# Data Structures Lower Bounds and Popular Conjectures

## Pavel Dvořák ✉
Charles University, Prague, Czech Republic

## Michal Koucký ✉
Charles University, Prague, Czech Republic

## Karel Král ✉
Charles University, Prague, Czech Republic

## Veronika Slívová ✉
Charles University, Prague, Czech Republic

—— **Abstract** ——————————————————————————————————

In this paper, we investigate the relative power of several conjectures that attracted recently lot of interest. We establish a connection between the Network Coding Conjecture ($NCC$) of Li and Li [24] and several data structure like problems such as non-adaptive function inversion of Hellman [18] and the well-studied problem of polynomial evaluation and interpolation. In turn these data structure problems imply super-linear circuit lower bounds for explicit functions such as integer sorting and multi-point polynomial evaluation.

## 1 Introduction

One of the central problems in theoretical computer science is proving lower bounds in various models of computation such as circuits and data structures. Proving super-linear size lower bounds for circuits even when their depth is restricted is rather elusive. Similarly, proving polynomial lower bounds on query time for certain static data structure problems seems out of reach. To deal with this situation researchers developed various conjectures which if true would imply the sought after lower bounds. In this paper, we investigate the relative power of some of those conjectures. We establish a connection between the Network Coding Conjecture ($NCC$) of Li and Li [24] used recently to prove various lower bounds such as lower bounds on circuit size counting multiplication [3] and a number of IO operations for external memory sorting [12].

Another problem researchers looked at is a certain data structure type problem for function inversion [18] which is popular in cryptography. Corrigan-Gibbs and Kogan [9] observed that lower bounds for the function inversion problem imply lower bounds for logarithmic depth circuits. In this paper we establish new connections between the problems, and identify some interesting instances. Building on the work of Afshani et al. [3] we show that the Network Coding Conjecture implies certain weak lower bounds for the inversion data structure problems. That in turn implies the same type of circuit lower bounds as given

by Corrigan-Gibbs and Kogan [9]. We show that similar results apply to a host of other data structure problems such as the well-studied polynomial evaluation problem or the Finite Field Fourier transform problem. Corrigan-Gibbs and Kogan [9] gave their circuit lower bound for certain apriori undetermined function. We establish the same circuit lower bounds for sorting integers which is a very explicit function. Similarly, we establish a connection between data structure for polynomial evaluation and circuits for multi-point polynomial evaluation. Our results sharpen and generalize the picture emerging in the literature.

The data structure problems we consider in this paper are for static, non-adaptive, systematic data structure problems, a very restricted class of data structures for which lower bounds should perhaps be easier to obtain. Data structure problems we consider have the following structure: Given the input data described by $N$ bits, create a data structure of size $s$. Then we receive a single query from a set of permissible queries and we are supposed to answer the query while non-adaptively inspecting at most $t$ locations in the data structure and in the original data. The non-adaptivity means that the inspected locations are chosen only based on the query being answered but not on the content of the inspected memory. We show that when $s \geq \omega\big(N \log N / \log \log N\big)$, polynomial lower bounds on $t$ for certain problems would imply super-linear lower bounds on log-depth circuits for computing sorting, multi-point polynomial evaluation, and other problems.

We show that logarithmic lower bounds on $t$ for the data structures can be derived from the Network Coding Conjecture even in the more generous setting of $s \geq \varepsilon N$ and when inspecting locations in the data structure is for free. This matches the lower bounds of Afshani [3] for certain circuit parameters derived from the Network Coding Conjecture. One can recover the same type of result they showed from our connection between the Network Coding Conjecture, data structure lower bounds, and circuit lower bounds.

In this regard, the Network Coding Conjecture seems the strongest among the conjectures, which is the hardest to prove. One would hope that for the strongly restricted data structure problems, obtaining the required lower bounds should be within our reach.

**Organization.** This paper is organized as follows. In the next section we review the data structure problems we consider. Then we provide a precise definition of Network Coding Conjecture in Section 3. Section 4 contains the statement of our main results. In Sections 5 and 6 we prove our main result for the function inversion and the polynomial problems. In Section 7 we discus the connection between data structure and circuit lower bounds for explicit functions.

## 2 Data Structure Problems

In this paper, we study lower bounds on *systematic data structures* for various problems – function inversion, polynomial evaluation, and polynomial interpolation. We are given an input $I = \{x_0, \ldots, x_{n-1}\}$, where each $x_i \in [n] = \{0, \ldots, n-1\}$ or each $x_i$ is an element of some field $\mathbb{F}$. First, a data structure algorithm can preprocess $I$ to produce an advice string $\mathbf{a}_I$ of $s$ bits (we refer to the parameter $s$ as *space* of the data structure $\mathcal{D}$). Then, we are given a query $q$ and the data structure should produce a correct answer (what is a correct answer depends on the problem). To answer a query $q$, the data structure $\mathcal{D}$ has access to the whole advice string $\mathbf{a}_I$ and can make $t$ queries to the input $I$, i.e., read at most $t$ elements from $I$. We refer to the parameter $t$ as query time of the data structure.

We consider non-uniform data structures as we want to provide connections between data structures and non-uniform circuits. Formally, a non-uniform systematic data structure $\mathcal{D}_n$ for an input $I = \{x_0, \ldots, x_{n-1}\}$ is a pair of algorithms $(\mathcal{P}_n, \mathcal{Q}_n)$ with oracle access to $I$. The

algorithm $\mathcal{P}_n$ produces the advice string $\mathbf{a}_I \in \{0,1\}^s$. The algorithm $\mathcal{Q}_n$ with inputs $\mathbf{a}_I$ and a query $q$ outputs a correct answer to the query $q$ with at most $t$ oracle queries to $I$. The algorithms $\mathcal{P}_n$ and $\mathcal{Q}_n$ can differ for each $n \in \mathbb{N}$.

## 2.1 Function Inversion

In the function inversion problem, we are given a function $f : [n] \to [n]$ and a point $y \in [n]$ and we want to find $x \in [n]$ such that $f(x) = y$. This is a central problem in cryptography as many cryptographic primitives rely on the existence of a function that is hard to invert. To sum up we are interested in the following problem.

| **Function Inversion** | |
|---|---|
| *Input:* | A function $f : [n] \to [n]$ as an oracle. |
| *Preprocessing:* | Using $f$, prepare an advice string $\mathbf{a}_f \in \{0,1\}^s$. |
| *Query:* | Point $y \in [n]$. |
| *Answer:* | Compute the value $f^{-1}(y)$, with a full access to $\mathbf{a}_f$ and using at most $t$ queries to the oracle for $f$. |

We want to design an efficient data structure, i.e., make $s$ and $t$ as small as possible. There are two trivial solutions. The first one is that the whole function $f^{-1}$ is stored in the advice string $\mathbf{a}_f$, thus $s = O(n \log n)$ and $t = 0$. The second one is that the whole function $f$ is queried during answering a query $y \in [n]$, thus $t = O(n)$ and $s = 0$. Note that the space $s$ of the data structure is the length of the advice string $\mathbf{a}_f$ in bits, but with one oracle-query $x_i$ the data structure reads the whole $f(x_i)$, thus with $n$ oracle-queries we read the whole description of $f$, i.e., $n \log n$ bits.

The question is whether we can design a data structure with $s, t \leq o(n)$. Hellman [18] gave the first non-trivial solution and introduced a randomized systematic data structure which inverts a function with a constant probability (over the uniform choice of the function $f$ and the query $y \in [n]$) and $s = O\left(n^{2/3} \log n\right)$ and $t = O\left(n^{2/3} \log n\right)$. Fiat and Naor [13] improved the result and introduced a data structure that inverts any function at any point, however with a slightly worse trade-off: $s^3 t = O\left(n^3 \log n\right)$. Hellman [18] also introduced a more efficient data structure for inverting a permutation – it inverts any permutation at any point and $st = O(n \log n)$. Thus, it seems that inverting a permutation is an easier problem than inverting an arbitrary function.

In this paper, we are interested in lower bounds for the inversion problem. Yao [35] gave a lower bound that any systematic data structure for the inversion problem must have $st \geq \Omega(n \log n)$, however, the lower bound is applicable only if $t \leq O(\sqrt{n})$. Since then, only slight progress was made. De et al. [10] improved the lower bound of Yao [35] that it is applicable for the full range of $t$. Abusalah et al. [1] improved the trade-off, that for any $k$ it must hold that $s^k t \geq \Omega\left(n^k\right)$. Seemingly, their result contradicts Hellman's trade-off $\left(s = t = O\left(n^{2/3} \log n\right)\right)$ as it implies $s = t \geq n^{k/k+1}$ for any $k$. However, for Hellman's attack [18] we need that the function can be efficiently evaluated and the functions introduced by Abusalah et al. [1] cannot be efficiently evaluated. There is also a series of papers [16, 29, 11, 8] which study how the probability of successful inversion depends on the parameters $s$ and $t$. However, none of these results yields a better lower bound than $st \geq \Omega(n \log n)$. Hellman's trade-off is still the best known upper bound trade-off for the inversion problem. Thus, there is still a substantial gap between the lower and upper bounds.

Another caveat of all known data structures for the inversion is that they heavily use adaptivity during answering queries $y \in [n]$. I.e., queries to the oracle depend on the advice string $\mathbf{a}$ and answers to the oracle queries which have been already made. We are interested

in non-adaptive data structures. We say a systematic data structure is *non-adaptive* if all oracle queries depend only on the query $y \in [n]$.

