Electronic Colloquium on Computational Complexity, Report No. 54 (2024)

On the Power of Adaptivity for Function Inversion

Karthik Gajulapalli[*] Alexander Golovnev[†] Samuel King[‡]

Abstract

We study the problem of *function inversion with preprocessing* where, given a function $f : [N] \to [N]$ and a point y in its image, the goal is to find an x such that $f(x) = y$ using at most T oracle queries to f and S bits of preprocessed advice that depend on f.

The seminal work of Corrigan-Gibbs and Kogan [TCC 2019] initiated a line of research that shows many exciting connections between the non-adaptive setting of this problem and other areas of theoretical computer science. Specifically, they introduced a very weak class of algorithms (strongly non-adaptive) where the points queried by the oracle depend only on the inversion point y, and are independent of the answers to the previous queries and the S bits of advice. They showed that proving even mild lower bounds on strongly non-adaptive algorithms for function inversion would imply a breakthrough result in circuit complexity.

We prove that every strongly non-adaptive algorithm for function inversion (and even for its special case of permutation inversion) must have $ST = \Omega(N \log(N) \log(T))$. This gives the first improvement to the long-standing lower bound of $ST = \Omega(N \log N)$ due to Yao [STOC 90]. As a corollary, we conclude the first separation between strongly non-adaptive and adaptive algorithms for permutation inversion, where the adaptive algorithm by Hellman [TOIT 80] achieves the trade-off $ST = O(N \log N)$.

Additionally, we show equivalence between lower bounds for strongly non-adaptive data structures and the one-way communication complexity of certain partial functions. As an example, we recover our lower bound on function inversion in the communication complexity framework.

[*]Georgetown University. Email: kg816@georgetown.edu.
[†]Georgetown University. Email: alexgolovnev@gmail.com.
[‡]Georgetown University. Email: sik29@georgetown.edu.

ISSN 1433-8092

1 Introduction

We study the fundamental problem of *function inversion* where, given oracle access to a function $f : [N] \to [N]$ and a point y in the image of f, the goal is to find some x such that $f(x) = y$.

Clearly, to work for all functions, this would require any algorithm to make at least $N-1$ oracle calls. However, to make the problem more interesting, we consider a pair of algorithms $(\mathcal{P}, \mathcal{A})$ that work in two phases. In the first phase, using unlimited computational power, the *pre-processing* algorithm $\mathcal{P}$ is allowed to analyze the function f and write down S bits of *advice* $\sigma \in \{0,1\}^S$. Then in the second phase, the *online* algorithm $\mathcal{A}$, given inputs y and σ and at most T oracle queries to f, is required to output x such that $f(x) = y$. We informally refer to S and T as *space* and *time*, and the goal is to find algorithms $(\mathcal{P}, \mathcal{A})$ for function inversion that minimize S and T. Note that the problem is trivial when $S = N \log N$ or $T = N$. We are interested in the trade-offs between time and space when in between these two cases.

This model has received a lot of attention, especially for its applications to cryptanalysis [BS00, BSW01, Oec03, NS05], cryptography [Hel80, FN91, GT00, Wee05, Unr07, DTT10, DGK17, CDG18, CDGS18, GGPS23], circuit and data structure lower bounds [Yao90, CK19, DKKS21], algorithms [KP19, GGH+20], information theory [DKKS21], and most recently even meta-complexity [MP24, HIW24].

Function inversion and *permutation inversion*, a special case of function inversion where f is a permutation, were initially studied by Hellman [Hel80]. Hellman constructed an elegant algorithm that inverts any permutation when $ST = \Omega(N \log N)$. Later Yao [Yao90] showed that this algorithm was optimal by proving a tight lower bound of $ST = \Omega(N \log N)$ for permutation inversion (assuming $S = \Omega(\log N)$). For function inversion, Hellman gave an algorithm that inverts a random function when $S^2T = \widetilde{\Omega}(N^2)$.[1] Fiat and Naor [FN91] extended Hellman's construction, giving an algorithm that inverts any function when $S^3T = \widetilde{\Omega}(N^3)$.

One key facet of all the upper bounds mentioned above is that the queries made to f are highly adaptive; i.e., deciding which point $\mathcal{A}$ is going to query next depends on the inversion point y, the advice string σ, and the values of the points queried before. A long-standing open question has been to see if any of the upper bounds could be made non-adaptive. This question was extensively studied in [CK19], and they introduced the notion of strongly non-adaptive algorithms where the points queried by $\mathcal{A}$ are a fixed set depending only on the inversion point y. This makes the model much weaker compared to even the standard non-adaptive (weakly non-adaptive) setting where the fixed set of points queried by $\mathcal{A}$ is allowed to depend on the inversion point y and the advice string σ. Upper bounds for non-adaptive algorithms would be really useful, as they would lead to efficient parallelisation. Perhaps even more interestingly, lower bounds even in this very weak model would already imply circuit and communication lower bounds [CK19] and data structure lower bounds [CK19, GGH+20, DKKS21].

