Cell-Probe Proofs

YITONG YIN
Nanjing University

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 verification instead of computation 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.

—There exists a data structure problem with high nondeterministic cell-probe complexity.

—For the exact nearest neighbor search (NNS) problem or the partial match problem in high dimensional Hamming space, for any data structure with Poly(n) cells, each of which contains $O\left(n^C\right)$ bits where $C < 1$, the nondeterministic cell-probe complexity is at least $\Omega\left(\log(d/\log n)\right)$, where d is the dimension and n is the number of points in the data set.

—For the polynomial evaluation problem of d-degree polynomial over finite field of size 2^k where $d \leq 2^k$, for any data structure with s cells, each of which contains b bits, the nondeterministic cell-probe complexity is at least $\min\left(\frac{k}{b}(d-1), \frac{k-\log(d-1)}{\log s}\right)$.

Categories and Subject Descriptors: E.1 [**Data**]: Data Structures; F.1.2 [**Computation by Abstract Devices**]: Modes of Computation

General Terms: Theory

Additional Key Words and Phrases: Cell-probe model, nondeterminism

ACM Reference Format:
Yin, Y. 2010. Cell-probe proofs. ACM Trans. Comput. Theory, 2, 1, Article 1 (November 2010), 17 pages. DOI = 10.1145/1867719.1867720. http://doi.acm.org/10.1145/1867719.1867720.

1. INTRODUCTION

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

This work was supported by the Program for New Century Excellent Talents at University (NCET) and by the National Science Foundation of China under grant 60721002.
A preliminary version of this article appeared in the *Proceedings of the 35th International Colloquium on Automata, Languages and Programming*, pp. 72–83.
Author's address: Y. Yin, Nanjing University, Department of Computer Science and Technology, 22 Hankou Road, Nanjing, Jiangsu, China 210093; email: yinyt@nju.edu.cn.

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) [Barkol and Rabani 2002; Borodin et al. 1999; Indyk et al. 2004]: 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.

Partial match [Borodin et al. 1999; Jayram et al. 2004; Pătraşcu 2008]: $X = \{0, 1, *\}^d$, $Y = \binom{\{0,1\}^d}{n}$, 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 [Miltersen 1995]: $X = 2^k$ 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 Yao [1981]. 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 space. We speculate that it is an important problem with following motivations:

(1) In considering the complexity of data structures, nondeterminism is a very natural extension to the cell-probe model. Instead of adaptive computations, nondeterministic cell-probes capture the notion of verification, which is a natural and important aspect of data structures.

(2) A classic tool for analyzing the cell-probe model is communication complexity [Kushilevitz and Nisan 1997; Miltersen et al. 1998], for which the computations in a data structure are viewed as communications between two adaptive players. This observation makes computations in data structures analyzable by granting the data structure extra power (i.e., an adaptive table). By the same light, assuming nondeterminism provides us a different way of proving lower bounds: instead of having a table that can "talk," it assumes a probing algorithm that can "guess".

Although nondeterministic cell-probe complexity is an important problem, the optimal nondeterministic bounds for static data structure problems are still unknown. For many naturally defined data structure problems, nondeterminism trivializes the problem in the sense that a constant number of nondeterministic cell-probes are sufficient to answer the queries. For the problems that remain nontrivial after allowing nondeterminism, no lower bounds are known for the 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 relatively high. 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 article, we initiate the study of nondeterministic cell-probe complexity for static data structures.

We introduce cell-probe proofs, a proof system in the cell-probe model. This notion of proof corresponds to considering verification instead of computation in the cell-probe model. Unlike the fully adaptive computation 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 cell-probe proofs with specified parameters.

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

—There exists a data structure problem with high nondeterministic cell-probe complexity. This result can be seen as a nondeterministic parallel to the existential lower bound for deterministic cell-probe complexity in Miltersen [1999], which is proved by counting. As mentioned in Miltersen [2008], the nondeterministic lower bound cannot be easily proved by the similar counting argument.

