Cell-Probe Proofs and Nondeterministic Cell-Probe Complexity

Yitong Yin[⋆]

Department of Computer Science, Yale University
yitong.yin@yale.edu.

Abstract. We study the nondeterministic cell-probe complexity of static data structures. We introduce *cell-probe proofs (CPP)*, a proof system for the cell-probe model, which describes verifications instead of computations in the cell-probe model. We present a combinatorial characterization of CPP. With this novel tool, we prove the following lower bounds for the nondeterministic cell-probe complexity of static data structures:

- We show that there exist data structure problems which have super-constant nondeterministic cell-probe complexity. In particular, we show that for the exact nearest neighbor search (NNS) problem or the partial match problem in high dimensional Hamming space, there does not exist a static data structure with Poly(n) cells, each of which contains $n^{o(1)}$ bits, such that the nondeterministic cell-probe complexity is $O(1)$, where n is the number of points in the data set for the NNS or partial match problem.
- For the polynomial evaluation problem, if single-cell nondeterministic probes are sufficient, then either the size of a single cell is close to the size of the whole polynomial, or the total size of the data structure is close to that of a naive data structure that stores results for all possible queries.

1 Introduction

We study the problem of nondeterministic cell-probe complexity of static data structures.

Given a set Y of data instances, and a set X of possible queries, a data structure problem can be abstractly defined as a function f mapping each pair consisting of a query $x \in X$ and a data instance $y \in Y$ to an answer. One of the most well-studied examples of data structure problems is the "membership query": $X = [m]$ is a data universe, $Y = \binom{[m]}{n}$, and $f(x, y) = 1$ if $x \in y$ and $f(x, y) = 0$ if otherwise.

There are some other important examples of data structure problems:

Exact nearest neighbor search (NNS): given a metric space U, let $X = U$ and $Y = \binom{U}{n}$, and for every $x \in X$ and $y \in Y$, $f(x, y)$ is defined as the closest point to x in y according to the metric.

[⋆] Supported by a Kempner Foundation Fellowship and NSF grant CNS-0435201.

Partial match: $X = \{0, 1, *\}^d$, $Y = \binom{\{0,1\}^d}{n}$, and $f(x, y) \in \{0, 1\}$ such that for every $x \in X$ and $y \in Y$, $f(x, y) = 1$ if and only if there exists $z \in y$ having either $x_i = z_i$ or $x_i = *$ for every i.
Polynomial evaluation: $X = 2^k$ is a finite field, $Y = 2^{kd}$ is the set of all $(d-1)$-degree polynomials over the finite field 2^k, and $f(x, y)$ returns the value of $y(x)$.

A classic computational model for static data structures is the cell-probe model [11]. For each data instance y, a table of cells is constructed to store y. This table is called a static data structure for some problem f. Upon a query x, an all-powerful algorithm tries to compute $f(x, y)$, based on adaptive random access (probes) to the cells.

The cell-probe model is a clean and general model for static data structures and serves as a great tool for the study of lower bounds. Previous research on static data structures in the cell-probe model has focused on the complexity of adaptive cell-probes. In this work, we focus on the complexity of nondeterministic cell-probes and the tradeoff between the number of probes needed and with space. We speculate that it is an important problem because: (1) in considering the complexity of data structures, nondeterminism is a very natural extension to the cell-probe model; (2) instead of adaptive computations, nondeterministic cell-probes capture the question of verification, which is a natural and important aspect of data structures.

Although nondeterministic cell-probe complexity is an important problem, there are few general tools and techniques for studying it, especially for the case of static data structures. In fact, because of the great generality of the cell-probe model, even for deterministic cell-probe complexity, super-constant lower bounds for static data structures are rare. Nondeterminism grants the cell-probe model extra power and makes non-trivial lower bounds even rarer. For many standard examples of data structure problems, such as membership query, it is easy to construct a data structure that has standard space usage and *constant* nondeterministic cell-probe complexity.

It is thus worth asking whether there exists any data structure problem such that in data structures with feasible sizes (polynomial in the size of data set), the nondeterministic cell-probe complexity is super-constant. More importantly, it calls for a general technique to prove lower bounds on the nondeterministic cell-probe complexity of static data structures.

1.1 Our contribution

In this paper, we initiate the study of nondeterministic cell-probe complexity for static data structures. As a first step, we characterize the power of a single-cell nondeterministic probe. Although at first glance this may seem like a very restricted case, by applying a trivial parameter reduction, we show that the case of a single-cell probe is actually a canonical case for all nondeterministic cell-probe mechanisms, and is thus sufficient to prove super-constant lower bounds for general nondeterministic cell-probe mechanisms.