Indeed, as shown in [CK19], a lower bound of $S = \omega(N \log N / \log \log N)$ when $T = N^\varepsilon$ would imply a circuit lower bound against Boolean circuits of linear size and logarithmic depth, and thus resolve a long-standing open question due to Valiant [Val77]. A similar argument shows that even a lower bound of $S = \omega(N \log N / (\log \log (T / \log N)))$ for any $T = \Omega(\log N)$ would imply a super-linear circuit lower bound for series-parallel circuits [Val77, Cal08, Vio09].

The only known strongly non-adaptive algorithm is the trivial one where the pre-processing algorithm stores the value of f at $S/\log N$ points as advice, and the online algorithm queries the

[1] The notation $\widetilde{\Omega}(\cdot)$ and $\widetilde{O}(\cdot)$ suppresses factors polynomial in $\log N$.

remaining $N - S/\log N$ points, giving $S/\log N + T = N$. On the other hand, the best known lower bound for the non-adaptive setting is still $ST \geq \Omega(N \log N)$ obtained by Yao's compression argument [Yao90] that works even for adaptive algorithms. Hence, it might still be conceivable that the algorithm by Hellman can be made non-adaptive, which leads us to the natural question:

> *Are non-adaptive algorithms for permutation inversion as efficient as adaptive algorithms?*

1.1 Our Results

We answer this question in the negative by showing a lower bound of $ST = \Omega(N \log(N) \log(T))$ for any strongly non-adaptive algorithm for permutation inversion (and, thus, for the more general problem of function inversion).

Theorem 1. *Every strongly non-adaptive algorithm that solves permutation inversion with S bits of preprocessing and $T \leq N/5$ queries must have*

$$S = \Omega\left(\frac{N \log(N) \log(T)}{T}\right) .$$

Since permutation inversion can be solved adaptively when $ST = O(N \log N)$ [Hel80], Theorem 1 gives us the first separation between adaptive and strongly non-adaptive algorithms for permutation inversion for every super-constant T. (No separation is possible for constant T as in this case the problem is maximally hard, $S = \Omega(N \log N)$, even in the adaptive setting.)

We remark that the result of Theorem 1 comes tantalizingly close to the bound sufficient for a super-linear lower bound for series-parallel circuits. For example, for the case of $T = \Theta(\log(N) \log\log(N))$, Theorem 1 gives us $S = \Omega(N)$, whereas a bound of $S = \omega(N \log(N)/\log\log\log\log(N))$ would already imply a breakthrough in circuit complexity [CK19, Val77, Cal08].

The proof of Theorem 1 goes in two steps. First, we show that a compression argument can be used to get a lower bound on the amount of space required when, for a large enough set of inversion points, the union of all points queried by the online algorithm is small. In the following, we abuse notation when X is a set and define $\varphi(X) = \bigcup_{x \in X} \varphi(x)$.

Theorem 2. *For every $T \in \mathbb{N}$, $\varphi : [N] \to \binom{[N]}{T}$, and $X \subseteq [N]$ such that $|X| < N - |\varphi(X)|$, every strongly non-adaptive algorithm that solves permutation inversion with S bits of preprocessing and the query function φ must have*

$$S \geq |X| \log(N - |\varphi(X)| - |X|) .$$

We can now already recover Yao's lower bound for strongly non-apdative algorithms by Theorem 2. To see this, just consider the set $X = \{1, 2, ...N/(2T)\}$. Then $|\varphi(X)| \leq N/2$, and we get $S = \Omega((N \log N)/T)$.

This result also achieves optimal lower bounds for a specific subclass of query functions of interest: query functions which admit some $X \subseteq [N]$ of size $|X| = \Theta(N)$ with $|X| < N - |\varphi(X)|$. For example, take the query function φ which queries $\varphi(x) = (x, x+1, \ldots, x+T-1 \mod N)$ for each $x \in [N]$. When $T \leq N/4$, $X = \{1, 2, \ldots, N/4\}$ witnesses a lower bound of $S = \Omega(N \log N)$.

We note, however, that such query functions make up a small fraction of all possible query functions; random φ do not have this property.