—For the exact nearest neighbor search (NNS) problem or the partial match problem in high dimensional Hamming space, for any data structure with Poly(n) cells, each of which contains $O(n^C)$ bits where $C < 1$, the nondeterministic cell-probe complexity is at least $\Omega\left(\log\left(d/\log n\right)\right)$, where d is the dimension and n is the number of points in the data set. The highest known deterministic cell-probe lower bound for the problems for polynomial space is $\Omega(d/\log n)$ (see Barkol and Rabani [2002] and Pătraşcu [2008]). These lower bounds are proved by communication complexity techniques.

—For the polynomial evaluation problem of d-degree polynomial over a finite field of size 2^k where d is high, for any data structure with s cells, each of which contains b bits, the nondeterministic cell-probe complexity is at least $\min\left(\frac{k}{b}(d-1), \frac{k-\log(d-1)}{\log s}\right)$. This bound is nearly equal to the

$\left(\min\left(d+1, \frac{k-\log d}{\log s}\right)\right)$ deterministic lower bound [Miltersen 1995], in which it is assumed that $b = k$.

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 nontrivial 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. [1998].

Fredman and Saks [1989], introduce the *chronogram method*. This powerful technique is specialized for proving the query/update tradeoff for dynamic data structures, especially for the problems that 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 [1998], and recently by Pătraşcu and Demaine [2006]. However, as pointed in Husfeldt and Rauhe [1998], this is only a byproduct of the nondeterministic nature of chronogram method, and can only yield amortized query/update tradeoffs for dynamic data structure problems with a certain property. Due to the unique structure of the chronogram method, this technique cannot be utilized to prove lower bounds for static data structures.

2. CELL-PROBE PROOFS

A static data structure problem is represented as a boolean function $f : X \times Y \to \{0, 1\}$. For the purposes of proving lower bounds, it is sufficient to consider only the 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., Fredman and Saks [1989]; Yao [1981]), 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 $f(x, y)$ can be uniquely decided on the query input x and the probed information.

Formally, a nondeterministic cell-probing algorithm is a function A which maps the query input x and the probed cells to the result value or $\perp$, such that for any query x and data y,

(1) there exists a set of t cells $i_1, i_2, \ldots, i_t$, such that

$$A\big(x, \langle i_1, T_y(i_1)\rangle, \langle i_2, T_y(i_2)\rangle, \ldots, \langle i_t, T_y(i_t)\rangle\big) = f(x, y);$$

(2) for any set of t cells $i_1, i_2, \ldots, i_t$, it holds that

$$A\big(x, \langle i_1, T_y(i_1)\rangle, \langle i_2, T_y(i_2)\rangle, \ldots, \langle i_t, T_y(i_t)\rangle\big) \neq \overline{f(x, y)}.$$

In order to formally characterize nondeterministic cell-probes for data structures, we introduce a new concept, *cell-probe proofs*, which formalizes the notion of proof and verification 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 in the following.

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 cannot 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:

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

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

Example. For the membership problem [Yao 1981], where $X = [m]$ and $Y = \binom{[m]}{n}$, and $f(x, y) = 1$ if and only if $x \in y$, a naive construction shows there is 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 [Beame and Fich 2002].

The fact that this simple example also works for predecessor has an important implication. Combining with the super-constant lower bound for predecessor search [Pătraşcu and Thorup 2006], this gives an example where one can separate deterministic and nondeterministic complexity.

We remark that cell-probe proofs characterize the nondeterministic probes in the cell-probe model. Specifically, the verifier v is just a nondeterministic cell-probing algorithm. 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 [Buhrman and de Wolf 2002]. 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.

We should also be specific that the model of cell-probe proofs prohibits randomization. The nondeterministic model for randomized data structures is a possible future direction.

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 2.1 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 3.1. 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 many 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. The following theorem provides a full characterization of cell-probe proofs.

THEOREM 3.2. *There is an (s, b, t)-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 exist $F_1, \ldots, F_t \in \mathcal{F}(y)$ such that $\left| f(x, \cap_{i=1}^{t} F_i) \right| = 1$.*

PROOF. First, we prove the "only if" part.

Let $T : Y \times [s] \to \{0, 1\}^b$ be the table structure in the (s, b, t)-CPP. Define the map of the table structure as an $s \times 2^b$ matrix M such that $M_{ij} = \{y \in Y \mid T_y(i) = j\}$, that is, M_{ij} is the set of all such data instances y that the content of the i-th cell of the table is j, assuming that the data instance is y. It is easy to verify that $\{M_{ij}\}$ is an $s \times 2^b$-partition of Y.

