is shared. In the opposite order, x_3 and x_1 cannot always be shared, and the bits x_3x_1 may not be among preferred substrings of r. The right order is also expected based on the effect of thresholding. It guaranties that in all addresses of randomly selected lines of $f_{\mathscr{R}}$, if r_1r_2 is found in position $i, i+1$ in any of the lines the probability that r_3 follows is high or increases with increasing threshold. This in turn is reflected in high values of both the correlation $\varphi_{x_1,x_2}^{i,i+1}$ and $\varphi_{x_2,x_3}^{i+1,i+2}$. The thresholding argument penalizes the incorrect order of the correlation functions. Moreover, it penalizes strings with repeated subsings r_1r_2 not followed by r_3.

Therefore, we argue that Algorithm 1 which formally solves Problem II satisfies the minimum requirements of solver of Problem I. Thus, the output x_{II}^{*} of Algorithm 1 is also, with high probability, a solution of Problem I, as confirmed in the numerical simulations.

The arguments presented here are focused on providing an understanding based on simplified assumptions including a single rule of size three. The generalization for multiple different rules which is still confirmed by the numerical tests is in many cases a complicated problem. The rigorous proof can go along the same line of reasoning. However, calculating related probabilities rigorously can require solving counting satisfying assignments of logical formulas which are #P problems. Computing upper and lower bounds on the success probability remains an open problem.

4.1 Proof of Proposition 1—Larger moments of substrings of the rules

Proof. Formally,

$$\varphi_{x}^{i,i+1} = (\Phi_{x}^{i,i+1})^T F_N(r),$$

where

$$\Phi_{x}^{i,i+1} = (1_1 \otimes \ldots \otimes 1_{i-1} \otimes x \otimes 1_{i+2} \otimes \ldots \otimes 1_N)$$

and

$$F_N(r) = r \otimes 1_4 \otimes \ldots \otimes 1_N + 1_1 \otimes r \otimes 1_5 \otimes \ldots \otimes 1_N \ldots + 1_1 \otimes \ldots \otimes 1_{N-3} \otimes r.$$

i) For any position $2 \leq i \leq N-2$,

For a measurement bits x_1x_2 to match the rules at those positions, there are four possibilities shown in Table S2:

- If x_1x_2 matches either $r_1r_2r_1r_2r_2r_3 \rightarrow$ the energy increases by 2^{N-3}
- If x_1 matches $r_3 \rightarrow$ the energy increases by 2^{N-4}
- If x_2 matches $r_1 \rightarrow$ the energy increases by 2^{N-4}
- Any other case $\rightarrow$ the energy increases by 0

.	.	x_1	x_2	.	.
r_1	r_2	r_3	.	.	.
.	r_1	r_2	r_3	.	.
.	.	r_1	r_2	r_3	.
.	.	.	r_1	r_2	r_3

Supplementary Table S2. Configurations in which two bits x_1 and x_2 inside a string can mach the rule $r_1r_2r_3$.

x_1	x_2	.	.	.	.
r_1	r_2	r_3	.	.	.
.	r_1	r_2	r_3	.	.
.	.	r_1	r_2	r_3	.
.	.	.	r_1	r_2	r_3

Supplementary Table S3. Configurations in which two bits x_1 and x_2 at the beginning of a string can mach the rule $r_1r_2r_3$.

.	.	.	.	x_1	x_2
r_1	r_2	r_3	.	.	.
.	r_1	r_2	r_3	.	.
.	.	r_1	r_2	r_3	.
.	.	.	r_1	r_2	r_3

Supplementary Table S4. Configurations in which two bits x_1 and x_2 at the end of a string can mach the rule $r_1r_2r_3$.

So in total, any sketching pattern that are substring of a rule (r_1r_2 or r_2r_3) will bring 2^{N-3} energy. The other most energetic case is $x_1x_2 = r_3r_1$ which brings $2 \times 2^{N-4} = 2^{N-3}$ energy (same amount).

All other cases bring less energy. So we have indeed for all subtring r and any z $m_r^{i,i+1} \geq m_z^{i,i+1}$. If it's true for all, then it's also true for at least one.

ii) We study matching at the beginning and at the end

a) matching in position 1,2, (Table S3).

- If x_1x_2 match $r_1r_2 \rightarrow$ the energy increases by 2^{N-3}
- If x_2 matches $r_1 \rightarrow$ the energy increases by 2^{N-4}
- Any other case $\rightarrow$ the energy increases by 0

So there is one substring r , namely r_1r_2, that brings more energy than any other string.

The other substring all bring 0 energy for this position.

b) matching in position N-1,N (Table S4).

- If x_xm_2 match $r_2r_3 \rightarrow$ the energy increases by 2^{N-3}
- If x_1 matches $r_3 \rightarrow$ the energy increases by 2^{N-4}
- Any other case $\rightarrow$ the energy increases by 0

So there is one substring r , namely r_2r_3, that brings more energy than any other string.

The other substring all bring 0 energy for this position.

Part (iii) easily follows from (ii). □

For the problems with multiple rules the above reasoning should be completed by multiplying powers of two by appropriate factors proportional to values of rewards. The statements regarding the ordering of $< \varphi_x^{i,i+1} >$ for x belonging to different rules, or appearing in different rules, or not belonging to any of them is more complicated. However, similar arguments to Propositions 1 supported by numerical simulations allow us to conjecture that Algorithm 1 assigns higher values $\langle \varphi_x^{i,i+1} \rangle$ to pairs of bits x which are parts of the rules.