To get an improvement over Yao's bound, our second step involves picking a large enough set X of size $\Theta((N \log T)/T)$ with a small enough $\varphi(X)$. We show the existence of such a set via the probabilistic method. We start by viewing φ as a left T-regular bipartite graph, and prove the following graph lemma, where $N(X)$ denotes the neighborhood of the set of vertices X.

Lemma 3. *Let $G = (L \sqcup R, E)$ be an undirected bipartite graph with $|L| = |R| = n$ and $|E| \leq dn$, where $d \leq n/5$. Then for large enough n, there exists a subset of vertices $X \subseteq L$, such that*

$$|X| \geq (n \log d)/(30d) \quad \textit{and}$$
$$|N(X)| \leq n - n/d^{4/5} .$$

It is not hard to see that Lemma 3 is tight for a random left T-regular bipartite graph.

1.2 Related Work

In the case of *adaptive algorithms*, the tight upper bound of $ST = O(N \log N)$ for permutation inversion is due to Hellman [Hel80]. Fiat and Naor [Hel80,FN91] gave upper bounds of $S^2T = \widetilde{O}(N^2)$ and $S^3T = \widetilde{O}(N^3)$ for inverting random and worst-case functions, respectively. It was recently observed [GGPS23] that the algorithm of Fiat and Naor for the worst-case function inversion can be extended to an upper bound of $TS^2 \max\{T, S\} = \widetilde{O}(N^3)$. De, Trevisan and Tulsiani [DTT10] extended [FN91] and gave better trade-offs when inverting on only ε-fraction of the inputs.

The best known *strongly non-adaptive algorithm* is just the trivial one which achieves the trade-off $S/\log N + T = N$. For the case of weakly non-adaptive algorithms, where the online algorithm gets to see the advice first, there is an algorithm that slightly outperforms the trivial when $S > N$ [GGPS23]. The preprocessing algorithm stores $\log(N/T)$ first bits of a preimage for each $y \in [N]$, and the online algorithm queries all of the remaining T options, which results in $S = N \log(N/T)$.

The best *lower bound* is due to Yao [Yao90], and it works for adaptive permutation inversion and thus also for function inversion. Moreover, since it works in the adaptive setting, it also trivially carries over to both the weakly and strongly non-adaptive settings. An alternate proof was given by Impagliazzo [Imp11], and [GT00,Wee05,DTT10,DGK17] extend the lower bound to the setting of randomized algorithms inverting on ε-fraction of inputs.

Even in the case of strongly non-adaptive algorithms, the best known lower bound is still Yao's. While no unconditional improvement to Yao's bound is known prior to this work, for some *restricted models* there are better bounds. Barkan, Biham, and Shamir [BBS06] give a lower bound of $S^2T = \Omega(N^2/\log N)$ for Hellman-type algorithms. Chawin, Haitner and Mazor [CHM20] prove an adaptive lower bound of $S + T \log N = \Omega(N)$ when the pre-processing algorithm $\mathcal{P}$ computes a linear function. In the case of weakly non-adaptive algorithms they show that if the online algorithm $\mathcal{A}$ is an affine function over the query points and advice then $S = \Omega(N)$. Moreover they generalize these bounds to prove lower bounds in the case when $\mathcal{A}$ is an affine decision tree. [GGPS23] gives tight bounds for guess-and-check algorithms for weakly non-adaptive function inversion. These bounds are however incomparable to strongly non-adaptive function inversion (strongly non-adaptive algorithms can't look at the advice, but can output a point they haven't queried). Finally, Dvořák, Koucký, Král and Slívová [DKKS21] prove a conditional lower bound of $T = \Omega(\log N/\log\log N)$, when $S = \varepsilon N \log N$ under the network coding conjecture.

In the *quantum setting*, [NABT15, HXY19, CLQ20, CGLQ20] give tight bounds even with quantum advice showing that Grover's search is optimal in the setting when $S = \widetilde{O}(\sqrt{N})$. Any improvement on these bounds would imply circuit lower bounds as shown in [CK19].

Structure of the Paper. In Section 2, we provide the necessary definitions. In Section 3, we prove the main results of this paper: Theorem 1, Theorem 2 and Lemma 3. We conclude this paper with a discussion on the equivalence between function inversion and the communication complexity of certain partial functions in Section 4.