We introduce cell-probe proofs, a proof system in the cell-probe model. This notion of proofs corresponds to considering verifications instead of computations in the cell-probe model. Unlike the fully adaptive computations in the traditional cell-probe model, the formulation of cell-probe proofs shows a combinatorial simplicity. We introduce a combinatorial structure that fully characterizes which problems have single-cell proofs, and general cell-probe proofs are reduced to this case.

With these novel tools, we show following lower bounds on nondeterministic cell-probe complexity:

– We show that there exist static data structure problems with super-constant nondeterministic cell-probe complexity. In particular, we show that for the exact nearest neighbor search (NNS) problem or partial match problem in high dimensional Hamming space, there does not exist a static data structure with Poly(n) cells, each of which contains $n^{o(1)}$ bits, such that the nondeterministic cell-probe complexity is $O(1)$, where n is the number of points in the data set for the NNS or partial match problem.
– For the polynomial evaluation problem, if for a static data structure, the single-cell nondeterministic probes are sufficient to answer queries, then either the size of the single cell is close to the size of the whole polynomial, or the total size of the data structure is close to that of the naive data structure that stores results for all possible queries.

1.2 Related work

To the best of our knowledge, there is no general technique for proving lower bounds for nondeterministic cell-probe complexity of static data structures. Nor do there exist any non-trivial lower bounds for this question. Previous work on static data structures in the cell-probe model have focused on the complexity of adaptive cell-probes. The most important tool for proving such lower bounds is asymmetric communication complexity as introduced by Miltersen *et al.* in [10].

In [6], Fredman and Saks introduce the *chronogram method*. This powerful technique is specialized for proving the query/update trade-off for dynamic data structures, especially for the problems which are hard only in the dynamic case. It is worth noting that the chronogram method can prove nondeterministic lower bounds for certain dynamic data structure problems. This is formally addressed by Husfeldt and Rauhe in [7], and recently by Demaine and Pătraşcu in [5]. However, as pointed in [7], this is only a by-product of the nondeterministic nature of chronogram method and can only yield amortized query/update trade-offs for dynamic data structure problems with a certain property. Because of the unique structure of the chronogram method, this technique can not be utilized to prove lower bounds for static data structures.

2 Cell-probe proofs

A **static data structure problem** or just **data structure problem**, is represented as a boolean function $f : X \times Y \to \{0, 1\}$. For the purposes of proving lower bounds, we only consider decision problems. We refer to each $y \in Y$ as **data** and each $x \in X$ as a **query**. For each pair of x and y, $f(x, y)$ specifies the result of the query x to the data structure that represents the data y.

In the cell-probe model (c.f. [11, 6]), the data instance y is preprocessed and stored in cells, and for each query x, the value of $f(x, y)$ is decided by adaptive probes to the cells. Formally, a cell-probe scheme consists of a table structure and a query algorithm. The table structure $T : Y \times I \to \{0, 1\}^b$ specifies a table $T_y : I \to \{0, 1\}^b$ for each data instance y, which maps indices of cells to their contents. Given a query x, the query algorithm makes a sequence of probes $i_1, i_2, \ldots$ to the cells, where i_k depends on x and all previous cell probes $\langle i_1, T_y(i_1)\rangle, \langle i_2, T_y(i_2)\rangle, \ldots, \langle i_{k-1}, T_y(i_{k-1})\rangle$. The value of $f(x, y)$ is decided at last based on the collected information.

In this work, we focus on nondeterministic cell-probes. Given a query x to a data instance y, a set of t cells $i_1, i_2, \ldots, i_t$ are probed nondeterministically, such that the value of $f(x, y)$ is decided based on the probed information $\langle i_1, T_y(i_1)\rangle, \langle i_2, T_y(i_2)\rangle, \ldots, \langle i_t, T_y(i_t)\rangle$.

In order to formally characterize nondeterministic cell-probes for data structures, we introduce a new concept, **cell-probe proofs**, which formalizes the notion of proofs and verifications in the cell-probe model. For a specific data structure problem f, a cell-probe proof system (CPP) may be defined for f as described below.

We can think of a cell-probe proof system as a game played between an honest verifier and an untrusted prover. Both of them have unlimited computational power. Given an instance of data, a table of cells is honestly constructed according to the rules known to both prover and verifier. Both the prover and the verifier know the query, but only the prover can observe the whole table and thus knows the data. The prover tries to convince the verifier about the result of the query to the data by revealing certain cells. After observing the revealed cells, the verifier either decides the correct answer, or rejects the proof, but can not be tricked by the prover into returning a wrong answer.