We then show that for every $x \in X$ and $y \in Y$, there must exist $i_1, i_2, \ldots, i_t$ such that $\left| f\left(x, \cap_{k=1}^{t} M\left(i_k, j_k\right)\right)\right| = 1$, where $j_k = T_y(i_k)$. To the contrary, we assume that for some x, there exists such y that for all $i_1, i_2, \ldots, i_t$, there always exists $y' \in \cap_{k=1}^{t} M\left(i_k, T_y(i_k)\right)$ such that $f(x, y) \neq f(x, y')$. According to the definition of M, it implies that for any $i_1, i_2, \ldots, i_t$, there exists a y' that shares the same certificate $\left\{\left\langle i_k, T_y(i_k)\right\rangle \mid k = 1, 2, \ldots, t\right\}$ with y, but $f(x, y') \neq f(x, y)$. Due to the completeness of CPP, the verifier maps one of the certificates $\left\{\left\langle i_k, T_y(i_k)\right\rangle \mid k = 1, 2, \ldots, t\right\}$ of y to $f(x, y)$, but in this case the adversary can choose y' as the real data instance, contradicting the soundness of the CPP.

We then deal with the "if" part.

Let $\mathcal{F}$ be 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 $\left|f(x, F)\right| = 1$. We rewrite $\mathcal{F}$ as an $s \times 2^b$ matrix M such that M_{ij} is indexed as the jth partition set in the ith partition in $\mathcal{F}$, that is, $\{M_{ij}\}_{i,j} = \mathcal{F}$ and for any i, $\{M_{ij}\}_j$ is a partition of Y.

We can define the table structure $T : Y \times [s] \to \{0, 1\}^b$ as follows: $T_y(i)$ is assigned with the unique j that $y \in M_{ij}$.

The verifier is defined as follows: for any $x \in X$ and any certificate $\{\langle i_k, j_k\rangle \mid k = 1, 2, \ldots, t\}$, if $f(x, \cdot)$ is constant on $\cap_{k=1}^{t} M(i_k, j_k)$, then the verifier returns that value, otherwise it returns "$\perp$". It is easy to verify that both the completeness and soundness of CPP are satisfied. $\square$

For all CPPs, one-cell proofs are the simplest nontrivial proofs. As we explained in the last section, the case of a one-cell proof is also very important because all CPPs can be reduced to it. We apply the above theorem to the one-cell case in the following corollary.

COROLLARY 3.3. *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 $|f(x, F)| = 1$.*

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 $\mathcal{F}$. It can easily be noted that this characterization is equivalent to the one in Corollary 3.3. 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. AN EXISTENTIAL LOWER BOUND

In this section we prove an existential lower bound for cell-probe proofs by showing that uniformly random data structure problem has no efficient CPP with high probability, which means that for most data structure problems, there does not exist efficient nondeterministic data structures. Like the existential lower bound for *deterministic* data structures in Miltersen [1999], our lower bound is in the form of bit-probe complexity, that is, each cell contains only one bit.

We first introduce some notations.

—Given a set family $\mathcal{F}$, we denote that

$$I_t(\mathcal{F}) = \left\{ \bigcap_{i=1}^{t} A_i \mid A_1, A_2, \ldots, A_t \in \mathcal{F} \right\}.$$

We call $I_t(\mathcal{F})$ the t-intersection closure of $\mathcal{F}$.

—Similarly, we define the union closure of a set system $\mathcal{F}$ as $\bigcup^* \mathcal{F} = \left\{ \bigcup_{A \in \mathcal{G}} A \mid \mathcal{G} \subseteq \mathcal{F} \right\}$, which is a set of all the unions of the members of $\mathcal{F}$.

—For a fixed a set Y of data instances, we define the parameter $\chi(s,t)$ as the smallest integer that for every $s \times 2$-partition $\mathcal{F}$ of Y, $\chi(s,t) \geq \frac{1}{2} \left| \bigcup^* I_t(\mathcal{F}) \right|$. The value $\chi(s,t)$ depends on s, t, and the size of Y.

The following lemma shows a relation between the the parameter $\chi(s,t)$ and the existence of hard problems.