2 Preliminaries

All logarithms are base 2. For a non-negative integer N, by $[N]$ we denote the set $\{1, \ldots, N\}$, and by Π_N we denote the set of all permutations of $[N]$. For an undirected graph $G = (V, E)$ and a subset of its vertices $S \subseteq V$, $N(S)$ denotes its neighborhood; i.e.,

$$N(S) := \{v \in V : \exists u \in S \text{ s.t. } \{u, v\} \in E\} .$$

We will use the following Chernoff bound (see e.g., [MU17]):

Lemma 2.1. *Let $X_1, \ldots, X_n$ be independent random variables taking values in $\{0, 1\}$ and X denote their sum with $\mu = \mathbb{E}[X]$. Then for $0 \leq \varepsilon \leq 1$,*

$$\Pr[X \leq (1 - \varepsilon)\mu] \leq \exp\left(\frac{-\varepsilon^2 \mu}{2}\right) .$$

2.1 The Permutation Inversion Problem

In the following definitions, let $(\mathcal{P}, \mathcal{A})$ be a pair of algorithms.

Definition 2.2. We say that

1. $(\mathcal{P}, \mathcal{A})$ uses S bits of pre-processing if for all inputs, the output of $\mathcal{P}$ has bit-length at most S.
2. $(\mathcal{P}, \mathcal{A})$ makes T queries if for all inputs, $\mathcal{A}^f$ makes at most T queries to f.

In this paper, we provide lower bounds on *permutation* inversion, a subproblem of function inversion. Hence, our lower bounds extend to function inversion as well.

Definition 2.3. We say that $(\mathcal{P}, \mathcal{A})$ solves the permutation inversion problem if for all $\pi \in \Pi_N$ and $y \in [N]$,

$$\mathcal{A}^\pi(\mathcal{P}(\pi), y) = \pi^{-1}(y) .$$

We call $\mathcal{P}$ the *preprocessing algorithm* and $\mathcal{A}$ the *online algorithm.*

We say that $(\mathcal{P}, \mathcal{A})$ is *strongly non-adaptive* if the T queries to π made by $\mathcal{A}^\pi$ depend only on y and not on the output of $\mathcal{P}(\pi)$ nor the results of previous queries. In such a case, we can define the *query function* of $\mathcal{A}^\pi$ to be $\varphi : [N] \to \binom{[N]}{T}$.

For any set $X \subseteq [N]$, we let $\varphi(X) = \bigcup_{x \in X} \varphi(x)$.

3 Non-Adaptive Function Inversion

In this section, we prove our improved lower bound on non-adaptive permutation inversion and hence function inversion. We start by showing a generic space bound (Theorem 2) that follows via a compression argument. This already allows us to recover Yao's lower bound. We next introduce a special graph lemma (Lemma 3) on sparse bipartite graphs that guarantees the existence of a large enough subset of vertices with a small neighborhood. Finally, combining these two together, we get our improved lower bound (Theorem 1).

Theorem 2. *For every $T \in \mathbb{N}$, $\varphi : [N] \to \binom{[N]}{T}$, and $X \subseteq [N]$ such that $|X| < N - |\varphi(X)|$, every strongly non-adaptive algorithm that solves permutation inversion with S bits of preprocessing and the query function φ must have*

$$S \geq |X| \log(N - |\varphi(X)| - |X|) \ .$$

Proof. Let $(\mathcal{P}, \mathcal{A})$ be a strongly non-adaptive algorithm for permutation inversion with query function φ that uses S bits of preprocessing. Let $X \subseteq [N]$ be such that $|X| < N - |\varphi(X)|$. For ease of notation, we define $\overline{\varphi(X)} := [N] \setminus \varphi(X)$. Because $|X| < N - |\varphi(X)| = |\overline{\varphi(X)}|$, there exist injective functions from $\varphi(X)$ to $[N] \setminus X$. Fix τ to be any such function, and let $P = \{\pi \in \Pi_N \ : \ \pi|_{\varphi(X)} = \tau\}$; in particular, for any two $\pi_1, \pi_2 \in P$, $\pi_1|_{\varphi(X)} = \pi_2|_{\varphi(X)}$. Then by construction, we have that for each $\pi \in P$, $\pi^{-1}(X) \subseteq \overline{\varphi(X)}$. We now pick a maximal subset $Q \subseteq P$ such that for every distinct $\pi_1, \pi_2 \in Q$, $\pi_1^{-1}|_X \neq \pi_2^{-1}|_X$. Thus,

$$|Q| = \binom{|\overline{\varphi(X)}|}{|X|} \cdot |X|! = \frac{|\overline{\varphi(X)}|!}{(|\overline{\varphi(X)}| - |X|)!} \geq (|\overline{\varphi(X)}| - |X|)^{|X|} \ .$$