As non-adaptive data structures are weaker than adaptive ones, there is a hope that for non-adaptive data structures we could prove stronger lower bounds. Moreover, the non-adaptive data structure corresponds to circuits computation [30, 31, 33, 9]. Thus, we can derive a circuit lower bound from a strong lower bound for a non-adaptive data structure. Non-adaptive data structures were considered by Corrigan-Gibbs and Kogan [9]. They proved that improvement by a polynomial factor of Yao's lower bound [35] for non-adaptive data structures would imply the existence of a function $F : \{0,1\}^N \to \{0,1\}^N$ for $N = n \log n$ that cannot be computed by a linear-size and logarithmic-depth circuit. More formally, they prove that if a function $f : [n] \to [n]$ cannot be inverted by a non-adaptive data structure of space $O\left(n \log n / \log \log n\right)$ and query time $O(n^\varepsilon)$ for some $\varepsilon > 0$ then there exists a function $F : \{0,1\}^N \to \{0,1\}^N$ that cannot be computed by any circuit of size $O(N)$ and depth $O(\log N)$. They interpret $r \in \{0,1\}^N$ as $n$ numbers in $[n]$, i.e, $r = (r_1, \ldots, r_n) \in \{0,1\}^N$ where each $r_i \in [n]$. The function $F$ is defined as $F(y) = F(y_1, \ldots, y_n) = \left(f^{-1}(y_1), \ldots, f^{-1}(y_n)\right)$ where $f^{-1}(y_i) = \min\{x \in [n] \mid f(x) = y\}$ and $\min \emptyset = 0$. Informally, if the function $f$ is hard to invert at some points, then it is hard to invert at all points together. Moreover, they showed equivalence between function inversion and substring search. A data structure for the function inversion of space $s$ and query time $t$ yields a data structure of space $O(s \log s)$ and query time $O(t \log t)$ for finding pattern of length $O(\log n)$ in a binary text of length $O(n \log n)$ and vice versa – an efficient data structure for the substring search would yield an efficient data structure for the function inversion. Compared to results of Corrigan-Gibbs and Kogan [9], we provide an explicit function (sorting integers) which will require large circuits if any of the functions $f$ is hard to invert.

Another connection between data structures and circuits was made by Viola [34] who considered constant depth circuits with arbitrary gates.

## 2.2   Evaluation and Interpolation of Polynomials

In this section, we describe two natural problems connected to polynomials. We consider our problems over a finite field $\mathbb{F}$ to avoid issues with encoding reals.

| **Polynomial Evaluation over** $\mathbb{F}$ | |
| --- | --- |
| *Input:* | Coefficients of a polynomial $p \in \mathbb{F}[x]$: $\alpha_0, \ldots, \alpha_{n-1} \in \mathbb{F}$ (i.e., $p(x) = \sum_{i \in [n]} \alpha_i x^i$) |
| *Preprocessing:* | Using the input, prepare an advice string $\mathbf{a}_p \in \{0,1\}^s$. |
| *Query:* | A number $x \in \mathbb{F}$. |
| *Answer:* | Compute the value $p(x)$, with a full access to $\mathbf{a}_p$ and using at most $t$ queries to the coefficients of $p$. |

| **Polynomial Interpolation over** $\mathbb{F}$ | |
| --- | --- |
| *Input:* | Point-value pairs of a polynomial $p \in \mathbb{F}[x]$ of degree at most $n-1$: $\left(x_0, p(x_0)\right), \ldots, \left(x_{n-1}, p(x_{n-1})\right) \in \mathbb{F} \times \mathbb{F}$ where $x_i \neq x_j$ for any two indices $i \neq j$ |
| *Preprocessing:* | Using the input, prepare an advice string $\mathbf{a}_p \in \{0,1\}^s$. |
| *Query:* | An index $j \in [n]$. |
| *Answer:* | Compute $j$-th coefficient of the polynomial $p$, i.e., the coefficient of $x^j$ in $p$, with a full access to $\mathbf{a}_p$ and using at most $t$ queries to the oracle for point-value pairs. |

In the paper we often use a version of polynomial interpolation where the points $x_0, x_1, \ldots, x_{n-1}$ are fixed in advance and the input consists just of $p(x_0), p(x_1), \ldots, p(x_{n-1})$. Since we are interested in lower bounds, this makes our results slightly stronger.

Let $\mathbb{F} = \mathrm{GF}(p^k)$ denote the Galois Field of $p^k$ elements. Let $n$ be a divisor of $p^k - 1$. It is a well-known fact that for any finite field $\mathbb{F}$ its multiplicative group $\mathbb{F}^*$ is cyclic (see e.g. Serre [27]). Thus, there is an element $\sigma \in \mathbb{F}$ of order $n$ in the multiplicative group $\mathbb{F}^*$ (that is an element $\sigma$ such that $\sigma^n = 1$ and for each $1 \leq j < n$, $\sigma^j \neq 1$). In other words, $\sigma$ is our choice of primitive $n$-th root of unity. Pollard [26] defines the *Finite Field Fourier transform* (FFFT) (with respect to $\sigma$) as a linear function $\mathrm{FFFT}_{n,\sigma} \colon \mathbb{F}^n \to \mathbb{F}^n$ which satisfies:

$\mathrm{FFFT}_{n,\sigma}(\alpha_0, \ldots, \alpha_{n-1}) = (\beta_0, \ldots, \beta_{n-1})$ where
$$\beta_i = \sum_{j \in [n]} \alpha_j \sigma^{ij} \qquad \text{for any } i \in [n]$$

The inversion $\mathrm{FFFT}_{n,\sigma}^{-1}$ is given by:

$\mathrm{FFFT}_{n,\sigma}^{-1}(\beta_0, \ldots, \beta_{n-1}) = (\alpha_0, \ldots, \alpha_{n-1})$ where
$$\alpha_i = \frac{1}{n} \sum_{j \in [n]} \beta_j \sigma^{-ij} \qquad \text{for any } i \in [n]$$

Note that if we work over a finite field $\mathbb{F}$, our $n$ might not be an element of $\mathbb{F}$. For simplicity we slightly abuse the notation and use $\frac{1}{n} = \left( \sum_{i=1}^{n} 1 \right)^{-1}$. In our theorems we always set $n$ to be a divisor of $|\mathbb{F}| - 1 = p^k - 1$ thus $n$ modulo $p$ is non-zero and the inverse exists. Observe, that $\mathrm{FFFT}_{n,\sigma}^{-1} = \frac{1}{n} \mathrm{FFFT}_{n,\sigma^{-1}}$. Hence, FFFT is the finite field analog of Discrete Fourier transform (DFT) which works over complex numbers.

The FFT algorithm by Cooley and Tukey [7] can be used for the case of finite fields as well (as observed by Pollard [26]) to get an algorithm using $O(n \log n)$ field operations (addition or multiplication of two numbers). Thus we can compute $\mathrm{FFFT}_{n,\sigma}$ and its inverse in $O(n \log n)$ field operations.

It is easy to see that $\mathrm{FFFT}_{n,\sigma}$ is actually evaluation of a polynomial in multiple special points (specifically in $\sigma^0, \ldots, \sigma^{n-1}$). We can also see that it is a special case of interpolation by a polynomial in multiple special points since $\mathrm{FFFT}_{n,\sigma}^{-1} = \frac{1}{n} \mathrm{FFFT}_{n,\sigma^{-1}}$. We provide an NCC-based lower bound for data structures computing the polynomial evaluation. However, we use the data structure only for evaluating a polynomial in powers of a primitive root of unity. Thus, the same proof yields a lower bound for data structures computing the polynomial interpolation.

There is a great interest in data structures for polynomial evaluation in a cell probe model. In this model, some representation of a polynomial $p = \sum_{i \in [n]} \alpha_i x^i \in \mathbb{F}[x]$ is stored in a table $\mathcal{T}$ of $s_{\mathsf{cell}}$ cells, each of $w$ bits. Usually, $w$ is set to $O\big(\log |\mathbb{F}|\big)$, that we can store an element of $\mathbb{F}$ in a single cell. On a query $x \in \mathbb{F}$ the data structure should output $p(x)$ making at most $t_{\mathsf{cell}}$ probes to the table $\mathcal{T}$. A difference between data structures in the cell probe model and systematic data structures is that a data structure in the cell probe model is charged for any probe to the table $\mathcal{T}$ but a systematic data structure is charged only for queries to the input (the coefficients $\alpha_i$), reading from the advice string $\mathbf{a}_p$ is for free. Note that, the coefficients $\alpha_i$ of $p$ do not have to be even stored in the table $\mathcal{T}$. There are again two trivial solutions. The first one is that we store a value $p(x)$ for each $x \in \mathbb{F}$ and on a query $x \in \mathbb{F}$ we probe just one cell. Thus, we would get $t_{\mathsf{cell}} = 1$ and $s_{\mathsf{cell}} = |\mathbb{F}|$ (we assume that we can store an element of $\mathbb{F}$ in a single cell). The second one is that we store the coefficients of $p$ and on a query $x \in \mathbb{F}$ we probe all cells and compute the value $p(x)$. Thus, we would get $t_{\mathsf{cell}} = s_{\mathsf{cell}} = n$.

Let $k = \log |\mathbb{F}|$. Kedlaya and Umans [21] provided a data structure for the polynomial evaluation that uses space $n^{1+\varepsilon} \cdot k^{1+o(1)}$ and query time $\log^{O(1)} n \cdot k^{1+o(1)}$. Note that, $n \cdot k$ is the size of the input and $k$ is the size of the output.

The first lower bound for the cell probe model was given by Miltersen [25]. He proved that for any cell probe data structure for the polynomial evaluation it must hold that $t_{\text{cell}} \geq \Omega\big(k/\log s_{\text{cell}}\big)$. This was improved by Larsen [22] to $t_{\text{cell}} \geq \Omega\big(k/\log(s_{\text{cell}}w/nk)\big)$, that gives $t_{\text{cell}} \geq \Omega(k)$ if the data structure uses linear space $s_{\text{cell}} \cdot w = O(n \cdot k)$. However, the size of $\mathbb{F}$ has to be super-linear, i.e., $|\mathbb{F}| \geq n^{1+\Omega(1)}$. Data structures in a bit probe model were studied by Gál and Miltersen [14]. The bit probe model is the same as the cell probe model but each cell contains only a single bit, i.e., $w = 1$. They studied succinct data structures that are data structures such that $s_{\text{cell}} = (n + r) \cdot k$ for $r < o(n)$. Thus, the succinct data structures are related to systematic data structures but still, the succinct data structures are charged for any probe (as any other data structure in the cell probe model). Note that a succinct data structure stores only a few more bits than it is needed due to information-theoretic requirement. Gál and Miltersen [14] showed that for any succinct data structure in the bit probe model it holds that $r \cdot t_{\text{cell}} \geq \Omega(n \cdot k)$. We are not aware of any lower bound for systematic data structures for the polynomial evaluation.

Larsen et al. [23] also gives a log-squared lower bound for dynamic data structures in the cell probe model. Dynamic data structures also support updates of the polynomial $p$.

There is a great interest in algorithmic questions about the polynomial interpolation such as how fast we can interpolate polynomials [15, 5, 17], how many queries we need to interpolate a polynomial if it is given by oracle [6, 19], how to compute the interpolation in a numerically stable way over infinite fields [28] and many others. However, we are not aware of any results about data structures for the interpolation, i.e., when the interpolation algorithm has an access to some precomputed advice.