LEMMA 4.1. *Let $f : X \times Y \to \{0,1\}$ be a data structure problem where $X = \{0,1\}^m$ is the set of queries and $Y = \{0,1\}^n$ is the set of data instances. The following statements hold:*

(*1*) *With probability at least $1 - \left(\chi(s,t) 2^{-2^n}\right)^{2^m} 2^{s 2^n}$, there does not exists an $(s,1,t)$-CPP for uniformly random $f : X \times Y \to \{0,1\}$.*

(*2*) *If $\chi(s,t) < 2^{2^n (1 - s/2^m)}$, then there exists a problem f such that there does not exist an $(s,1,t)$-CPP for f.*

PROOF. We first prove 1. Then 2 follows naturally.

Let $\mathcal{F}$ be an $s \times 2$-partition of Y and $h : Y \to \{0,1\}$ be a uniformly random 2-coloring of Y. We evaluate the probability that for all $y \in Y$, there exists some $A \in I_t(\mathcal{F})$ such that $y \in A$ and h is constant over A. Let $Q(h)$ be a predicate defined on $h : Y \to \{0,1\}$ such that $Q(h)$ is true iff $\forall y \in Y, \exists A \in I_t(\mathcal{F})(y), |h(A)| = 1$. The probability is $\Pr_h[Q(h)]$.

If $Q(h)$ holds, it must hold that there exist $B_0, B_1 \in \bigcup^* I_t(\mathcal{F})$ such that $\{B_0, B_1\}$ is a bipartition of Y. In fact, given an $h : Y \to \{0,1\}$ such that $\forall y \in Y$, $\exists A_y \in I_t(\mathcal{F})(y)$, $\left|h(A_y)\right| = 1$, we can construct such B_0 and B_1 as

$$B_0 = \bigcup_{y : h(y) = 0} A_y, \quad \text{and } B_1 = \bigcup_{y : h(y) = 1} A_y.$$

It is easy to check that $\{B_0, B_1\} \subset \bigcup^* I_t(\mathcal{F})$ is a bipartition of Y and $h(B_b) = \{b\}$ for $b \in \{0,1\}$.

Therefore, each h with the desirable property $Q(h)$ corresponds to a distinct pair of members in $\bigcup^* I_t(\mathcal{F})$. According to the definition of $\chi(s,t)$, the total number of the h that $Q(h)$ holds is upper bounded by $\chi(s,t)$. There are totally 2^{2^n} different $h : Y \to \{0,1\}$. Therefore,

$$\Pr_h[Q(h)] \leq \frac{1}{2}\left|\bigcup^* I_t(\mathcal{F})\right| / 2^{2^n} \leq \chi(s,t)2^{-2^n}.$$

Let f be a uniformly random function $f : X \times Y \to \{0,1\}$, it holds that

$$\Pr_f\left[\forall x \in X, Q(f(x,\cdot))\right] \leq \left(\Pr_h[Q(h)]\right)^{|X|} \leq \left(\chi(s,t)2^{-2^n}\right)^{2^m}.$$

There are at most 2^{s2^n} different $s \times 2$-partitions of Y. By union bound, for uniformly random $f : X \times Y \to \{0,1\}$, with probability at most $\left(\chi(s,t)2^{-2^n}\right)^{2^m} 2^{s2^n}$, there exists an $s \times 2$-partition $\mathcal{F}$ of Y such that for all $x \in X$ and $y \in Y$, there exists $A \in I_t(\mathcal{F})(y)$ such that $\left|f(x,A)\right| = 1$.

Due to Theorem 3.2, it holds that the probability that there exists an $(s,1,t)$-CPP for a uniformly random $f : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}$ is at most $(\chi(s,t)2^{-2^n})^{2^m} 2^{s2^n}$. Statement 1 is proved.

If $\chi(s,t) < 2^{2^n(1-s/2^m)}$, then $\left(\chi(s,t)2^{-2^n}\right)^{2^m} 2^{s2^n} < 1$. With positive probability, a uniformly random f has no $(s,1,t)$-CPP. There must exist a data structure problem without $(s,1,t)$-CPP. Statement 2 holds. □

The following lemma gives an estimation of the parameter $\chi(s,t)$.