Assume for the sake of contradiction that $2^S < |Q|$. Then by the pigeon hole principle, there exist two distinct $\pi_1, \pi_2 \in Q$ such that $\mathcal{P}(\pi_1) = \mathcal{P}(\pi_2)$. This implies that for all $i \in X$, $\mathcal{A}^{\pi_1}(\mathcal{P}(\pi_1), i) = \mathcal{A}^{\pi_2}(\mathcal{P}(\pi_2), i)$, since by construction $\pi_1|_{\varphi(X)} = \pi_2|_{\varphi(X)}$. This is a contradiction, as we know there exists some $i \in X$ for which $\pi_1^{-1}(i) \neq \pi_2^{-1}(i)$. Hence, $2^S \geq |Q| \geq (|\overline{\varphi(X)}| - |X|)^{|X|}$, and $S \geq |X| \log(|\overline{\varphi(X)}| - |X|)$. □

From this, we can get a lower bound on the size of the preprocessed advice for any query function φ which has a large X with small $\varphi(X)$. In the following lemma, we show that all query functions (viewed as bipartite graphs) admit such a subset X.

Lemma 3. *Let $G = (L \sqcup R, E)$ be an undirected bipartite graph with $|L| = |R| = n$ and $|E| \leq dn$, where $d \leq n/5$. Then for large enough n, there exists a subset of vertices $X \subseteq L$, such that*

$$|X| \geq (n \log d)/(30d) \quad \textit{and}$$
$$|N(X)| \leq n - n/d^{4/5} \ .$$

Proof. When $d \leq 32$, we can take $X \subseteq L$ simply to be the subset of $(n \log d)/(30d)$ vertices on the left with the smallest degrees. Then we have $|N(X)| \leq d \cdot |X| = (n \log d)/30 < n - n/d^{4/5}$. Thus, in the following, assume $d \geq 33$.

To prove the existence of such a subset X, we will first pick a random subset X of vertices from L. We will then bound the probability of X being small or having a large neighborhood away from 1. This will imply the existence of a set of X that satisfies both conditions of our lemma.

Let $p = (\log d)/(3d) \in (0,1)$, and let each vertex $a \in L$ be in X independently with probability p. We now compute the probability of our two bad events. First, to bound the probability of picking a small X, we can apply a Chernoff Bound (Lemma 2.1) to get that $\Pr\left[|X| \leq \frac{pn}{10}\right] \leq e^{-0.405pn} < e^{-pn/3}$.

Now, to get a bound on the probability that the size of the neighborhood $N(X)$ is close to n, let us first compute the expected size of $N(X)$:

$$\begin{aligned}\mathbb{E}\left[|N(X)|\right] &= \sum_{b \in R} \Pr[b \in N(X)] \\ &= n - \sum_{b \in R} \Pr[b \notin N(X)]\ . \qquad (1)\end{aligned}$$

The probability that $b \notin N(X)$ is the probability that none of the vertices $a \in N(b)$ were picked in X; i.e., $\Pr[b \notin N(X)] = (1-p)^{|N(b)|}$. Substituting into Equation (1) we get

$$\begin{aligned}\mathbb{E}\left[|N(X)|\right] &= n - \sum_{b \in R} (1-p)^{|N(b)|} \\ &\leq n - n\,(1-p)^{\frac{1}{n}\sum_b |N(b)|} \qquad (2)\\ &\leq n - n\,(1-p)^d\ , \qquad (3)\end{aligned}$$

where Equation (2) follows from the AM-GM inequality and Equation (3) follows from the fact that G has at most dn edges. Note that for $d > 1$ and $p = (\log d)/(3d)$, we have $0 < p < 1/4$. From this, we get for all $d \geq 33$

$$\begin{aligned}(1-p)^d &\geq e^{-d(p+p^2)} \\ &\geq e^{-d(p+\frac{p}{4})} \\ &= e^{-\frac{5pd}{4}} \\ &= d^{-\frac{5\log e}{12}} \\ &> \frac{2}{d^{4/5}}\ .\end{aligned}$$

Now we can conclude $\mathbb{E}\left[|N(X)|\right] < n - 2n/d^{4/5}$. With an upper bound on the expected size of $N(X)$, we apply Markov's inequality to get

$$\begin{aligned}\Pr\left[|N(X)| > n - \frac{n}{d^{4/5}}\right] &< \frac{n - 2n/d^{4/5}}{n - n/d^{4/5}} \\ &= 1 - \frac{1}{d^{4/5}(1 - 1/d^{4/5})} \\ &\leq 1 - d^{-4/5}\ .\end{aligned}$$