Formally, a cell-probe proof system (CPP) consists of three parts:

- A table structure $T : Y \times I \to \{0, 1\}^b$. For any data y, a table $T_y : I \to \{0, 1\}^b$ is a mapping from indices of cells to their contents.
- A prover P. For every x and y, $P_{xy} \subseteq I$ is a set of cells. We refer to P_{xy} as a **proof** and $\{\langle i, T_y(i)\rangle \mid i \in P_{xy}\}$ as a **certificate**.
- A verifier v, which maps the queries with the certificates to the answers $\{0, 1, \perp\}$. Given an instance of data y, for any query x, both of the following conditions hold:

$$\begin{array}{ll}\text{(Completeness)} & \exists P_{xy} \subseteq I : v(x, \{\langle i, T_y(i)\rangle \mid i \in P_{xy}\}) = f(x, y), \text{ and} \\ \text{(Soundness)} & \forall P' \subseteq I : \; v(x, \{\langle i, T_y(i)\rangle \mid i \in P'\}) = \begin{cases} f(x, y) \\ \perp \end{cases} .\end{array}$$

An (s, b, t)-CPP is a CPP such that for every x and y: (1) the table has s cells, i.e. $|I| = s$; (2) each cell contains b bits; (3) each proof consists of t cell probes, i.e. $|P_{xy}| = t$.

Example: For the membership problem[11], where $X = [m]$ and $Y = \binom{[m]}{n}$, and $f(x, y) = 1$ if and only if $x \in y$, a naive construction shows a 2-cell proof: with a sorted table storing y, if $x \in y$, the proof is the cell that contains x, if $x \notin y$, the proof consists of two consecutive cells which are the predecessor and successor of x. The same CPP also works for predecessor search[2].

The notion of cell-probe proofs captures the necessary information to answer queries, and characterizes the nondeterministic probes in the cell-probe model. It is natural to see that for a cell-probe scheme, for any query, the cells probed by an adaptive algorithm contain a cell-probe proof. This can be seen as a data structure counterpart of $P \subseteq NP$.

It is important to note that although a data structure problem is nothing but a boolean function, CPP is very different from the certificate complexity of boolean functions [4]. In CPP, the prover and the verifier communicate with each other via a table structure, which distinguishes CPP from standard certificate complexity. For any data structure problem, the table structure can always store the results for all queries, making one cell-probe sufficient to prove the result, which is generally impossible in the model of certificate complexity.

Unlike adaptive cell-probes, CPP has a static nature, which is convenient for reductions. As stated by the following lemma, any CPP can be trivially reduced to 1-cell proofs.

Lemma 1 (reduction lemma). *For any data structure problem f, if there exists an (s, b, t)-CPP, then there exists an $(s^t, bt, 1)$-CPP.*

Proof. Just store every t-tuple of cells in the (s, b, t)-CPP as a new cell in the $(s^t, bt, 1)$-CPP. □

3 Characterization of CPPs

We now introduce a combinatorial characterization of CPP. Given a set system $\mathcal{F} \subseteq 2^Y$, for any $y \in Y$, we let $\mathcal{F}(y) = \{F \in \mathcal{F} \mid y \in F\}$. For convenience, for a partition $\mathcal{P}$ of Y, we abuse this notation and let $\mathcal{P}(y)$ denote the set $F \in \mathcal{P}$ that $y \in F$.

Definition 1. *We say a set system $\mathcal{F} \subseteq 2^Y$ is an $s \times k$-partition of Y, if $\mathcal{F}$ is a union of s number of partitions of Y, where the cardinality of each partition is at most k.*

This particular notion of partitions of Y fully captures the structure of cell-probe proofs. In this extended abstract, we only provide the characterization of 1-cell proofs. As shown by Lemma 1, this is a canonical case for cell-probe proofs.

Theorem 1. *There is an $(s, b, 1)$-CPP for $f : X \times Y \to \{0, 1\}$, if and only if there exists an $s \times 2^b$-partition $\mathcal{F}$ of Y, such that for every $x \in X$ and every $y \in Y$, there is an $F \in \mathcal{F}(y)$ that $|f(x, F)| = 1$.*

Proof. ($\Longrightarrow$) Given a table structure $T : Y \times I \to \{0,1\}^b$, define the map of the table structure as a $s \times 2^b$ matrix M such that $M_{ij} = \{y \in Y \mid T_y(i) = j\}$, i.e. M_{ij} is the set of such data set y that the content of the i's cell of the table storing y is j. It is clear that each row i of M is a partition of Y with at most 2^b partition sets, because each data set y has one and only one value of $T_y(i)$, and there are at most 2^b possible values for a cell, therefore the matrix M is an $s \times 2^b$-partition $\mathcal{F}$ of Y, where each M_{ij} is an $F \in \mathcal{F}$.