LEMMA 4.2. *If*

$$\binom{s}{t}2^t \leq 2^n\left(1 - \frac{s}{2^m}\right),$$

then it holds that $\chi(s,t) < 2^{2^n(1-s/2^m)}$.

PROOF. For any $s \times 2$-partition $\mathcal{F}$ of Y, $I_t(\mathcal{F})$ is a $l \times k$-partition of Y, where $l \leq \binom{s}{t}$ and $k \leq 2^t$, thus $\left|I_t(\mathcal{F})\right| \leq \binom{s}{t}2^t$. Note that the cardinality of the union closure of a set system is upper bounded by the cardinality of its power set. Therefore, for any $s \times 2$-partition $\mathcal{F}$ of Y, it holds that $\frac{1}{2}\left|\bigcup^* I_t(\mathcal{F})\right| \leq 2^{-1+\binom{s}{t}2^t} < 2^{2^n(1-s/2^m)}$, which means that $\chi(s,t) < 2^{2^n(1-s/2^m)}$. □

Combining the above two lemmas, we have the following theorem.

THEOREM 4.3. *If* $\binom{s}{t}2^t \leq 2^n\left(1-\frac{s}{2^m}\right)$, *there exists a problem* $f : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}$ *such that there does not exist* $(s,1,t)$*-CPP for* f.

Applying the above theorem to specific s, t, m, and n, we can see that there exists nondeterministically hard problem for some usual settings of these parameters.

—$s = 2^m - 1$ and $t < \frac{n}{m}$; so the size of the data structure is one bit less than the naïve solution of storing answers to all queries, and the size of the proof is less than $\frac{1}{m}$ of the size of a raw data instance.

—The typical case for practically efficient data structures: $m = \omega(\log n) \cap o(n)$, $s = \mathrm{Poly}(n)$, and $t = o\left(\frac{n}{\log n}\right)$.

5. 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 $f(x,y) \in \{0,1\}$ answer whether there exists a point in y within distance λ from the x. As in Borodin et al. [1999] and Barkol and Rabani [2002], we assume that $d = \omega(\log n) \cap n^{o(1)}$ to make the problem nontrivial.

We prove that with the above setting, there does not exist a (s, b, t)-CPP for the λ-NN problem, if $s = \mathrm{Poly}(n)$, $b < n^{1-\alpha}$ for some positive constant α and $t = o\left(\log\left(d/\log n\right)\right)$. To prove this, we prove that the lower bound holds for the partial match problem [Indyk et al. 2004; Jayram et al. 2004; Pătraşcu 2008], which is an instantiation of the λ-NN problem as shown in Borodin et al. [1999].

The partial match problem is defined as follows: 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, that is, $Y = \binom{\{0,1\}^d}{n}$. 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 5.1. *There is no $(s, b, 1)$-CPP for the partial match problem if $s < (d/2\log n)^{\alpha \log n}$ and $b < n^{1-\alpha}$ for some constant $0 < \alpha < 1$.*

PROOF. We denote the problem as f. From the characterization of $(s, b, 1)$-CPP given in Theorem 3.3, 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 $F \in \mathcal{F}(y)$, $\left|f(x,F)\right| = 2$. 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, $\Pr[\forall F \in \mathcal{F}(y), \left|f(x,F)\right| = 2] > 0$.

For the rest of the proof, we assume that y is uniformly selected from Y and that x is generated by uniformly choosing $r = 1 + \log n$ 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 5.2. *For any partition $\mathcal{P}$ of Y, if $|\mathcal{P}| \le 2^b$, then for $k < \log n$ it holds that*

$$\Pr_y\left[\left|\mathcal{P}(y)\right| \le \binom{\left(1 - 2^{-k}\right)2^d}{n}\right] \le \exp\left(b \ln 2 - n2^{-k}\right).$$

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,

$$\begin{aligned}
\Pr_y\left[\left|\mathcal{P}(y)\right| \le \binom{\left(1-2^{-k}\right)2^d}{n}\right] &\le 2^b \cdot \Pr\left[Z = p_i \text{ where } p_i \le \frac{\binom{(1-2^{-k})2^d}{n}}{\binom{2^d}{n}}\right] \\
&\le \frac{2^b\binom{(1-2^{-k})2^d}{n}}{\binom{2^d}{n}} \\
&\le \frac{2^b\left(\left(1-2^{-k}\right)2^d\right)^n}{2^{dn}} \\
&\le \exp\left(b\ln 2 - n2^{-k}\right). \qquad \square
\end{aligned}$$