## 3    Network Coding

We prove our conditional lower bounds based on the Network Coding Conjecture. In network coding, we are interested in how much information we can send through a given network. A *network* consists of a graph $G = (V, E)$, positive capacities of edges $c : E \to \mathbb{R}^+$ and $k$ pairs of vertices $(s_0, t_0), \ldots, (s_{k-1}, t_{k-1})$. We say a network $R = \big(G, c, (s_i, t_i)_{i \in [k]}\big)$ is *undirected* or *directed (acyclic)* if the graph $G$ is undirected or directed (acyclic). We say a network is *uniform* if the capacities of all edges in the network equal to some $q \in \mathbb{R}^+$ and we denote such network as $\big(G, q, (s_i, t_i)_{i \in [k]}\big)$.

A goal of a coding scheme for directed acyclic network $R = \big(G, c, (s_i, t_i)_{i \in [k]}\big)$ is that at each target $t_i$ it will be possible to reconstruct an input message $w_i$ which was generated at the source $s_i$. The coding scheme specifies messages sent from each vertex along the outgoing edges as a function of received messages. Moreover, the length of the messages sent along the edges have to respect the edge capacities.

More formally, each source $s_i$ of a network receives an input message $w_i$ sampled (independently of the messages for the other sources) from the uniform distribution $\mathbf{W}_i$ on a set $W_i$. Without loss of generality we can assume that each source $s_i$ has an in-degree 0 (otherwise we can add a vertex $s_i'$ and an edge $(s_i', s_i)$ and replace $s_i$ by $s_i'$). There is an alphabet $\Sigma_e$ for each edge $e \in E(G)$. For each source $s_i$ and each outgoing edge $e = (s_i, u)$ there is a function $f_{s_i, e} : W_i \to \Sigma_e$ which specifies the message sent along the edge $e$ as a function of the received input message $w_i \in W_i$. For each non-source vertex $v \in V, v \neq s_i$ and each outgoing edge $e = (v, u)$ there is a similar function $f_{v,e} : \prod_{e' = (u', v)} \Sigma_{e'} \to \Sigma_e$ which

specifies the message sent along the edge $e$ as a function of the messages sent to $v$ along the edges incoming to $v$. Finally, each target $t_i$ has a decoding function $d_i : \prod_{e'=(u',t_i)} \Sigma_{e'} \to W_i$. The coding scheme is executed as follows:

1. Each source $s_i$ receives an input message $w_i \in W_i$. Along each edge $e = (s_i, u)$ a message $f_{s_i,e}(w_i)$ is sent.

2. When a vertex $v$ receives all messages $m_1, \ldots, m_a$ along all incoming edges $(u', v)$ it sends along each outgoing edge $e = (v, u)$ a message $f_{v,e}(m_1, \ldots, m_a)$. As the graph $G$ is acyclic, this procedure is well-defined and each vertex of non-zero out-degree will eventually send its messages along its outgoing edges.

3. At the end, each target $t_i$ computes a string $\tilde{w}_i = d_i(m'_1, \ldots, m'_b)$ where $m'_j$ denotes the received messages along the incoming edges $(u', t_i)$. We say the encoding scheme is *correct* if $\tilde{w}_i = w_i$ for all $i \in [k]$ and any input messages $w_0, \ldots, w_{k-1} \in W_0 \times \cdots \times W_{k-1}$.

The coding scheme has to respect the edge capacities, i.e., if $\mathbf{M}_e$ is a random variable that represents a message sent along the edge $e$, then $H(\mathbf{M}_e) \leq c(e)$, where $H(\cdot)$ denotes the Shannon entropy. A *coding rate* of a network $R$ is the maximum $r$ such that there is a correct coding scheme for input random variables $\mathbf{W}_0, \ldots, \mathbf{W}_{k-1}$ where $H(\mathbf{W}_i) = \log |W_i| \geq r$ for all $i \in [k]$. A network coding can be defined also for directed cyclic networks or undirected networks but we will not use it here.

Network coding is related to multicommodity flows. A multicommodity flow for an undirected network $\bar{R} = (\bar{G}, c, (s_i, t_i)_{i \in [k]})$ specifies flows for each commodity $i$ such that they transport as many units of commodity from $s_i$ to $t_i$ as possible. A flow of the commodity $i$ is specified by a function $f^i : V \times V \to \mathbb{R}_0^+$ which describes for each pair of vertices $(u, v)$ how many units of the commodity $i$ are sent from $u$ to $v$. Each function $f^i$ has to satisfy:

1. If $u, v$ are not connected by an edge, then $f^i(u, v) = f^i(v, u) = 0$.

2. For each edge $\{u, v\} \in E(\bar{G})$, it holds that $f^i(u, v) = 0$ or $f^i(v, u) = 0$.

3. For each vertex $v$ that is not the source $s_i$ or the target $t_i$, it holds that what comes to the vertex $v$ goes out from the vertex $v$, i.e.,

$$\sum_{u \in V} f^i(u, v) = \sum_{u \in V} f^i(v, u).$$

4. What is sent from the source $s_i$ arrives to the target $t_i$, i.e.,

$$\sum_{u \in V} f^i(s_i, u) - f^i(u, s_i) = \sum_{u \in V} f^i(u, t_i) - f^i(t_i, u).$$

Moreover, all flows together have to respect the capacities, i.e., for each edge $e = \{u, v\} \in E(\bar{G})$ it must hold that $\sum_{i \in [k]} f^i(u, v) + f^i(v, u) \leq c(e)$. A *flow rate* of a network $\bar{R}$ is the maximum $r$ such that there is a multicommodity flow $F = (f^0, \ldots, f^{k-1})$ that for each $i$ transports at least $r$ units of the commodity $i$ from $s_i$ to $t_i$, i.e., for all $i$, it holds that $\sum_{u \in V} f^i(u, t_i) - f^i(t_i, u) \geq r$. A multicommodity flow for directed graphs is defined similarly, however, the flows can transport the commodities only in the direction of edges.

Let $R$ be a directed acyclic network of a flow rate $r'$. It is clear that for a coding rate $r$ of $R$ it holds that $r \geq r'$. As we can send the messages without coding and thus reduce the encoding problem to the flow problem. The opposite inequality does not hold: There is a directed network $R = (G, c, (s_i, t_i)_{i \in [k]})$ such that its coding rate is $\Omega(|V(G)|)$-times larger than its flow rate as shown by Adler et al. [2]. Thus, the network coding for directed networks provides an advantage over the simple solution given by the maximum flow. However, such a result is not known for undirected networks. Li and Li [24] conjectured that the network coding does not provide any advantage for undirected networks, thus for any undirected

network $\bar{R}$, the coding rate of $\bar{R}$ equals to the flow rate of $\bar{R}$. This conjecture is known as *Network Coding Conjecture* (NCC) and we state a weaker version of it below.

For a directed graph $G = (V, E)$ we denote by $\mathsf{un}(G)$ the undirected graph $(V, \bar{E})$ obtained from $G$ by making each directed edge in $E$ undirected (i.e., replacing each $(u, v) \in E(G)$ by $\{u, v\}$). For a directed acyclic network $R = \big(G, c, (s_i, t_i)_{i \in [k]}\big)$ we define the undirected network $\mathsf{un}(R) = \big(\mathsf{un}(G), \bar{c}, (s_i, t_i)_{i \in [k]}\big)$ by keeping the source-target pairs and capacities the same, i.e, $c\big((u, v)\big) = \bar{c}\big(\{u, v\}\big)$.

▶ **Conjecture 1** (Weaker NCC). *Let $R$ be a directed acyclic network, $r$ be a coding rate of $R$ and $\bar{r}$ be a flow rate of $\mathsf{un}(R)$. Then, $r = \bar{r}$.*

This conjecture was used to prove a conditional lower bound for sorting algorithms with an external memory [12] and for circuits multiplying two numbers [3].

## 4    NCC Implies Data Structure Lower Bounds

In this paper, we provide several connections between lower bounds for data structures and other computational models. The first connection is that NCC (Conjecture 1) implies lower bounds for data structures for the permutation inversion and the polynomial evaluation and interpolation. Assuming NCC, we show that a query time $t$ of a non-adaptive systematic data structure for any of the above problems satisfies $t \geq \Omega\big(\log n / \log \log n\big)$, even if it uses linear space, i.e., the advice string **a** has size $\varepsilon n \log n$ for sufficiently small constant $\varepsilon > 0$. Formally, we define $t_{\mathsf{Inv}}(s)$ as a query time of the optimal non-adaptive systematic data structure for the permutation inversion using space at most $s$. Similarly, we define $t_{\mathsf{Eval}}^{\mathbb{F}}(s)$ and $t_{\mathsf{Interp}}^{\mathbb{F}}(s)$ for the polynomial evaluation and interpolation over $\mathbb{F}$.

▶ **Theorem 2.** *Let $\varepsilon > 0$ be a sufficiently small constant. Assuming NCC, it holds that $t_{\mathsf{Inv}}(\varepsilon n \log n) \geq \Omega\big(\log n / \log \log n\big)$.*

▶ **Theorem 3.** *Let $\mathbb{F}$ be a field and $n$ be a divisor of $|\mathbb{F}| - 1$. Let $s = \varepsilon n \log |\mathbb{F}|$ for a sufficiently small constant $\varepsilon > 0$. Then assuming NCC, it holds that $t_{\mathsf{Eval}}^{\mathbb{F}}(s), t_{\mathsf{Interp}}^{\mathbb{F}}(s) \geq \Omega\big(\log n / \log \log n\big)$.*

Note that by Theorem 2, assuming NCC, it holds that $s \cdot t \geq \Omega\big(n \log^2 n / \log \log n\big)$ for $s = \varepsilon n \log n$ and $t = t_{\mathsf{Inv}}(s)$. The same holds for $t_{\mathsf{Eval}}^{\mathbb{F}}$ and $t_{\mathsf{Interp}}^{\mathbb{F}}$ by Theorem 3. Thus, these conditional lower bounds cross the barrier $\Omega(n \log n)$ for $s \cdot t$ given by the best unconditional lower bounds known for the function inversion [35, 10, 1, 16, 29, 11, 8] and the lower bound for the succinct data structures for the polynomial evaluation by Gál and Miltersen [14]. The lower bound by Larsen [22] says that any cell probe data structure for the polynomial evaluation using linear space $s_{\mathsf{cell}} = O(n \log n)$ needs at least logarithmic query time $t_{\mathsf{cell}} \geq \Omega(\log n)$ if the size of the field is of super-linear size in $n$, i.e., $|\mathbb{F}| \geq n^{1+\Omega(1)}$. Then $s_{\mathsf{cell}} \cdot t_{\mathsf{cell}} \geq \Omega(n \log^2 n)$. The lower bound given by Theorem 3 says that assuming NCC a non-adaptive data structure needs to read at least logarithmically many coefficients $\alpha_i$ of $p$ even if we know $\varepsilon n \log |\mathbb{F}|$ bits of information about the polynomial $p$ for free. Our lower bound holds also for linear-size fields.