A union-bound over the probability of the two bad events happening gives

$$\Pr\left[|X| < \frac{pn}{10} \text{ or } |N(X)| > n - \frac{n}{d^{4/5}}\right] < 1 - d^{-4/5} + e^{-pn/3}\ . \qquad (4)$$

Now, because $d \leq n/5$, $d \leq (5\log e)n/36$ and hence $4/5 \leq (n \log e)/(9d)$. This gives us $d^{4/5} \leq d^{(n\log e)/(9d)} = e^{pn/3}$. From this, we can conclude that the probability in Equation (4) is strictly less than 1. This implies that there exists some $X \subseteq L$ with $|X| \geq pn/10$ and $|N(X)| \leq n - n/d^{4/5}$. $\square$

Now by combining Theorem 2 and Lemma 3, we get our main result.

Theorem 1. *Every strongly non-adaptive algorithm that solves permutation inversion with S bits of preprocessing and $T \leq N/5$ queries must have*

$$S = \Omega\left(\frac{N\log(N)\log(T)}{T}\right).$$

Proof. If $T < 3$, then take $T = 3$ by making more queries, and the following lower bound still holds. So, without loss of generality assume that $3 \leq T \leq N/5$.

Consider the bipartite graph of left-degree T defined by φ on $(L \sqcup R, E)$, where $L = \{\ell_1, \ldots, \ell_N\}$, $R = \{r_1, \ldots, r_N\}$, and for every $i \in [N]$ and $j \in \varphi(i)$ we have $\{\ell_i, r_j\} \in E$. Now let $X \subseteq [N]$ be the set guaranteed to exist by Lemma 3, so $|X| = \lceil N\log(T)/(30T)\rceil$ and $|\varphi(X)| \leq N - N/T^{4/5}$. Note that for all $T > 0$, $\log T < 15T^{1/5}$, so $N\log T/(15T) < N/T^{4/5}$. Thus, $|X| < |\overline{\varphi(X)}|$. Therefore, by Theorem 2, $S \geq |X|\log(|\overline{\varphi(X)}| - |X|)$. Note that

$$\begin{aligned}|\overline{\varphi(X)}| - |X| &\geq \frac{N}{T^{4/5}} - \frac{N\log(T)}{30T}\\ &= \frac{N}{T^{4/5}}\left(1 - \frac{\log(T)}{30T^{1/5}}\right)\\ &\geq \frac{N}{2T^{4/5}}\end{aligned}$$

for $T > 0$. Thus, we have

$$S \geq \frac{N\log(T)}{30T}\log\left(\frac{N}{2T^{4/5}}\right) \geq \frac{N\log(T)}{30T}\log\left(\frac{N^{1/5}}{2}\right) = \Omega\left(\frac{N\log(N)\log(T)}{T}\right). \qquad \square$$

4 Connections to Communication Complexity

In this section, we discuss an alternate approach to proving lower bounds for strongly non-adaptive function inversion via communication complexity. This approach generalizes to other strongly non-adaptive data structure problems.

Let $(\mathcal{P}, \mathcal{A})$ be a strongly non-adaptive algorithm for permutation inversion. We say that two permutations *conflict* under a query function φ if there exists an i such that $\pi^{-1}(i) \neq \tau^{-1}(i)$ and for every $j \in \varphi(i)$, $\pi(j) = \tau(j)$. Hence, to distinguish two conflicting permutations, we must have $\mathcal{P}(\pi) \neq \mathcal{P}(\tau)$. Now consider the following promise equality problem (PromEQ$_\varphi$).

Definition 4.1. For a given query function $\varphi : [N] \to \binom{[N]}{T}$, PromEQ$_\varphi$ is the following promise decision problem. Given two permutations $\pi, \tau \in \Pi_N$ such that either $\pi = \tau$ or π and τ conflict under ϕ, decide which one of the two conditions holds.

For a (promise) problem f, let $\mathrm{CC}^1(f)$ denote the one-way deterministic communication complexity of f. We then observe that $\mathrm{CC}^1(\mathrm{PromEQ}_\varphi)$ is the minimum amount of space needed for preprocessing to solve permutation inversion using the query function φ. On one hand, given a strongly non-adaptive algorithm $(\mathcal{P}, \mathcal{A})$, in the communication protocol Alice can send Bob $\mathcal{P}(\pi)$. To verify, Bob just checks if $\mathcal{P}(\pi) = \mathcal{P}(\tau)$. When $\pi = \tau$, equality is preserved. Otherwise, when π and τ conflict we are guaranteed to have $\mathcal{P}(\pi) \neq \mathcal{P}(\tau)$. On the other hand, assume that we have a one-way communication protocol for PromEQ_φ, and let σ_π be the message Alice sends to Bob when she receives π as input. We can then construct an algorithm $(\mathcal{P}, \mathcal{A})$ for permutation inversion, where $\mathcal{P}(\pi) = \sigma_\pi$. By the correctness of our communication protocol, we are guaranteed that there are no two conflicting permutations which share the same message σ_π. Hence, $\mathcal{A}$ can identify the inverse of the given point from σ_π and the points it queries. In particular, the question of understanding the complexity of strongly non-adaptive function inversion is equivalent to the following question.