If there is an $(s, b, 1)$-CPP of f, due to the completeness of CPP, for every $x \in X$ and every $y \in Y$, there exists a cell i that $\langle i, j\rangle$ becomes the certificate where $j = T_y(i)$, and due to the soundness of CPP, there must not be any other $y' \in Y$ such that $T_{y'}(i) = j$ and $f(x, y') \neq f(x, y)$. Note that by definition of M, M_{ij} contains all y' such that $T_{y'}(i) = j$, thus $|f(x, M_{ij})| = 1$.

($\Longleftarrow$) Assuming that $\mathcal{F}$ is an $s \times 2^b$-partition of Y such that for every x and every y there is an $F \in \mathcal{F}(y)$ that $|f(x, F)| = 1$, we rewrite $\mathcal{F}$ in the form of an $s \times 2^b$ matrix M that M_{ij} is the $F \in \mathcal{F}$ which is indexed as the jth partition set in the ith partition. We can define our table structure $T : Y \times I \to \{0,1\}^b$ in the way that $T_y(i)$ is assigned with the unique j that $y \in M_{ij}$. Because each row of M is a partition of Y, such T is well-defined.

For every $x \in X$ and every $y \in Y$, there is an $F \in \mathcal{F}(y)$ that $|f(x, F)| = 1$, i.e. there is an $M_{ij} \ni y$ that $|f(x, M_{ij})| = 1$, then we use $\langle i, j\rangle$ as the certificate. Since for every x and y, there exists such i, the corresponding CPP is complete, and since $f(x, \cdot)$ is constant on such M_{ij}, the CPP is also sound. $\square$

Let $Y_0^x = \{y \in Y \mid f(x, y) = 0\}$ and $Y_1^x = \{y \in Y \mid f(x, y) = 1\}$. An alternative characterization is that there is a $(s, b, 1)$-CPP for a problem $f : X \times Y \to \{0,1\}$, if and only if there exists an $s \times 2^b$-partition $\mathcal{F}$ of Y, such that $\{Y_0^x, Y_1^x\}_{x \in X}$ is contained by the union-closure of $\mathcal{F}$. Note that this statement is equivalent to the statement in Theorem 1, so we state it without proof. With this formulation, we get some intuition about 1-cell proofs, that is, a problem $f : X \times Y \to \{0,1\}$ has simple proofs, if and only if there exists some set system $\mathcal{F} \subseteq 2^Y$ with a simple structure, such that the complexity of $\mathcal{F}$ matches the complexity of the problem.

4 Nearest neighbor search

We consider the decision version of nearest neighbor search, λ-near neighbor (λ-NN), in a high dimensional Hamming cube $\{0,1\}^d$. Here $X = \{0,1\}^d$, $Y = \binom{\{0,1\}^d}{n}$ and $f(x, y) \in \{0,1\}$ answers whether there exists a point in y within distance λ from the x. As in [3, 1], we assume that $d = \omega(\log n) \cap n^{o(1)}$ to make the problem non-trivial.

We prove that with the above setting, there does not exist a $(\mathrm{Poly}(n), n^{o(1)}, 1)$-CPP for the λ-NN problem, thus due to Lemma 1, a super-constant lower bound holds for the problem. To show this, we show the same lower bound for the partial match problem[9, 8], which is an instantiation of the λ-NN problem as shown in [3].

The partial match problem is defined as follow: The domain is a Hamming cube $\{0,1\}^d$, where $d = \omega(\log n) \cap n^{o(1)}$, and each data instance y is a set of n points from the domain, i.e. $Y = \binom{\{0,1\}^d}{n}$. The set of queries is $X = \{0, 1, *\}^d$. Given a data instance $y \in \binom{\{0,1\}^d}{n}$ and a query $x \in \{0, 1, *\}^d$, $f(x, y) = 1$ if and only if there is a $z \in y$ such that z matches x except for the bits assigned with "$*$".

Theorem 2. *There is no $(s, b, 1)$-CPP for the partial match problem, if $s =$ Poly(n) and $b = n^{o(1)}$.*

Proof. We denote the problem as f. From the characterization of $(s, b, 1)$-CPP given in Theorem 1, it is sufficient to show that for any $s \times 2^b$ partition $\mathcal{F}$ of Y, there exist $x \in X$ and $y \in Y$ such that for all $F \in \mathcal{F}(y)$, $|f(x, F)| = 2$. We prove this with the probabilistic method. With some distribution of x and y, we show that for any $s \times 2^b$ partition $\mathcal{F}$ of Y, $\Pr[\forall F \in \mathcal{F}(y), |f(x, F)| = 2] > 0$.

For the rest of the proof, we assume that y is uniformly selected from Y, and x is generated by uniformly choosing $r = 2 \log n$ bits and fixing each of them uniformly and independently at random with 0 or 1, and setting the other bits to "$*$".

We then prove two supporting lemmas. Recall that for a partition $\mathcal{P}$ of Y, $\mathcal{P}(y)$ denotes the set $F \in \mathcal{P}$ that $y \in F$.