For simplicity, we generalize the notation of f to an 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 5.3. *For any $A \subseteq \{0,1\}^d$, if $|A| > \left(1-2^{-k}\right)2^d$ for some $k < \log n$, then*

$$\Pr_x[f(x,A)=0] \le \left(\frac{r}{d}\right)^k.$$

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,

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

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 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,

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

To bound the probability $\Pr\left[\exists F \in \mathcal{F}(y),\ f(x,F)=\{0\}\right]$, 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

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

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

We choose a threshold $k = \alpha \log n$, and separate the case that $\left|\mathcal{P}(y)\right| \leq \binom{(1-2^{-k})2^d}{n}$ and the case that $|\mathcal{P}(y)| > \binom{(1-2^{-k})2^d}{n}$. According to Lemma 5.2, for any partition $\mathcal{P}$ of Y with $|\mathcal{P}| \leq 2^b$, the probability that $\left|\mathcal{P}(y)\right| \leq \binom{(1-2^{-k})2^d}{n}$ is at most $\exp\left(b \ln 2 - n^{(1-\alpha)}\right)$.

We let $A = \bigcup \mathcal{P}(y) = \bigcup_{y' \in \mathcal{P}(y)} y'$. Note that $A \subseteq \{0,1\}^d$ and $f\left(x, \mathcal{P}(y)\right) = \{0\}$ imply that $f(x, A) = 0$. For such $\mathcal{P}(y)$ that $\left|\mathcal{P}(y)\right| > \binom{(1-2^{-k})2^d}{n}$, it holds by the Pigeonhole Principle that $|A| \geq \left(1 - 2^{-k}\right) 2^d$. Then due to Lemma 5.3, that $f(x, A) = 0$ holds with probability at most $(\frac{r}{d})^{\alpha \log n}$. Putting the above arguments together, it holds for any partition $\mathcal{P}$ of Y with $|\mathcal{P}| \leq 2^b$ that

$$\begin{aligned}
\Pr_{x,y}\left[f\left(x, \mathcal{P}(y)\right) = \{0\}\right] &\leq \Pr_{y}\left[\left|\mathcal{P}(y)\right| \leq \binom{\left(1 - 2^{-k}\right) 2^d}{n}\right] \\
&\quad + \Pr_{x,y}\left[f\left(x, \mathcal{P}(y)\right) = \{0\} \;\middle|\; \left|\mathcal{P}(y)\right| > \binom{(1 - 2^{-k})2^d}{n}\right] \\
&\leq \exp\left(b \ln 2 - n^{(1-\alpha)}\right) \\
&\quad + \Pr_{x}\left[f\left(x, \bigcup \mathcal{P}(y)\right) = 0 \;\middle|\; \left|\bigcup \mathcal{P}(y)\right| > \left(1 - 2^{-k}\right) 2^d\right] \\
&\leq \exp\left(b \ln 2 - n^{(1-\alpha)}\right) + \left(\frac{r}{d}\right)^{\alpha \log n}.
\end{aligned}$$

Combining with (1), we have that

$$\Pr_{x,y}\left[\exists F \in \mathcal{F}(y),\, f(x, F) = \{0\}\right] \leq s \cdot \left(\exp\left(b \ln 2 - n^{(1-\alpha)}\right) + \left(\frac{1 + \log n}{d}\right)^{\alpha \log n}\right).$$

For such s and b that $s < (d/2 \log n)^{\alpha \log n}$ and $b < n^{1-\alpha}$, the above probability is $o(1)$. Therefore,

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

It follows that for any $s \times 2^b$ partition $\mathcal{F}$ of Y with the above setting of s and b, 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 Corollary 3.3, there is no $(s, b, 1)$-CPP for f with such range of s and b.

In Borodin et al. [1999], 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. Therefore, the same lower bound holds for λ-NN.