To prove Theorems 2 and 3, we use the technique of Farhadi et al. [12]. The proof can be divided into two steps:

**1.** From a data structure for the problem we derive a network $R$ with $O(tn)$ edges such that $R$ admits an encoding scheme that is correct on a large fraction of the inputs. This step is distinct for each problem and the reductions are shown in Sections 5 and 6. This step uses new ideas and interestingly, it uses the data structure twice in a sequence.

**2.** If there is a network $R$ with $dn$ edges that admits an encoding scheme which is correct for a large fraction of inputs, then $d \geq \Omega(\log n / \log \log n)$. This step is common to all the problems. It was implicitly proved by Farhadi et al. [12] and Afshani et al. [3]. For the sake of completeness, we give a proof of this step in Appendix A.

## 5    NCC Implies a Weak Lower Bound for the Function Inversion

In this section, we prove Theorem 2 that assuming NCC, any non-adaptive systematic data structure for the permutation inversion requires query time at least $\Omega(\log n / \log \log n)$ even if it uses linear space. Let $\mathcal{D}$ be a data structure for inverting permutations of a linear space $s = \varepsilon n \log n$, for sufficiently small constant $\varepsilon < 1$, with query time $t = t_{\mathsf{Inv}}(s)$. Recall that $t_{\mathsf{Inv}}(s)$ is a query time of the optimal non-adaptive systematic data structure for the permutation inversion using space $s$. From $\mathcal{D}$ we construct a directed acyclic network $R = \big(G, c, (s_i, t_i)_{i \in [n]}\big)$ and an encoding scheme of a coding rate $\log n$. By Conjecture 1 we get that the flow rate of $\mathsf{un}(R) = \big(\bar{G}, c, (s_i, t_i)_{i \in [n]}\big)$ is $\log n$ as well. We prove that there are many source-target pairs of distance at least $\Omega(\log_t n)$. Since the number of edges of $\bar{G}$ will be $O(tn)$ and flow rate of $\mathsf{un}(R)$ is $\log n$, we are able to derive a lower bound $t \geq \Omega(\log n / \log \log n)$.

We construct the network $R$ in two steps. First, we construct a network $R'$ that admits an encoding scheme $E'$ such that $E'$ is correct only on a substantial fraction of all possible inputs. This might create correlations among messages received by the sources. However, to use the Network Coding Conjecture we need to have a coding scheme that is able to reconstruct messages sampled from independent distributions. To overcome this issue we use a technique introduced by Farhadi et al. [12] and from $R'$ we construct a network $R$ that admits a correct encoding scheme.

Let $R = \big(G, c, (s_i, t_i)_{i \in [k]}\big)$ be a directed acyclic network. Let each source receive a binary string of length $r$ as its input message, i.e., each $W_i = \{0,1\}^r$. If we concatenate all input messages $w_i$ we get a string of length $r \cdot k$, thus the set of all possible inputs for an encoding scheme for $R$ corresponds to the set $\mathcal{I} = \{0,1\}^{rk}$. We say an encoding scheme is correct on an input $\bar{w} = (w_0, \ldots, w_{k-1}) \in \mathcal{I}$ if it is possible to reconstruct all messages $w_i$ at appropriate targets. An $(\varepsilon, r)$-*encoding scheme* is an encoding scheme which is correct on at least $2^{(1-\varepsilon)rk}$ inputs in $\mathcal{I}$.

We say a directed network $R = \big(G, c, (s_i, t_i)_{i \in [k]}\big)$ is $(\delta, d)$-*long* if for at least $\delta k$ source-target pairs $(s_i, t_i)$, it holds that distance between $s_i$ and $t_i$ in $\mathsf{un}(G)$ is at least $d$. Here, we measure the distance in the undirected graph $\mathsf{un}(G)$, even though the network $R$ is directed. The following lemma is implicitly used by Farhadi et al. [12] and Afshani et al. [3]. We give its proof in Appendix A for the sake of completeness.

▶ **Lemma 4** (Implicitly used in [12, 3]). *Let $R = \big(G, q, (s_i, t_i)_{i \in [k]}\big)$ be a $(\delta, d)$-long directed acyclic uniform network for $\delta > \frac{5}{6}$ and sufficiently large $q \in \mathbb{R}^+$. Assume there is an $(\varepsilon, r)$-encoding scheme for $R$ for sufficiently small $\varepsilon$. Then assuming NCC, it holds that $\frac{|E(G)|}{k} \geq \delta' \cdot d$, where $\delta' = \frac{\delta - 5/6}{10}$.*

Now we are ready to prove a conditional lower bound for the permutation inversion. For the proof we use the following fact which follows from well-known Stirling's formula:

▶ **Fact 1.** *The number of permutations $[n] \to [n]$ is at least $2^{n \log n - 2n}$.*

▶ **Theorem 2.** *Let $\varepsilon > 0$ be a sufficiently small constant. Assuming NCC, it holds that $t_{\mathsf{Inv}}(\varepsilon n \log n) \geq \Omega(\log n / \log \log n)$.*
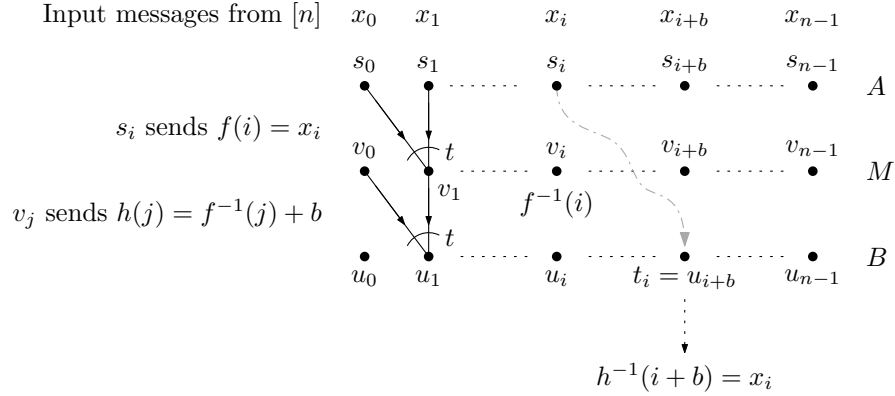
**Figure 1** A sketch of the graph $G'$ and encoding scheme $E$.

**Proof.** Let $\mathcal{D} = \mathcal{D}_n$ be the optimal data structure for the inversion of permutation on $[n]$ using space $\varepsilon n \log n$. We set $t = t_{\mathsf{Inv}}(\varepsilon n \log n)$. We will construct a directed acyclic uniform network $R = \big(G, r, (s_i, t_i)_{i \in [n]}\big)$ where $r = \log n$. Let $\varepsilon' = 2 \cdot \varepsilon + \frac{2}{q} + \frac{2}{\log n}$ for sufficiently large $q$ so that we could apply Lemma 4. The network $R$ will admit an $(\varepsilon', r)$-encoding scheme $E$. The number of edges of $G$ will be at most $2tn$ and the network $R$ will be $\big(\frac{9}{10}, d\big)$-long for $d = \frac{1}{2} \log_{qt} n$. Thus, by Lemma 4 we get that

$$2t = \frac{2tn}{n} \geq \Omega\big(\log_{qt} n\big),$$

from which we can conclude that $t \geq \Omega\big(\log n / \log \log n\big)$. Thus, it remains to construct the network $R$ and the scheme $E$.

First, we construct a graph $G'$ which will yield the graph $G$ by deleting some edges. The graph $G'$ has three layers of $n$ vertices: a source layer $A$ of $n$ sources $s_0, \ldots, s_{n-1}$, a middle layer $M$ of $n$ vertices $v_0, \ldots, v_{n-1}$ and a target layer $B$ of $n$ vertices $u_0, \ldots, u_{n-1}$. The targets $t_0, \ldots, t_{n-1}$ of $R$ will be assigned to the vertices $u_0, \ldots, u_{n-1}$ later.

We add edges according to the data structure $\mathcal{D}$: Let $Q_j \subseteq [n]$ be a set of oracle queries, which $\mathcal{D}$ makes during the computation of $f^{-1}(j)$, i.e., for each $i \in Q_j$, it queries the oracle of $f$ for $f(i)$. As $\mathcal{D}$ is non-adaptive, the sets $Q_j$ are well-defined. For each $j \in [n]$ and $i \in Q_j$ we add edges $(s_i, v_j)$ and $(v_i, u_j)$. We set a capacity of all edges to $r = \log n$. This finishes the construction of $G'$, see Fig. 1 for illustration of the graph $G'$.

The graph $G'$ has exactly $2tn$ edges. Moreover, the vertices of the middle and the target layer have in-degree at most $t$ as the incoming edges correspond to the oracle queries made by $\mathcal{D}$. However, some vertices of the source and the middle layer might have large outdegree, which is a problem that might prevent the network $R$ to be $\big(\frac{9}{10}, d\big)$-long. For example, the data structure $\mathcal{D}$ could always query $f(0)$. Then, there would be edges $(s_0, v_j)$ and $(v_0, u_j)$ for all $j \in [n]$, hence all vertices would be at distance at most 4 in $\mathsf{un}(G')$. So we need to remove edges adjacent to high-degree vertices. Let $W \subseteq V(G')$ be the set of vertices of out-degree larger than $qt$. We remove all edges incident to $W$ from $G'$ to obtain the graph $G$. (For simplicity, we keep the degree 0 vertices in $G$). Thus, the maximum degree of $G$ is at most $qt$. Since the graph $G'$ has $2tn$ edges, it holds that $|W| \leq \frac{2}{q} \cdot n$.