Lemma 2. *For any partition $\mathcal{P}$ of Y, if $|\mathcal{P}| \le 2^b$, where $b = n^{o(1)}$, then*

$$\Pr_y \left[|\mathcal{P}(y)| \le \binom{2^d}{n} \Big/ 2^{n^{\Omega(1)}} \right] \le n^{-\omega(1)} .$$

Proof. We let $\mathcal{P} = \{F_1, F_2, \ldots, F_k\}$ where $k \le 2^b$, and let $p_i = |F_i|/|Y|$. Because $\mathcal{P}$ is a partition of Y, we know that $\sum_i p_i = 1$. We define a random variable $Z = |\mathcal{P}(y)|/|Y|$. Since y is picked uniformly at random from Y, it holds that $Z = p_i$ with probability p_i. Since there are at most 2^b different $\mathcal{P}(y)$, by union bound,

$$\begin{aligned}
\Pr_y \left[|\mathcal{P}(y)| \le \binom{2^d}{n} \Big/ 2^{n^{\Omega(1)}} \right] &\le 2^b \cdot \Pr\left[Z = p_i \text{ where } p_i \le 2^{-n^{\Omega(1)}} \right] \\
&= 2^{b - n^{\Omega(1)}} \\
&= n^{-\omega(1)} .
\end{aligned}$$

□

For simplicity, we generalize the notation of f to arbitrary point set $A \subseteq \{0,1\}^d$, where $f(x, A)$ is conventionally defined to indicate whether there is a $z \in A$ that matches x

Lemma 3. *For any $A \subseteq \{0,1\}^d$, if $|A| > (1 - 2^{-k})2^d$ for $k = \frac{1}{2} \log n$, then*

$$\Pr_x [f(x, A) = 0] \le n^{-\omega(1)} .$$

Proof. We let $B = \{0,1\}^d \setminus A$ be the complement of A in the d-dimensional cube. Note that $|B| < 2^{d-k}$. According to our definition of the distribution of x, x is in fact a random $(d-r)$-dimensional subcube in $\{0,1\}^d$, and $f(x, A) = 0$ only if the cube specified by x is contained in B. This chance is maximized when B itself is a cube. Thus without loss of generality, we can assume that B is the set of $z \in \{0,1\}^d$ whose first k bits are all ones. Therefore,

$$\Pr_x[f(x,A) = 0] \le \Pr_x[x\text{'s first } k \text{ bits are all ones}] \le \frac{\binom{d-k}{r-k}}{\binom{d}{r}} \le \left(\frac{r}{d}\right)^k = n^{-\omega(1)} .$$

□

We then prove that for all $s \times 2^b$ partitions $\mathcal{F}$ of Y, the probability $\Pr[\exists F \in \mathcal{F}(y), f(x,F) = \{1\}]$ and $\Pr[\exists F \in \mathcal{F}(y), f(x,F) = \{0\}]$ are both very small.

For any $F \in \mathcal{F}(y)$, we have $y \in F$, thus $\exists F \in \mathcal{F}(y), f(x,F) = \{1\}$ implies that $f(x,y) = 1$, therefore for an arbitrary $s \times 2^b$ partition $\mathcal{F}$ of Y,

$$\begin{aligned}\Pr_{x,y}[\exists F \in \mathcal{F}(y), f(x,F) = \{1\}] &\le \Pr_{x,y}[f(x,y) = 1] \\ &\le \Pr_{x,y}[\exists z \in y, x \text{ matches } z] \\ &\le n \cdot 2^{-r} \\ &= o(1) .\end{aligned}$$

To bound the probability $\Pr[\exists F \in \mathcal{F}(y), f(x,F) = \{0\}]$, we observe that each $s \times 2^b$ partition $\mathcal{F}$ is just a union of s many partitions of Y, each of which is with cardinality at most 2^b, therefore, by union bounds, it holds that

$$\Pr_{x,y}[\exists F \in \mathcal{F}(y), f(x,F) = \{0\}] \le s \cdot \Pr_{x,y}[f(x, \mathcal{P}(y)) = \{0\}] . \qquad (1)$$

for some partition $\mathcal{P}$ of Y where $|\mathcal{P}| \le 2^b$. It is then sufficient to show that for arbitrary such partition $\mathcal{P}$, the probability $\Pr[f(x, \mathcal{P}(y)) = \{0\}]$ is very small.

We choose a threshold $k = \frac{1}{2}\log n$, and separate the case that $|\mathcal{P}(y)| \le \binom{(1-2^{-k})2^d}{n}$ and the case that $|\mathcal{P}(y)| > \binom{(1-2^{-k})2^d}{n}$. According to Lemma 2, for any partition $\mathcal{P}$ of Y with $|\mathcal{P}| \le 2^b$, the probability that $|\mathcal{P}(y)| \le \binom{(1-2^{-k})2^d}{n} = \binom{2^d}{n}/2^{n^{\Omega(1)}}$ is at most $n^{-\omega(1)}$.