Open Problem 4.2. *Find the minimum one-way deterministic communication complexity of* PromEQ_φ *among all* $\varphi\colon [N] \to \binom{[N]}{T}$,

$$\min_{\varphi\colon [N]\to\binom{[N]}{T}} \mathrm{CC}^1(\mathrm{PromEQ}_\varphi)\ .$$

Note that each PromEQ_φ problem is a "suproblem" of equality (the accept sets of PromEQ_φ and equality are identical, and the reject set of PromEQ_φ is a subset of the reject set of equality). Recall that while equality admits an efficient randomized communication protocol, it has maximum deterministic communication complexity. Thus, to prove a polynomial lower bound for PromEQ_φ via a reduction from some known problem, the reduction must be deterministic. Moreover, the problem we reduce from must admit an efficient randomized communication protocol, while being sufficiently hard for any deterministic protocol.

Recovering our improved bound To illustrate this approach, we now demonstrate how our main result (Theorem 1) can be obtained in this communication complexity framework. Given the discussion above, Theorem 1 is equivalent to proving a lower bound of $\mathrm{CC}^1(\mathrm{PromEQ}_\varphi) = \Omega(N\log(N)\log(T)/T)$ for all $\varphi\colon [N] \to \binom{[N]}{T}$. In order to do this, we first introduce an auxiliary promise problem $\mathrm{PermEQ}_{k,\Sigma}$ which checks equality of k-permutations over an alphabet Σ.

Definition 4.3. For a given alphabet Σ and $k \le |\Sigma|$, $\mathrm{PermEQ}_{k,\Sigma}$ is the following promise decision problem. Given two k-permutations of Σ (strings of length k with distinct characters), decide if they are equal or not.

In order to get a lower bound on $\mathrm{CC}^1(\mathrm{PromEQ}_\varphi)$, we reduce $\mathrm{PermEQ}_{k,\Sigma}$ to PromEQ_φ; then known lower bounds on $\mathrm{CC}^1(\mathrm{PermEQ}_{k,\Sigma})$ extend to $\mathrm{CC}^1(\mathrm{PromEQ}_\varphi)$. The following is a sketch of this reduction: Given some φ, we use Lemma 3 to get a large $X \subseteq [N]$ with small $\varphi(X)$. Then we take $\Sigma = \overline{\varphi(X)}$ and $k = |X|$. Now given α, a k-permutation of Σ, we construct π_α, a permutation of $[N]$, where π_α maps α to X and $\varphi(X)$ to $[N] \setminus X$. In particular, $\pi_\alpha|_{\varphi(X)}$ does not depend on α. Then it is not hard to see that for distinct k-permutations α and β of Σ, π_α and π_β conflict. Thus, in the reduction from $\mathrm{PermEQ}_{k,\Sigma}$ to PromEQ_φ, Alice and Bob first construct π_α and π_β from their inputs α and β and then run the protocol for PromEQ_φ. The lower bound then follows from the known lower bound of $\mathrm{CC}^1(\mathrm{PermEQ}_{k,\Sigma}) \ge \Omega(k \log|\Sigma|)$.

Acknowledgements

We would like to thank Spencer Peters for fruitful discussions on this topic. This research is supported by the National Science Foundation CAREER award (grant CCF-2338730).

References

[BBS06] Elad Barkan, Eli Biham, and Adi Shamir. Rigorous bounds on cryptanalytic time/memory tradeoffs. In *CRYPTO*, 2006. 4

[BS00] Alex Biryukov and Adi Shamir. Cryptanalytic time/memory/data tradeoffs for stream ciphers. In *ASIACRYPT*, 2000. 2

[BSW01] Alex Biryukov, Adi Shamir, and David Wagner. Real time cryptanalysis of A5/1 on a PC. In *FSE*, 2001. 2

[Cal08] Chris Calabro. A lower bound on the size of series-parallel graphs dense in long paths. In *ECCC*, 2008. 2, 3