Now, we assign the targets of $R$ in such a way that $R$ is $\big(\frac{9}{10}, d\big)$-long. Let $C_v$ be the set of vertices of $G$ which have distance at most $d$ from $v$ in $\mathsf{un}(G)$. Since the maximum degree of $G$ is at most $qt$ and $d = \frac{1}{2} \log_{qt} n$, for each $v \in V(G)$, $|C_v| \leq 2\sqrt{n}$. In particular, for every source $s_i$ it holds that $|C_{s_i} \cap B| \leq 2\sqrt{n}$, i.e., there are at most $2\sqrt{n}$ vertices in the target

layer $B$ at distance smaller than $d$ from $v$. It follows from an averaging argument that there is an integer $b$ such that there are at least $n - 2\sqrt{n}$ sources $s_i$ with distance at least $d$ from $u_{i+b}$ in $\mathsf{un}(G)$. (Here the addition $i + b$ is modulo $n$.) We fix one such $b$ and set $t_i = u_{i+b}$. For $n$ large enough, it holds that $n - 2\sqrt{n} \geq \frac{9}{10} \cdot n$. Thus, the network $R$ is $\left(\frac{9}{10}, d\right)$-long.

It remains to construct the $(\varepsilon', r)$-encoding scheme $E$ for $R$ (see Fig. 1 for a sketch of the encoding $E$). Each source $s_i$ receives a number $x_i \in [n]$ as an input message. We interpret the string of the input messages $x_0, \ldots, x_{n-1}$ as a function. We define the function $f : [n] \rightarrow [n]$ as $f(i) = x_i$. We will consider only those inputs $x_0, \ldots, x_{n-1}$ which are pairwise distinct so that $f$ is a permutation.

At a vertex $v_j$ of the middle layer $M$ we want to compute $f^{-1}(j)$ using the data structure $\mathcal{D}$. To compute $f^{-1}(j)$ we need the appropriate advice string $\mathbf{a}_f$ and answers to the oracle queries $Q_j$. We fix an advice string $\mathbf{a}_f$ to some particular value which will be determined later, and we focus only on inputs $x_0, \ldots, x_{n-1}$ which have the same advice string $\mathbf{a}_f$. In $G'$ the vertex $v_j$ is connected exactly to the sources $s_i$ for $i \in Q_j$, but some of those connections might be missing in $G$. Thus for each $i$ such that $s_i \in W$, $x_i$ will be fixed to some particular value $c_i$ which will also be determined later. Each source $s_i$ sends the input $x_i$ along all outgoing edges incident to $s_i$. Thus, at a vertex $v_j$ we know the answers to all $f$-oracle queries in $Q_j$. Recall that $f(i) = x_i$ and each $x_i$ for $i \in Q_j$ was either fixed to $c_i$ or sent along the incoming edge $(s_i, v_j) \in E(G)$. We also know the advice string $\mathbf{a}_f$ as it was fixed. Therefore, we can compute $f^{-1}(j)$ at every vertex $v_j$. Note that $f^{-1}(j)$ is the index of the source which received $j$ as an input message, i.e., if $f^{-1}(j) = i$, then $x_i = j$.

Now, we define another permutation $h : [n] \rightarrow [n]$ as $h(j) = f^{-1}(j) + b$ where the addition is modulo $n$. Since $b$ is fixed, we can compute $h(j)$ at each vertex $v_j$. The goal is to compute $h^{-1}(\ell)$ at each vertex $u_\ell$ of the target layer. First, we argue that $h^{-1}(i + b) = x_i$. The permutation $f^{-1}$ maps an input message $x_i$ to the index $i$. The permutation $h$ maps an input message $x_i$ to the index $i + b$. Thus, the inverse permutation $h^{-1}$ maps the index $i + b$ to the input message $x_i$. If we are able to reconstruct $h^{-1}(i + b)$ at the target $t_i = u_{i+b}$, then in fact we are able to reconstruct $x_i$, the input message received by the source $s_i$.

To reconstruct $h^{-1}(\ell)$ at the vertex $u_\ell$ we use the same strategy as for reconstructing $f^{-1}(j)$ at vertices $v_j$. We use again $\mathcal{D}$, but this time for the function $h$. Again, we fix the advice string $\mathbf{a}_h$ of $\mathcal{D}$, and we fix $h(j)$ to some $d_j$ for each vertex $v_j \in W$. Each vertex $v_j$ sends the value $h(j)$ along all edges outgoing from $v_j$. To compute $h^{-1}(\ell)$ we need values $h(j)$ for all $j \in Q_\ell$, which are known to the vertex $u_\ell$. Again, they are either sent along the incoming edges or are fixed to $d_j$. Since the value of the advice string $\mathbf{a}_h$ is fixed, we can compute the value $h^{-1}(\ell) = x_{\ell-b}$ at the vertex $u_\ell$.

The network $R$ is correct on all inputs $x_0, \ldots, x_{n-1}$ which encode a permutation and which are consistent with the fixed advice strings and the fixed values to the degree zero vertices. Now, we argue that we can fix all the values so that there will be many inputs consistent with them. By Fact 1, there are at least $2^{(n \log n) - 2n}$ inputs $x_0, \ldots, x_{n-1}$ which encode a permutation. In order to make $R$ work, we fixed the following values:

1. Advice strings $\mathbf{a}_f$ and $\mathbf{a}_h$, in total $2\varepsilon \cdot n \log n$ bits.
2. An input message $c_i$ for each source $s_i$ in $W$ and a value $d_j$ for each vertex $v_j$ in $W$. Since $|W| \leq \frac{2}{q} \cdot n$ and $c_i, d_j \in [n]$, we fix $\frac{2}{q} \cdot n \log n$ bits in total.

Overall, we fix at most $\left(2\varepsilon + \frac{2}{q}\right) \cdot n \log n$ bits. Thus, the fixed values divide the input strings into at most $2^{(2\varepsilon + \frac{2}{q}) \cdot n \log n}$ buckets. In each bucket all the input strings are consistent with the fixed values. We conclude that there is a choice of values to fix so that its corresponding bucket contains at least $2^{(1 - 2\varepsilon - \frac{2}{q} - \frac{2}{\log n}) \cdot n \log n} = 2^{(1 - \varepsilon') \cdot n \log n}$ input strings which encode a permutation. We pick that bucket and fix the corresponding values. Thus, the scheme $E$ is

$(\varepsilon', r)$-encoding scheme, which concludes the proof. ◀

## 6    NCC Implies a Weak Lower Bound for the Polynomial Evaluation and Interpolation

In this section, we prove Theorem 3. The proof follows the blueprint of the proof of Theorem 2. The construction of a network $R$ from a data structure is basically the same. Thus, we mainly describe only an $(\varepsilon', r)$-encoding scheme for $R$.

▶ **Theorem 3.** *Let $\mathbb{F}$ be a field and $n$ be a divisor of $|\mathbb{F}| - 1$. Let $s = \varepsilon n \log |\mathbb{F}|$ for a sufficiently small constant $\varepsilon > 0$. Then assuming NCC, it holds that $t_{\mathsf{Eval}}^{\mathbb{F}}(s), t_{\mathsf{Interp}}^{\mathbb{F}}(s) \geq \Omega(\log n / \log \log n)$.*

**Proof.** Let $\mathcal{D} = \mathcal{D}_n$ be the optimal non-adaptive systematic data structure for the evaluation of polynomials of degree up to $n - 1$ over $\mathbb{F}$ and using space $s = \varepsilon n \log |\mathbb{F}|$. We set $t = t_{\mathsf{Eval}}^{\mathbb{F}}(s), r = \log |\mathbb{F}|$ and $\varepsilon' = 2\varepsilon + \frac{2}{q}$ for sufficently large $q$. Again, we will construct a network $R = \left( G, r, (s_i, t_i)_{i \in [n]} \right)$ from $\mathcal{D}$. To construct an $(\varepsilon', r)$-encoding scheme for $R$, we use entries of FFFT, i.e., we will evaluate polynomials of degree at most $n - 1$ in powers of a primitive $n$-th root of the unity. Thus, we fix a primitive $n$-th root of unity $\sigma \in \mathbb{F}$, which we know exists, as discussed in Section 2.2.
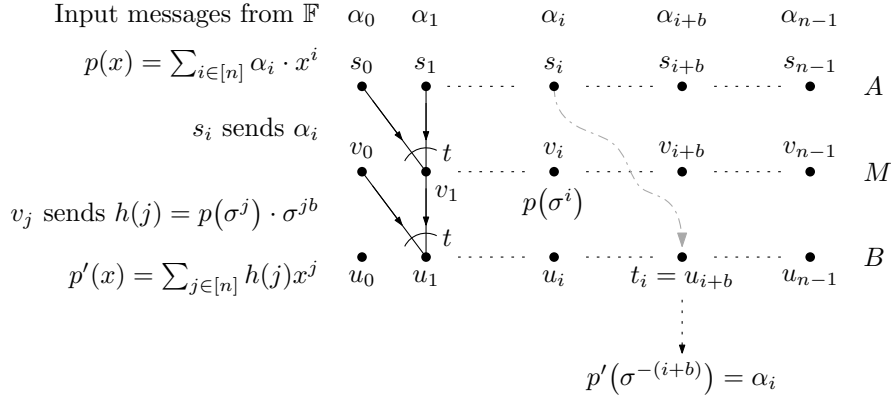
We create a network $R$ from $\mathcal{D}$ in the same way as we created in the proof of Theorem 2. By Lemma 4 we are able to conclude that $t \geq \Omega(\log n / \log \log n)$. First, we create a graph $G'$ of three layers $A = \{s_0, \ldots, s_{n-1}\}, M = \{v_0, \ldots, v_{n-1}\}$ and $B = \{u_0, \ldots, u_{n-1}\}$ and we add $2tn$ edges to $G'$ according to the queries of $\mathcal{D}$ – on the vertex $v_j$ we will evaluate a polynomial in a point $\sigma^j$ and on the vertex $u_j$ we will evaluate a polynomial in a point $\sigma^{-j}$. Then, we create a graph $G$ from $G'$ by removing edges incident to vertices in a set $W$, which contains vertices of degree higher than $qt$. Finally, we set a shift $b \in [n]$ and set $t_i = u_{i+b}$ in such a way that the network $R$ is $\left( \frac{9}{10}, d \right)$-long for $d = \frac{1}{2} \log_{qt} n$.