We let $A_y = \bigcup \mathcal{P}(y) = \bigcup_{y' \in \mathcal{P}(y)} y'$. Note that $A_y \subseteq \{0,1\}^d$, and $f(x, \mathcal{P}(y)) = \{0\}$ implies that $f(x, A_y) = 0$. For such $\mathcal{P}(y)$ that $|\mathcal{P}(y)| > \binom{(1-2^{-k})2^d}{n}$, by the Pigeonhole Principle, it holds that $|A_y| \ge (1-2^{-k})2^d$. Due to Lemma 3, $f(x, A_y) = 0$ with prohibitively small probability. Putting these together, it holds for an arbitrary partition $\mathcal{P}$ of Y with $|\mathcal{P}| \le 2^b$ that

$$\begin{aligned}\Pr_{x,y}[f(x,\mathcal{P}(y)) = \{0\}] &\le \Pr_{y}\left[|\mathcal{P}(y)| \le \binom{(1-2^{-k})2^d}{n}\right] \\ &\quad + \Pr_{x,y}\left[f(x,\mathcal{P}(y)) = \{0\} \;\middle|\; |\mathcal{P}(y)| > \binom{(1-2^{-k})2^d}{n}\right] \\ &\le n^{-\omega(1)} + \Pr_{x}\left[f(x,A_y) = 0 \;\middle|\; |A_y| > (1-2^{-k})2^d\right] \\ &\le n^{-\omega(1)}\,.\end{aligned}$$

Combining with (1), we have that

$$\Pr_{x,y}[\exists F \in \mathcal{F}(y), f(x,F) = \{0\}] \le s \cdot n^{-\omega(1)} = o(1)\,.$$

Therefore, for an arbitrary $s \times 2^b$ partition $\mathcal{F}$ of Y, it holds that

$$\begin{aligned}\Pr_{x,y}[\forall F \in \mathcal{F}(y), |f(x,F)| = 2] &\ge 1 - \Pr_{x,y}[\exists F \in \mathcal{F}(y), f(x,F) = \{1\}] \\ &\quad - \Pr_{x,y}[\exists F \in \mathcal{F}(y), f(x,F) = \{0\}] \\ &\ge 1 - o(1)\,.\end{aligned}$$

It follows that for any $s \times 2^b$ partition $\mathcal{F}$ of Y, where $s = \mathrm{Poly}(n)$ and $b = n^{o(1)}$, there exist $x \in X$ and $y \in Y$ such that for every $F \in \mathcal{F}(y)$, it holds that $|f(x,F)| = 2$. By Theorem 1, there is no $(s,b,1)$-CPP for f with the above range of s and b. □

In [3], it is shown that the partial match problem can be reduced to the λ-NN problem. Because the reduction only involves mapping between instances of problems, the existence of an $(s,b,1)$-CPP for λ-NN implies the existence of a CPP for partial match with essentially the same parameters. The following corollary is implied.

Corollary 1. *There does not exist a $(Poly(n), n^{o(1)}, 1)$-CPP for the nearest neighbor search problem with n points in d-dimensional Hamming space where $d = \omega(\log n) \cap n^{o(1)}$.*

Due to Lemma 1, the following super-constant lower bound on the nondeterministic cell-probe complexity holds.

Corollary 2. *There does not exist a $(Poly(n), n^{o(1)}, O(1))$-CPP for the nearest neighbor search problem or the partial match problem with n points in d-dimensional Hamming space where $d = \omega(\log n) \cap n^{o(1)}$.*

5 Polynomial evaluation

Let 2^k be a finite field. Let $Y = 2^{kd}$ be the set of all polynomials of degree $\le (d-1)$ over the finite field 2^k. Throughout this section, we assume that $d \le 2^k$.

Let $X = 2^{2k}$ be the set of all pairs of elements of the finite field 2^k. A decision version of the polynomial evaluation problem f is defined as: for every query $(x, z) \in X$ and every data instance $g \in Y$, $f((x, z), g) = 1$ if $g(x) = z$ and $f((x, z), g) = 0$ otherwise. Intuitively, a polynomial g is preprocessed and stored as a data structure, so that for each query (x, z), the data structure answers whether $g(x) = z$.

There are two naive upper bounds for one-cell proofs:

1. A $(1, kd, 1)$-CPP: store the whole polynomial in a single cell, and on each query, one probe reveals the whole polynomial;
2. A $(2^k, k, 1)$-CPP: each cell corresponds to an input x, and the cell stores the value of $g(x)$, thus on each query (x, z), one probe to the cell corresponding to x answers whether $g(x) = z$.