[CDG18] Sandro Coretti, Yevgeniy Dodis, and Siyao Guo. Non-uniform bounds in the random-permutation, ideal-cipher, and generic-group models. In *CRYPTO*, 2018. 2

[CDGS18] Sandro Coretti, Yevgeniy Dodis, Siyao Guo, and John Steinberger. Random oracles and non-uniformity. In *Eurocrypt*, 2018. 2

[CGLQ20] Kai-Min Chung, Siyao Guo, Qipeng Liu, and Luowen Qian. Tight quantum time-space tradeoffs for function inversion. In *FOCS*, 2020. 5

[CHM20] Dror Chawin, Iftach Haitner, and Noam Mazor. Lower bounds on the time/memory tradeoff of function inversion. In *TCC*, 2020. 4

[CK19] Henry Corrigan-Gibbs and Dmitry Kogan. The function-inversion problem: Barriers and opportunities. In *TCC*, 2019. 2, 3, 5

[CLQ20] Kai-Min Chung, Tai-Ning Liao, and Luowen Qian. Lower bounds for function inversion with quantum advice. In *ITC*, 2020. 5

[DGK17] Yevgeniy Dodis, Siyao Guo, and Jonathan Katz. Fixing cracks in the concrete: Random oracles with auxiliary input, revisited. In *EUROCRYPT*, 2017. 2, 4

[DKKS21] Pavel Dvořák, Michal Koucký, Karel Král, and Veronika Slívová. Data structures lower bounds and popular conjectures. In *ESA*, 2021. 2, 4

[DTT10] Anindya De, Luca Trevisan, and Madhur Tulsiani. Time space tradeoffs for attacks against one-way functions and PRGs. In *CRYPTO*, 2010. 2, 4

[FN91] Amos Fiat and Moni Naor. Rigorous time/space tradeoffs for inverting functions. In *STOC*, 1991. 2, 4

[GGH$^+$20] Alexander Golovnev, Siyao Guo, Thibaut Horel, Sunoo Park, and Vinod Vaikuntanathan. Data structures meet cryptography: 3SUM with preprocessing. In *STOC*, 2020. 2

[GGPS23] Alexander Golovnev, Siyao Guo, Spencer Peters, and Noah Stephens-Davidowitz. Revisiting time-space tradeoffs for function inversion. In *CRYPTO*, 2023. 2, 4

[GT00] Rosario Gennaro and Luca Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *FOCS*, 2000. 2, 4

[Hel80] Martin Hellman. A cryptanalytic time-memory trade-off. *IEEE Trans. Inf. Theory*, 26(4):401–406, 1980. 2, 3, 4

[HIW24] Shuichi Hirahara, Rahul Ilango, and Ryan Williams. Beating brute force for compression problems. In *STOC*, 2024. 2

[HXY19] Minki Hhan, Keita Xagawa, and Takashi Yamakawa. Quantum random oracle model with auxiliary input. In *ASIACRYPT*, 2019. 5

[Imp11] Russell Impagliazzo. Relativized separations of worst-case and average-case complexities for NP. In *CCC*, 2011. 4

[KP19] Tsvi Kopelowitz and Ely Porat. The strong 3SUM-INDEXING conjecture is false. *arXiv:1907.11206*, 2019. 2

[MP24] Noam Mazor and Rafael Pass. The non-uniform perebor conjecture for time-bounded Kolmogorov complexity is false. In *ITCS*, 2024. 2

[MU17] Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis.* Cambridge university press, 2017. 5

[NABT15] Aran Nayebi, Scott Aaronson, Aleksandrs Belovs, and Luca Trevisan. Quantum lower bound for inverting a permutation with advice. *Quantum Inf. Comput.*, 15(11-12):901–913, 2015. 5

[NS05] Arvind Narayanan and Vitaly Shmatikov. Fast dictionary attacks on passwords using time-space tradeoff. In *CCS*, 2005. 2

[Oec03] Philippe Oechslin. Making a faster cryptanalytic time-memory trade-off. In *CRYPTO*, 2003. 2

[Unr07] Dominique Unruh. Random oracles and auxiliary input. In *CRYPTO*, 2007. 2

[Val77] Leslie G. Valiant. Graph-theoretic arguments in low-level complexity. In *MFCS*, 1977. 2, 3

[Vio09] Emanuele Viola. On the power of small-depth computation. *Found. Trends Theor. Comput. Sci.*, 5(1):1–72, 2009. 2

[Wee05] Hoeteck Wee. On obfuscating point functions. In *STOC*, 2005. 2, 4

[Yao90] Andrew Chi-Chih Yao. Coherent functions and program checkers. In *STOC*, 1990. 2, 3, 4