Now, we desribe an $(\varepsilon', r)$-encoding scheme $E$ for $R$ using $\mathcal{D}$. Each source $s_i$ receives an input message $\alpha_i \in \mathbb{F}$ which we interpret as coefficients of a polynomial $p \in \mathbb{F}[x]$ (that is $p(x) = \sum_{i \in [n]} \alpha_i x^i$). Each source $s_i$ sends its input message $\alpha_i$ along all outgoing edges from $s_i$. Each vertex $v_j$ computes $p(\sigma^j)$ using $\mathcal{D}$. Again, we fix the advice string $\mathbf{a}_p$ and the input messages $\alpha_i$ for the sources $s_i$ in $W$. Each vertex $v_j$ computes a value $h(j) = p(\sigma^j) \cdot \sigma^{jb}$ and sends it along all outgoing edges from $v_j$. We define a new polynomial $p'(x) = \sum_{j \in [n]} h(j) x^j$. We fix the advice string $\mathbf{a}_{p'}$ and the values $h(j)$ for each vertex $v_j \in W$. Thus, each vertex $u_\ell$ can compute a value $p'(\sigma^{-\ell})$. We claim that $\frac{p'(\sigma^{-\ell})}{n} = \alpha_{\ell-b}$.

$$\frac{p'(\sigma^{-\ell})}{n} = \frac{1}{n} \sum_{j \in [n]} h(j) \sigma^{-\ell j} = \frac{1}{n} \sum_{j \in [n]} p(\sigma^j) \sigma^{jb} \sigma^{-\ell j} = \frac{1}{n} \sum_{j \in [n]} \left( \sum_{i \in [n]} \alpha_i \sigma^{ji} \right) \sigma^{jb} \sigma^{-\ell j}$$

$$= \frac{1}{n} \sum_{i \in [n]} \alpha_i \left( \sum_{j \in [n]} \sigma^{ji} \sigma^{jb} \sigma^{-\ell j} \right) = \frac{1}{n} \sum_{i \in [n]} \alpha_i \left( \sum_{j \in [n]} \sigma^{j(i+b-\ell)} \right)$$

$$= \alpha_{(\ell - b \mod n)}$$

The last equality is by noting that $\sum_{j \in [n]} \sigma^{j(i+b-\ell)} = n$ for $i = \ell - b$ and $0$ otherwise. Therefore, at each target $t_i = u_{i+b}$ we can reconstruct the input message $\alpha_i$. See Fig. 2 for a sketch of the scheme $E$.

Again, we can fix values of advice strings $\mathbf{a}_p$ and $\mathbf{a}_{p'}$ (at most $2\varepsilon \cdot n \log |\mathbb{F}|$ fixed bits), input messages $\alpha_i$ for each $s_i \in W$ and value of $h(j)$ for each $v_j \in W$ (at most $\frac{2}{q} \cdot n \log |\mathbb{F}|$

Input messages from $\mathbb{F}$   $\alpha_0$   $\alpha_1$    $\alpha_i$    $\alpha_{i+b}$    $\alpha_{n-1}$

$p(x) = \sum_{i\in[n]} \alpha_i \cdot x^i$

$s_i$ sends $\alpha_i$

$v_j$ sends $h(j) = p(\sigma^j) \cdot \sigma^{jb}$

$p'(x) = \sum_{j\in[n]} h(j)x^j$

$p'(\sigma^{-(i+b)}) = \alpha_i$

**Figure 2** Sketch of the encoding scheme $E$.

fixed bits) in such a way there is a set $\mathcal{F}$ of inputs $(\alpha_0, \ldots, \alpha_{n-1})$ consistent with such fixing and $|\mathcal{F}| \geq 2^{(1-2\varepsilon-\frac{2}{q})n\log|\mathbb{F}|}$. Therefore, the scheme $E$ is $(\varepsilon', r)$-encoding scheme. This finishes the proof that $t^{\mathbb{F}}_{\mathsf{Eval}}(s) \geq \Omega\big(\log n / \log\log n\big)$.

Essentially, the same proof can be used to prove the lower bound for $t^{\mathbb{F}}_{\mathsf{Interp}}(s)$. Note that, the data structure $\mathcal{D}$ is used only for evaluating some polynomials in powers of the primitive root $\sigma$, i.e., computing entries of $\mathrm{FFFT}_{n,\sigma}(\alpha_0, \ldots, \alpha_{n-1})$. However as discussed in Section 2.2, it holds that $\mathrm{FFFT}_{n,\sigma} = n \cdot \mathrm{FFFT}^{-1}_{n,\sigma^{-1}}$. Moreover, entries of $\mathrm{FFFT}^{-1}_{n,\sigma^{-1}}(\beta_0, \ldots, \beta_{n-1})$ can be computed by a data structure for the polynomial interpolation. Thus, we may replace both uses of the data structure for the polynomial evaluation with a data structure for the polynomial interpolation. Therefore, we can use a data structure for the interpolation as $\mathcal{D}$ and with slight changes of $R$ and $E$, we would get again an $(\varepsilon', r)$-encoding scheme. ◄

## 7   Strong Lower Bounds for Data Structures and Lower Bounds for Boolean Circuits

In this section, we study a connection between non-adaptive data structures and boolean circuits. We are interested in circuits with binary AND and OR gates, and unary NOT gates. (See e.g. [20] for background on circuits).

Corrigan-Gibbs and Kogan [9] describe a connection between lower bounds for non-adaptive data structures and lower bounds for boolean circuits for a special case when the data structure computes function inversion. They show that we would get a circuit lower bound if any non-adaptive data structure using $O(n^\varepsilon)$ queries must use at least $\omega\big(n\log n / \log\log n\big)$ bits of advice (for some fixed constant $\varepsilon > 0$). To be able to formally restate their theorem we present some of their definitions. We define a *boolean operator* to be a family of functions $(F_n)_{n\in\mathbb{N}}$ for $F_n\colon \{0,1\}^n \to \{0,1\}^n$ represented by boolean circuits with $n$ input and $n$ output bits and constant fan-in gates. A boolean operator is said to be an *explicit operator* if the decision problem whether the $j$-th output bit of $F_n$ is equal to one is in the complexity class NP.

▶ **Theorem 5** (Corrigan-Gibbs and Kogan [9], Theorem 3 (verbatim))**.** *If every explicit operator has fan-in-two boolean circuits of size $O(n)$ and depth $O(\log n)$ then, for every $\varepsilon > 0$, there exists a family of strongly non-adaptive black-box algorithms that inverts all functions $f\colon [N] \to [N]$ using $O\big(N\log N / \log\log N\big)$ bits of advice and $O(N^\varepsilon)$ online queries.*

To prove their theorem Corrigan-Gibbs and Kogan [9] use the common bits model of

boolean circuits described by Valiant [30, 31, 32]. Valiant proves that for any circuit there is a small cut, called *common bits*, such that each output bit is connected just to few input bits (formally stated in Theorem 6). Corrigan-Gibbs and Kogan [9] use the common bits of the given circuit to create a non-adaptive data structure by setting the advice string to the content of common bits and the queries are to those function values which are still connected to the particular output after removing the common bits.

Observe that it follows from the proof of Theorem 5 that the hard explicit operator is turning the function table into the table of its inverse function. The theorem is therefore slightly stronger in the sense that if we have a data structure lower bound we also have a lower bound for a concrete boolean operator. It is also not straightforward to state a connection between circuits computing FFFT and non-adaptive data structures computing polynomial evaluation (resp. polynomial interpolation) as a consequence of Theorem 5. Thus, we restate the Valiant's result to be able to state a more general theorem.

▶ **Theorem 6** (Valiant [30, 31, 32])**.** *For every constant $\varepsilon > 0$, for every family of constant fan-in boolean circuits $\{C_n\}_{n \in \mathbb{N}}$, where $C_n\colon \{0,1\}^n \to \{0,1\}^n$ is of size $O(n)$ and depth $O(\log n)$, and for every $n \in \mathbb{N}$ it holds that the circuit $C_n$ contains a set of gates called* common bits *of size $O(n/\log\log n)$ such that if we remove those gates then each output bit is connected to at most $O(n^\varepsilon)$ input bits.*

Now we can state a general theorem translating lower bounds for non-adaptive data structures to circuit lower bounds. This allows us to apply the theorem directly to many different problems.

▶ **Theorem 7.** *For every $n \in \mathbb{N}$ let us define $b(n) = \lceil \log n \rceil$ and for every function $f_n\colon \{0,1\}^{n \cdot b(n)} \to \{0,1\}^{n \cdot b(n)}$ and for every $i \in [n]$ we define a function $f_{n,i}\colon \{0,1\}^{n \cdot b(n)} \to \{0,1\}^{b(n)}$ as follows: $f_{n,i}(x) = f_n(x)_{i \cdot b(n), i \cdot b(n)+1, \ldots, (i+1) \cdot b(n)-1}$, i.e., $f_{n,i}$ returns the $(i+1)$-st consecutive block of $b(n)$ bits of the output of $f_n$.*

*If there is a size $O(n \log n)$ and depth $O(\log n)$ circuit family $\{C_n\}_{n \in \mathbb{N}}$, where $C_n$ evaluates a function $f_n\colon \{0,1\}^{n \cdot b(n)} \to \{0,1\}^{n \cdot b(n)}$, then for every constant $\varepsilon > 0$ there exists a family of non-adaptive data structures $\{\mathcal{D}_n\}_{n \in \mathbb{N}}$, where $\mathcal{D}_n$ on input $x \in \{0,1\}^{n \cdot b(n)}$ uses $O(n \log n / \log\log n)$ bits of advice and on a query $j \in [n]$ answers $f_{n,j}(x)$ using $O(n^\varepsilon)$ queries to the input.*

The proof of Theorem 7 is the same as the proof of Theorem 3 of Corrigan-Gibbs and Kogan [9] (restated here as Theorem 5). Note that the data structures are not uniform in the sense that the algorithms for producing the advice string and for answering queries may differ for different input sizes. If we would like to get a uniform algorithm we would need the assumption that the explicit operator has linear size and logarithmic depth *uniform* circuits.

Let us state concrete instances of Theorem 7. First, we formally state the stronger version of Theorem 3 of Corrigan-Gibbs and Kogan [9], which follows from their proof.

▶ **Corollary 8.** *If there is a circuit family $\{C_n\}_{n \in \mathbb{N}}$, such that $C_n\colon \{0,1\}^{n\lceil \log n \rceil} \to \{0,1\}^{n\lceil \log n \rceil}$ is of size $O(n \log n)$ and depth $O(\log n)$ and inverts a function $f_n\colon [n] \to [n]$ (given on input as a function table) on all points (i.e., returns function table[1] of $f_n^{-1}$), then for every constant $\varepsilon > 0$ there exists a family of non-adaptive data structures $\{\mathcal{D}_n\}_{n \in \mathbb{N}}$ such that $\mathcal{D}_n$ on all input functions $f_n\colon [n] \to [n]$ uses $O(n \log n / \log\log n)$ bits of advice and for any $x \in [n]$ it answers $f_n^{-1}(x)$ using $O(n^\varepsilon)$ queries to the input.*

---

[1] When $f_n$ is not a permutation we allow a table of any function which has zero if there is no preimage and any preimage if there are more possibilities.