We are going to prove that the above naive upper bounds are essentially optimal for single-probe proofs. We show that for any $(s, b, 1)$-CPP, either b is close to large enough to store a whole polynomial as in (1), or the total storage size $s \cdot b$ is exactly as large as in (2).

We first prove two lemmas. For any subset $P \subseteq Y$, let $\tau(P) = |\{x \in 2^k \mid \forall g_1, g_2 \in P, g_1(x) = g_2(x)\}|$, which represents the number of such assignments of x that all polynomials in P yield the same outcome. It is trivial to see that for $|P| \le 1$, $\tau(P) = 2^k$.

Lemma 4. *If $|P| > 1$, it holds that*

$$\tau(P) \le d - \frac{\log |P|}{k}\,.$$

Proof. We write $\tau(P)$ briefly as τ. Let $x_1, x_2, \ldots, x_\tau$ be such that all polynomials in P yield the same outcomes. We arbitrarily pick other $x_{\tau+1}, x_{\tau+2}, \ldots, x_d$. For any two different polynomials $g_1, g_2 \in P$, it can never hold that $g_1(x_i) = g_2(x_i)$ for all $i = \tau + 1, \tau + 2, \ldots, d$, since if otherwise, $g_1 \equiv g_2$ by interpolation. Recall that g is a polynomial over the finite field 2^k, thus for an arbitrary $g \in P$ and an arbitrary x, there are at most 2^k possible values for $g(x)$. Therefore, due to Pigeonhole Principle, in order to guarantee that no two polynomials in P agree on all $x_{\tau+1}, x_{\tau+2}, \ldots, x_d$, it must hold that $2^{k(d-\tau)} \ge |P|$, i.e. $\tau(P) \le d - \frac{\log |P|}{k}$. □

Lemma 5. *Given a partition $\mathcal{P}$ of Y, let g be a uniformly random polynomial in Y. $E\{\tau(\mathcal{P}(g))\}$ represents the expected number of the input xs such that all polynomials in the partition block $\mathcal{P}(g)$ yield the same outcome, where the expectation is taken over random g. For any partition $\mathcal{P}$ of Y such that $|\mathcal{P}| \le 2^b$ and $b < k(d-1) - \log k$, it holds that*

$$E\{\tau(\mathcal{P}(g))\} \le \frac{b}{k}\,.$$

Proof. Let $P_1, P_2, \dots, P_{2^b}$ denote the partition blocks, and let $q_1, q_2, \dots, q_{2^b}$ be the respective cardinalities. Naturally we have that $\sum_{i=1}^{2^b} q_i = 2^{kd}$. We assume that $q_i = 0$ for $i = 1, 2, \dots, m_0$, $q_i = 1$ for $i = m_0 + 1, m_0 + 2, \dots, m$, and $q_i > 1$ for $i > m$. For those P_i that $i \le m$, $|P_i| = q_i \le 1$, thus $\tau(P_i) = 2^k$. According to Lemma 4,

$$\begin{aligned} E\{\tau(\mathcal{P}(g))\} &= \sum_{i=1}^{m_0} \frac{0}{2^{kd}} \tau(P_i) + \sum_{i=m_0+1}^{m} \frac{1}{2^{kd}} \tau(P_i) + \sum_{i=m+1}^{2^b} \frac{q_i}{2^{kd}} \tau(P_i) \\ &\le (m - m_0) \cdot \frac{2^k}{2^{kd}} + \sum_{i=m+1}^{2^b} \frac{q_i}{2^{kd}} \left(d - \frac{\log q_i}{k} \right) . \qquad (2) \end{aligned}$$

Recall that $\sum_{i=m+1}^{2^b} q_i = 2^{kd} - \sum_{i=1}^{m} q_i = 2^{kd} - m + m_0$. According to Lagrange multipliers, (2) is maximized when all q_i for $i = m+1, m+2, \dots, 2^b$ are equal. Thus (2) is less than or equal to

$$\frac{m - m_0}{2^{k(d-1)}} + \frac{2^{kd} - m + m_0}{2^{kd}} \left(d - \frac{\log(2^{kd} - m + m_0) - \log(2^b - m)}{k} \right) .$$

Let $\epsilon = \frac{m - m_0}{2^{kd}}$. The above formula becomes

$$\begin{aligned} &2^k \epsilon + (1 - \epsilon) \left(d - \frac{\log 2^{kd}(1 - \epsilon) - \log 2^b(1 - 2^{-b}(2^{kd}\epsilon + m_0))}{k} \right) \\ &\le 2^k \epsilon + \frac{1}{k}(1 - \epsilon) \left(b + \log(1 - 2^{kd-b}\epsilon) - \log(1 - \epsilon) \right) . \end{aligned}$$

Note that $0 \le \epsilon < 2^{b-kd}$. By standard analysis, if $b < k(d-1) - \log k$, the above function of ϵ is maximized when $\epsilon = 0$, i.e. $E\{\tau(\mathcal{P}(g))\} \le \frac{b}{k}$. □

With the above lemmas, we can prove the following theorem.