COROLLARY 5.4. *There does not exist a $(s, b, 1)$-CPP for the nearest neighbor search problem with n points in d-dimensional Hamming space where $d = \omega(\log n) \cap n^{o(1)}$ if $s < (d/2\log n)^{\alpha \log n}$ and $b < n^{1-\alpha}$ for some constant $0 < \alpha < 1$.*

Due to Lemma 2.1, the following lower bound on the nondeterministic cell-probe complexity holds.

COROLLARY 5.5. *There does not exist a (s, b, t)-CPP for the nearest neighbor search problem or the partial match problem with n points in d-dimensional Hamming space where $d = \omega(\log n) \cap n^{o(1)}$ if $s = Poly(n)$, $b < n^{1-\alpha}$ for some constant $\alpha > 0$ and $t = o\left(\log(d/\log n)\right)$.*

6. POLYNOMIAL EVALUATION

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

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 follows: for every query $(x, z) \in X$ and every data instance $g \in Y$, $f\left((x, z), g\right) = 1$ if $g(x) = z$ and $f\left((x, z), g\right) = 0$ otherwise. 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 naïve 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 $\left(2^k, k, 1\right)$-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 case (1), or the total storage size $s \cdot b$ is exactly as large as in case (2).

We first prove two lemmas. For any subset $P \subseteq Y$, let $\tau(P) = \left|\left\{x \in 2^k \mid \forall g_1, g_2 \in P, g_1(x) = g_2(x)\right\}\right|$, 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 $|P| \leq 1$, $\tau(P) = 2^k$.

LEMMA 6.1. *If $|P| > 1$, it holds that*

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

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 the 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 $2^{k(d-\tau)} \geq |P|$, that is, $\tau(P) \leq d - \frac{\log|P|}{k}$. □

LEMMA 6.2. *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 $|\mathcal{P}| \leq 2^b$ and $b \leq k(d-1)$, it holds that*

$$E\left\{\tau\big(\mathcal{P}(g)\big)\right\} \leq \frac{b}{k}.$$

PROOF. Let $P_1, P_2, \ldots, P_{2^b}$ denote the partition blocks and let $q_1, q_2, \ldots, q_{2^b}$ be the respective cardinalities. Naturally, we have that $\sum_{i=1}^{2^b} q_i = 2^{kd}$. We assume that $q_i = 0$ for $i = 1, 2, \ldots, m_0$, $q_i = 1$ for $i = m_0+1, m_0+2, \ldots, m$, and $q_i > 1$ for $i > m$. For those P_i that $i \leq m$, $|P_i| = q_i \leq 1$, thus $\tau(P_i) = 2^k$. According to Lemma 6.1,

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

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 $i = m+1, m+2, \ldots, 2^b$ are equal. Thus (2) is less than or equal to

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

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

$$\begin{aligned} &2^{b-k(d-1)}\epsilon \\ &\quad + \left(1-2^{b-kd}\epsilon\right)\left(d - \frac{\log 2^{kd}\left(1-2^{b-kd}\epsilon\right) - \log 2^b\left(1-2^{-b}\left(2^b\epsilon+m_0\right)\right)}{k}\right) \\ \leq\ & 2^{b-k(d-1)}\epsilon + \frac{1}{k}\left(1-2^{b-kd}\epsilon\right)\left(b + \log(1-\epsilon) - \log\left(1-2^{b-kd}\epsilon\right)\right) \\ \leq\ & \left(2^{b-k(d-1)} - \frac{b+1-2^{b-kd}}{k}\right)\epsilon - \frac{1}{k}\sum_{n=2}^{\infty}\left(\frac{1-2^{b-kd}}{n-1}\right)\epsilon^n. \end{aligned}$$

Note that $\epsilon \in [0,1)$. If $b \leq k(d-1)$, the above function of ϵ is monotonically decreasing over $[0,1)$, and thus its value is maximized when $\epsilon = 0$, that is, $E\{\tau(\mathcal{P}(g))\} \leq \frac{b}{k}$. □

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

THEOREM 6.3. *For any $(s, b, 1)$-CPP for the polynomial evaluation problem with parameters k and d where $d \le 2^k$, either $b > k(d-1)$ 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 $b \le k(d-1)$ 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 6.2,

$$\Pr_{x,g}\left[\forall g_1, g_2 \in \mathcal{P}(g), g_1(x) = g_2(x)\right] = \frac{1}{2^k} E\left\{\tau(\mathcal{P}(g))\right\} \le \frac{b}{k2^k}.$$