Theorem 7 is general enough to easily capture the problem of computing FFFT over a finite field. Note that by the connection of FFFT to polynomial evaluation and interpolation the following corollary captures both problems.

▶ **Corollary 9.** *Let $\mathcal{S} = \left\{ p^k \mid p \text{ is a prime}, k \in \mathbb{N}, k \neq 0 \right\}$ be the set of all sizes of finite fields. For each $n \in \mathcal{S}$, let $\mathbb{F}_n = GF(n)$ and $\sigma_n$ be a primitive $(n-1)$-th root of unity (thus a generator of the multiplicative group $\mathbb{F}_n^*$).*

*If there is a circuit family computing $FFFT_{n-1,\sigma_n}$ (over $\mathbb{F}_n$) of size $O(n \log n)$ and depth $O(\log n)$ (where each input and output number is represented by $\log |\mathbb{F}_n|$ bits) then for every $\varepsilon > 0$ there is a family of non-adaptive data structures $\{\mathcal{D}_n\}_{n \in \mathcal{S}}$ where $\mathcal{D}_n$ uses advice of size $O\big(n \log n / \log \log n\big)$ and on a query $j \in [n-1]$ outputs the $j$-th output of $FFFT_{n-1,\sigma_n}$ using $O(n^\varepsilon)$ queries to the input.*

To put the corollary in counter-positive way: if for some $\varepsilon > 0$, there are no non-adaptive data structures for polynomial interpolation, polynomial evaluation or FFFT with advise of size $o\big(n \log n / \log \log \log n\big)$ that use $O(n^\varepsilon)$ queries to the input then there are no linear-size circuits of logarithmic depth for FFFT.

In Theorem 2, resp. Theorem 3, we prove a conditional lower bound for permutation inversion, resp. polynomial evaluation and polynomial interpolation, of the form, that a non-adaptive data structure using $\varepsilon n \log n$ bits must do at least $\Omega\big(\log n / \log \log n\big)$ queries. It is not clear if assuming NCC we can get a sufficiently strong lower bound which would rule out non-adaptive data structures with sublinear advice string using $O(n^\varepsilon)$ oracle queries.

▶ **Corollary 10.** *We say that a circuit $C_n \colon \{0,1\}^{n\lceil \log n \rceil} \to \{0,1\}^{n\lceil \log n \rceil}$ sorts its input if on an input viewed as $n$ binary strings $x_1, x_2, \ldots, x_n \in \{0,1\}^{\lceil \log n \rceil}$ outputs the strings sorted lexicographically.*

*If there is a circuit family $(C_n)_{n \in \mathbb{N}}$, where $C_n \colon \{0,1\}^{n\lceil \log n \rceil} \to \{0,1\}^{n\lceil \log n \rceil}$ sorts its inputs, and each circuit $C_n$ is of size $O(n \log n)$ and depth $O(\log n)$ then for every $\varepsilon > 0$, for every permutation $f \colon [n] \to [n]$ there is a non-adaptive data structure for inverting $f$ that uses advice of size $O\big(n \log n / \log \log n\big)$ and $O(n^\varepsilon)$ queries.*

The works of Farhadi et al. [12] and Asharov et al. [4] connect the NCC conjecture directly to lower bounds for sorting. Their work studies sorting $n$ numbers of $k + w$ bits by their first $k$ bits. Namely Asharov et al. [4] show that NCC implies that constant fan-in constant fan-out circuits must have size $\Omega\big(nk(w - \log(n) + k)\big)$ whenever $w > \log(n) - k$ and $k \leq \log n$. This is incomparable to our results as we have $w = 0$.

### References

1. Hamza Abusalah, Joël Alwen, Bram Cohen, Danylo Khilko, Krzysztof Pietrzak, and Leonid Reyzin. Beyond hellman's time-memory trade-offs with applications to proofs of space. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II*, volume 10625 of *Lecture Notes in Computer Science*, pages 357–379. Springer, 2017. `doi:10.1007/978-3-319-70697-9\_13`.

2. Micah Adler, Nicholas J. A. Harvey, Kamal Jain, Robert Kleinberg, and April Rasala Lehman. On the capacity of information networks. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, SODA '06, page 241–250, USA, 2006. Society for Industrial and Applied Mathematics.

3. Peyman Afshani, Casper Benjamin Freksen, Lior Kamma, and Kasper Green Larsen. Lower Bounds for Multiplication via Network Coding. In Christel Baier, Ioannis Chatzigiannakis,

Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 10:1–10:12, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `http://drops.dagstuhl.de/opus/volltexte/2019/10586`, `doi:10.4230/LIPIcs.ICALP.2019.10`.

4    Gilad Asharov, Wei-Kai Lin, and Elaine Shi. Sorting short keys in circuits of size o (n log n). In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2249–2268. SIAM, 2021.

5    Michael Ben-Or and Prasoon Tiwari. A deterministic algorithm for sparse multivariate polynomial interpolation. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, page 301–309, New York, NY, USA, 1988. Association for Computing Machinery. `doi:10.1145/62212.62241`.

6    Michael Clausen, Andreas Dress, Johannes Grabmeier, and Marek Karpinski. On zero-testing and interpolation of k-sparse multivariate polynomials over finite fields. *Theor. Comput. Sci.*, 84(2):151–164, July 1991. `doi:10.1016/0304-3975(91)90157-W`.

7    James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

8    Sandro Coretti, Yevgeniy Dodis, Siyao Guo, and John Steinberger. Random oracles and non-uniformity. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2018 Proceedings*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pages 227–258. Springer Verlag, 2018. 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT 2018 ; Conference date: 29-04-2018 Through 03-05-2018. `doi:10.1007/978-3-319-78381-9_9`.

9    Henry Corrigan-Gibbs and Dmitry Kogan. The function-inversion problem: Barriers and opportunities. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography*, pages 393–421, Cham, 2019. Springer International Publishing.

10   Anindya De, Luca Trevisan, and Madhur Tulsiani. Time space tradeoffs for attacks against one-way functions and prgs. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, pages 649–665, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

11   Yevgeniy Dodis, Siyao Guo, and Jonathan Katz. Fixing cracks in the concrete: Random oracles with auxiliary input, revisited. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017*, pages 473–495, Cham, 2017. Springer International Publishing.

12   Alireza Farhadi, MohammadTaghi Hajiaghayi, Kasper Green Larsen, and Elaine Shi. Lower bounds for external memory integer sorting via network coding. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, page 997–1008, New York, NY, USA, 2019. Association for Computing Machinery. `doi:10.1145/3313276.3316337`.

13   Amos Fiat and Moni Naor. Rigorous time/space trade-offs for inverting functions. *SIAM J. Comput.*, 29(3):790–803, December 1999. `doi:10.1137/S0097539795280512`.

14   Anna Gál and Peter Bro Miltersen. The cell probe complexity of succinct data structures. In Jos C. M. Baeten, Jan Karel Lenstra, Joachim Parrow, and Gerhard J. Woeginger, editors, *Automata, Languages and Programming*, pages 332–344, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

15   Joachim von zur Gathen and Jrgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, USA, 3rd edition, 2013.

16   Rosario Gennaro, Yael Gertner, Jonathan Katz, and Luca Trevisan. Bounds on the efficiency of generic cryptographic constructions. *SIAM J. Comput.*, 35(1):217–246, July 2005. `doi:10.1137/S0097539704443276`.

**17**    Dima Grigoryev, Marek Karpinski, and Michael Singer. Fast parallel algorithms for sparse multivariate polynomial interpolation over finite fields. *SIAM J. Comput.*, 19:1059–1063, 12 1990. `doi:10.1137/0219073`.

**18**    M. Hellman. A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory*, 26(4):401–406, 1980. `doi:10.1109/TIT.1980.1056220`.

**19**    Gábor Ivanyos, Marek Karpinski, Miklos Santha, Nitin Saxena, and Igor E. Shparlinski. Polynomial interpolation and identity testing from high powers over finite fields. *Algorithmica*, 80(2):560–575, February 2018. `doi:10.1007/s00453-016-0273-1`.

**20**    Stasys Jukna. *Boolean function complexity: advances and frontiers*, volume 27. Springer Science & Business Media, 2012.

**21**    K. S. Kedlaya and C. Umans. Fast modular composition in any characteristic. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 146–155, 2008. `doi:10.1109/FOCS.2008.13`.

**22**    Kasper Green Larsen. Higher cell probe lower bounds for evaluating polynomials. In *Proceedings of the 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, FOCS '12, page 293–301, USA, 2012. IEEE Computer Society. `doi:10.1109/FOCS.2012.21`.

**23**    Kasper Green Larsen, Omri Weinstein, and Huacheng Yu. Crossing the logarithmic barrier for dynamic boolean data structure lower bounds. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, page 978–989, New York, NY, USA, 2018. Association for Computing Machinery. `doi:10.1145/3188745.3188790`.

**24**    Zongpeng Li and Baochun Li. Network coding: The case of multiple unicast sessions. *Proceedings of the 42nd Allerton Annual Conference on Communication, Control, and Computing*, 10 2004.

**25**    Peter Bro Miltersen. On the cell probe complexity of polynomial evaluation. *Theor. Comput. Sci.*, 143(1):167–174, May 1995. `doi:10.1016/0304-3975(95)80032-5`.

**26**    John M Pollard. The fast fourier transform in a finite field. *Mathematics of computation*, 25(114):365–374, 1971.

**27**    Jean-Pierre Serre. *A course in arithmetic*, volume 7. Springer Science & Business Media, 2012.

**28**    A. Smoktunowicz, I. Wróbel, and P. Kosowski. A new efficient algorithm for polynomial interpolation. *Computing*, 79(1):33–52, February 2007. `doi:10.1007/s00607-006-0185-z`.

**29**    Dominique Unruh. Random oracles and auxiliary input. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007*, pages 205–223, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

**30**    Leslie G Valiant. Graph-theoretic arguments in low-level complexity. In *International Symposium on Mathematical Foundations of Computer Science*, pages 162–176. Springer, 1977.

**31**    Leslie G Valiant. Exponential lower bounds for restricted monotone circuits. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 110–117, 1983.

**32**    Leslie G Valiant. Why is boolean complexity theory difficult. *Boolean Function Complexity*, 169(84-94):4, 1992.