Theorem 3. *For any $(s, b, 1)$-CPP for the polynomial evaluation problem with parameters k and d where $d \le 2^k$, either $b \ge k(d-1) - \log k$ or $s \cdot b \ge k \cdot 2^k$.*

Proof. We will prove that there does not exist an $(s, b, 1)$-CPP for the polynomial evaluation problem if $b < k(d-1) - \log k$ and $s \cdot b < k \cdot 2^k$.

Let x be a uniformly random element of 2^k, and let g be a uniformly random polynomial from Y. For any partition $\mathcal{P}$ of Y that $|\mathcal{P}| \le 2^b$, according to Lemma 5,

$$\Pr_{x,g}[\forall g_1, g_2 \in \mathcal{P}(g), g_1(x) = g_2(x)] = \frac{1}{2^k} E\{\tau(\mathcal{P}(g))\} \le \frac{b}{k 2^k} .$$

Therefore, for any $s \times 2^b$ partition $\mathcal{F}$ of Y, it holds that,

$$\begin{aligned} &\Pr_{x,g}[\exists F \in \mathcal{F}(g), \forall g_1, g_2 \in F, g_1(x) = g_2(x)] \\ &\le s \cdot \Pr_{x,g}[\forall g_1, g_2 \in \mathcal{P}(g), g_1(x) = g_2(x)] \\ &\le \frac{s \cdot b}{k 2^k} \\ &< 1 , \end{aligned}$$

where the first inequality is due to the observation that $\mathcal{F}$ is a union of s instances of 2^b-partitions of Y. Therefore, for any $s \times 2^b$ partition $\mathcal{F}$ of Y,

$$\Pr_{x,g}[\forall F \in \mathcal{F}(g) \exists g_1, g_2 \in F, g_1(x) \neq g_2(x)] > 0\,.$$

By probabilistic methods, we know that for any $s \times 2^b$ partition $\mathcal{F}$ of Y, there exists some $(x, z) \in X$ and some $g \in Y$ such that $g(x) = z$, but for all $F \in \mathcal{F}(g)$, there exists $h \in F$ such that $h(x) \neq z$.

According to Theorem 1, we know that there does not exist $(s, b, 1)$-CPP with the given range of s and b. $\square$

Acknowledgment. I would like to thank James Aspnes for helpful discussions and editing assistance, and Dana Angluin for her comments on an early version of the paper.

References

1. Barkol, O., Rabani, Y.: Tighter lower bounds for nearest neighbor search and related problems in the cell probe model. Journal of Computer and System Sciences **64**(4) (2002) 873–896
2. Beame, P., Fich, F.: Optimal bounds for the predecessor problem and related problems. Journal of Computer and System Sciences **65**(1) (2002) 38–72
3. Borodin, A., Ostrovsky, R., Rabani, Y.: Lower bounds for high dimensional nearest neighbor search and related problems. Proceedings of the thirty-first annual ACM Symposium on Theory of Computing (1999) 312–321
4. Buhrman, H., de Wolf, R.: Complexity measures and decision tree complexity: a survey. Theoretical Computer Science **288**(1) (2002) 21–43
5. Demaine, E., Pătraşcu, M.: Logarithmic lower bounds in the cell-probe model. SIAM Journal of Computing **35**(4) (2006) 932–963
6. Fredman, M., Saks, M.: The cell probe complexity of dynamic data structures. Proceedings of the twenty-first annual ACM Symposium on Theory of Computing (1989) 345–354
7. Husfeldt, T., Rauhe, T.: Hardness results for dynamic problems by extensions of Fredman and Saks' chronogram method. Proceedings of the 25th International Colloquium on Automata, Languages and Programming (1998) 67–78
8. Indyk, P., Goodman, J., O'Rourke, J.: Nearest neighbors in high-dimensional spaces. Handbook of Discrete and Computational Geometry, chapter 39 (2004)
9. Jayram, T., Khot, S., Kumar, R., Rabani, Y.: Cell-probe lower bounds for the partial match problem. Journal of Computer and System Sciences **69**(3) (2004) 435–447
10. Miltersen, P., Nisan, N., Safra, S., Wigderson, A.: On data structures and asymmetric communication complexity. Journal of Computer and System Sciences **57**(1) (1998) 37–49
11. Yao, A.: Should tables be sorted? Journal of the ACM **28**(3) (1981) 615–628