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

**34**    Emanuele Viola. Lower bounds for data structures with space close to maximum imply circuit lower bounds. *Theory of Computing*, 15(18):1–9, 2019. URL: `http://www.theoryofcomputing.org/articles/v015a018`, `doi:10.4086/toc.2019.v015a018`.

**35**    A. C.-C. Yao. Coherent functions and program checkers. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, STOC '90, page 84–94, New York, NY, USA, 1990. Association for Computing Machinery. `doi:10.1145/100216.100226`.

## A    Proof of Lemma 4

▶ **Lemma 4** (Implicitly used in [12, 3]). *Let $R = \big(G, q, (s_i, t_i)_{i \in [k]}\big)$ be a $(\delta, d)$-long directed acyclic uniform network for $\delta > \frac{5}{6}$ and sufficiently large $q \in \mathbb{R}^+$. Assume there is an*

$(\varepsilon, r)$-*encoding scheme for* $R$ *for sufficiently small* $\varepsilon$. *Then assuming NCC, it holds that* $\frac{|E(G)|}{k} \geq \delta' \cdot d$, *where* $\delta' = \frac{\delta - 5/6}{10}$.

For the proof, we use an $\mathcal{F}$-correction game introduced by Farhadi et al. [12] (the statement of the following definition and lemma is due to Afshani et al. [3]).

▶ **Definition 11** ($\mathcal{F}$-correction game [12, 3]). *Let* $\mathcal{F} \subseteq \{0,1\}^{m\ell}$. *The* $\mathcal{F}$-*correction game* with $\ell + 1$ *players is defined as follows. The game is played by* $\ell$ *ordinary players* $p_0, \ldots, p_{\ell-1}$ *and one designated supervisor player* $\bar{u}$. *The supervisor* $\bar{u}$ *receives* $\ell$ *strings* $\alpha_0, \ldots, \alpha_{\ell-1} \in \{0,1\}^m$ *chosen independently at random. For every* $i \in [\ell]$, $\bar{u}$ *sends* $p_i$ *a message* $\beta_i$. *Given* $\beta_i$, *the player* $p_i$ *produces a string* $\gamma_i \in \{0,1\}^m$ *such that* $\big((\alpha_0 \oplus \gamma_0), \ldots, (\alpha_{\ell-1} \oplus \gamma_{\ell-1})\big) \in \mathcal{F}$.

▶ **Lemma 12** ([12, 3]). *If* $|\mathcal{F}| \geq 2^{(1-\varepsilon)m\ell}$, *then there exists a protocol for the* $\mathcal{F}$-*correction game with* $\ell + 1$ *players such that the messages* $(\beta_i)_{i \in [\ell]}$ *are prefix-free and*

$$\sum_{i \in [\ell]} \mathbb{E}|\beta_i| \leq 3\ell + 2\ell \log\left(\sqrt{\frac{\varepsilon}{2}} \cdot m + 1\right) + \sqrt{\frac{\varepsilon}{8}} \cdot m\ell \log \frac{2}{\varepsilon}. \tag{1}$$

Observe that for sufficiently small $\varepsilon$ and sufficiently large $m$ the formula in Equation 1 can be bounded by $\frac{m\ell}{4}$. Thus, we can suppose that the expected total length of the messages sent by supervisor $u$ in the $\mathcal{F}$-correction game is at most $\frac{m\ell}{4}$.

**Proof of Lemma 4.** Let $R = \big(G, q, (s_i, t_i)_{i \in [k]}\big)$ be a network given by the assumption of the lemma. We will create a directed acyclic network $R' = \big(G', c', (s'_i, t_i)_{i \in [k]}\big)$ which will admits a correct encoding scheme. Thus, we will be able to apply NCC to $R'$. Note that, the network $R'$ has new sources $s'_i$ but the original targets $t_i$.

The network $R'$ is defined as follows. We add new sources $s'_1, \ldots, s'_k$ and one special vertex $u$ to the graph $G$. For each $i$ we add the following new edges:

- Edge $(s'_i, s_i)$ and $(s'_i, u)$ of capacity $r$, i.e, edges connecting the new sources with the original ones and the new special vertex.
- Edges connecting the new special vertex $u$ with the original sources $s_i$ and the targets $t_i$, i.e, the edges $(u, s_i)$ and $(u, t_i)$ of capacity $\mathbb{E}|\beta_i|$, where $\beta_i$ is the message sent by the supervisor $\bar{u}$ to the player $p_i$ in the protocol for the $\mathcal{F}$-correction game given by Lemma 12.

This finishes the construction of $R'$. By assumption, there is a set $\mathcal{F} \subseteq \{0,1\}^{kr}$ and an encoding scheme $E$ for $R$ such that $E$ is correct on inputs in $\mathcal{F}$. Note that $R$ is a subnetwork of $R'$. Thus, to create an encoding scheme for $R'$ which will be correct on every input in $\{0,1\}^{kr}$ we use an encoding $E$ to recover some messages $w = (w_0, \ldots, w_{k-1}) \in \mathcal{F}$ and the special vertex $u$ which will send messages as the supervisor $\bar{u}$ in the $\mathcal{F}$-correction game. After that, the targets $t_i$ will be able to reconstruct the input messages $w'_i$ received at the new sources $s'_i$.

More formally, let $w'_i$ be an input message received at the source $s'_i$. Each $w'_i$ is uniformly sampled from $\{0,1\}^r$ (independently on other $w'_j$). Now, the encoding scheme $E'$ for $R'$ works as follows:

1. Each source $s'_i$ sends the input message $w'_i$ to the vertex $s_i$ and $u$.
2. The vertex $u$ computes the messages $\beta_0, \ldots, \beta_{k-1}$ according to the protocol given by Lemma 12 (applied for the messages $\alpha_i = w'_i$). Then for each $i \in [k]$, the vertex $u$ sends the messages $\beta_i$ to the vertex $s_i$ and $t_i$.

**3.** Each vertex $s_i$ computes the string $\gamma_i$. By Lemma 12, it holds that

$$\big((w'_0 \oplus \gamma_0), \ldots, (w'_{k-1} \oplus \gamma_{k-1})\big) \in \mathcal{F}.$$

Thus, we can use the encoding scheme $E$ for $R$ to reconstruct strings $w'_i \oplus \gamma_i$ at each target $t_i$.

**4.** Each target $t_i$ can reconstruct strings $w'_i \oplus \gamma_i$ and $\gamma_i$. Thus, it can reconstruct the input message $w'_i$.

By construction of the network $R'$, it is clear that the encoding scheme $E'$ respects the capacities $c'$.

The encoding scheme $E'$ witnesses that the coding rate of $R'$ is at least $r$. Thus by NCC (Conjecture 1), we conclude that the flow rate of $\mathsf{un}(R')$ is at least $r$ as well, i.e., there is a multicommodity flow $F = (f^0, \ldots, f^{k-1})$ for $\mathsf{un}(R')$ which transports at least $r$ units of each commodity $i$. Now, we argue that there is only a small fraction of the total flow which goes through the special vertex $u$.

▷ **Claim 13.** The size of total flow which goes through $u$ is at most $\frac{3}{4}kr$.

Proof. The total capacity of the edges incident to the vertex $u$ is at most $\frac{3}{2}kr$. The vertex $u$ is incident to $k$ edges $\{u, s'_i\}$ of capacity $r$, which contribute by $kr$ to the total capacity. Then for each $i$, the vertex $u$ is incident to the edges $\{u, s_i\}$ and $\{u, t_i\}$, which have both capacity $\mathbb{E}|\beta_i|$. By Lemma 12, we have that $\sum_i \mathbb{E}|\beta_i| \leq \frac{1}{4}kr$. Thus, these edges contribute by $\frac{1}{2}kr$ to the total capacity. By conservation of the flow, it must hold that

$$\sum_{i \in [k]} \sum_{v \in V(G')} f^i(v, u) \leq \frac{3}{4}kr,$$

$$\sum_{i \in [k]} \sum_{v \in V(G')} f^i(u, v) \leq \frac{3}{4}kr.$$

◁

Let $A \subseteq [k]$ be a set of indices of source-target pairs $(s_i, t_i)$ such that at least $\frac{r}{10}$ units of the commodity $i$ do not go through the vertex $u$. It follows that the set $A$ is substantially large.

▷ **Claim 14.** $|A| \geq \frac{k}{6}$.

Proof. Suppose opposite, $|A| < \frac{k}{6}$, i.e., there are at least $\frac{5k}{6}$ source-target pairs $(s_i, t_i)$ such that strictly more than $\frac{9r}{10}$ units of the commodity $i$ goes through the vertex $u$. Therefore, the total size of the flow going through $u$ is strictly larger than $\frac{5k}{6} \cdot \frac{9r}{10} = \frac{3}{4}kr$, which contradicts Claim 13. ◁

Let $L \subseteq [k]$ be a set of indices of pairs $(s_i, t_i)$ such that their distance in $\mathsf{un}(G)$ is at least $d$. By the assumption of the lemma, it holds that $|L| \geq \delta k$. Note that for each $i$, the distance between $s_i$ and $t_i$ in $G'$ is 2 because of the vertex $u$. However, due to Claim 14 there is a lot of source-target pairs $(s_i, t_i)$ which are far in $\mathsf{un}(G)$ and some units of the commodity $i$ do not go through $u$:

$$|A \cap L| \geq |A| - |[k] \setminus L| = \left(\delta - \frac{5}{6}\right) \cdot k.$$

Let $L' = A \cap L$, i.e., the set $L'$ contains indices $i \in [k]$ such that distance between $s_i$ and $t_i$ in $\mathsf{un}(G)$ is at least $d$ and at least $\frac{r}{10}$ units of the commodity $i$ do not go through $u$ – thus,

it has to go through paths of length at least $d$. Now, we are ready to prove the assertion of the lemma. Let $\bar{E} = E\big(\mathsf{un}(G)\big)$.

$$
\begin{aligned}
r \cdot |\bar{E}| = \sum_{e \in \bar{E}} c(e) &\geq \sum_{\{v,w\} \in \bar{E}} \sum_{i \in [k]} f^i(v,w) + f^i(w,v) \\
&\geq \sum_{i \in L'} \sum_{\{v,w\} \in \bar{E}} f^i(v,w) + f^i(w,v) \\
&\geq \left(\delta - \frac{5}{6}\right) k \cdot \frac{r}{10} d = \delta' \cdot kr \cdot d \qquad\qquad \text{By definition of } L'.
\end{aligned}
$$

It follows that $\frac{|\bar{E}|}{k} \geq \delta' \cdot d$. ◀