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

$$\begin{aligned}
& \Pr_{x,g}\left[\exists F \in \mathcal{F}(g), \forall g_1, g_2 \in F, g_1(x) = g_2(x)\right] \\
\le\ & s \cdot \Pr_{x,g}\left[\forall g_1, g_2 \in \mathcal{P}(g), g_1(x) = g_2(x)\right] \\
\le\ & \frac{s \cdot b}{k2^k} \\
<\ & 1,
\end{aligned}$$

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,

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

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 $h(x) \ne z$.

According to Theorem 3.3, we know that there does not exist $(s, b, 1)$-CPP with the given range of s and b. □

Due to Lemma 2.1, the following general lower bound holds.

COROLLARY 6.4. *If $t < \min\left(\frac{k}{b}(d-1), \frac{k-\log(d-1)}{\log s}\right)$, there does not exist (s, b, t)-CPP for the polynomial evaluation problem with parameters k and d.*

PROOF. If $t < \min\left(\frac{k}{b}(d-1), \frac{k-\log(d-1)}{\log s}\right)$, then it holds that $s^t < 2^k/(d-1)$ and $bt \le k(d-1)$, thus $s^t \cdot bt < k \cdot 2^k$ and $bt \le k(d-1)$. According to Theorem 6.3, there does not exists $(s^t, bt, 1)$-CPP for the polynomial evaluation problem with parameters k and d. And due to Lemma 2.1, there does not exist (s, b, t)-CPP for the polynomial evaluation problem with parameters k and d. □

Applying this corollary to the interesting case of $d = k^{\omega(1)} \cap o(2^k)$, that is, the degree of the polynomial is high, but not too high to trivialize the problem. For the feasible setting of a data structure that $s = \mathrm{Poly}(kd)$ and $b = \mathrm{Poly}(k)$, there exists a (s, b, t)-CPP for polynomial evaluation only if $t = \Omega\left(\frac{k}{\log kd}\right)$.

7. CONCLUSIONS

The results in this article contribute a general way of proving lower bounds for nondeterministic data structures: thus reducing the nondeterministic cell-probe complexity of data structure problems to the existence of the set systems with certain combinatorial structures, and proving the existence of this structure via probabilistic methods (or by any methods for proving the existence of combinatorial structures). Possible future work may be to apply this technique on other hard problems to discover new lower bounds.

There is also another interesting future direction, which is fundamental to the technique of proving lower bounds with cell-probe proofs. In the current work, the lower bounds for specific problems are proved by reducing general cell-probe proofs with parameter (s, b, t) to single-cell proofs $\left(s^t, bt, 1\right)$. Although it makes the analysis easier, the reduction actually amplifies the power of CPPs. Any analysis based on the parameter reduction in the above form cannot derive any lower bound better than $t > \frac{\log|X|}{\log s}$, which is the same barrier shared by the communication complexity model [Miltersen et al. 1998]. The existential lower bound proved in Section 4 shows that CPPs can support static lower bounds that beat communication complexity. In order to prove higher lower bound for CPPs than the communication model or to explore the intractability of nondeterministic data structures (e.g., to verify whether the conjecture of "the curse of dimensionality" holds nondeterministically), we have to directly analyze the CPPs with general parameters.

ACKNOWLEDGMENTS

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. I also want to thank Peter Bro Miltersen for suggesting to study the existential lower bounds.

REFERENCES

BARKOL, O. AND RABANI, Y. 2002. Tighter lower bounds for nearest neighbor search and related problems in the cell probe model. *J. Comput. Syst. Sci. 64,* 4, 873–896.

BEAME, P. AND FICH, F. 2002. Optimal bounds for the predecessor problem and related problems. *J. Comput. Syst. Sci. 65,* 1, 38–72.

BORODIN, A., OSTROVSKY, R., AND RABANI, Y. 1999. Lower bounds for high dimensional nearest neighbor search and related problems. In *Proceedings of the ACM Annual Symposium on Theory of Computing*. ACM, New York, 312–321.

BUHRMAN, H. AND DE WOLF, R. 2002. Complexity measures and decision tree complexity: a survey. *Theor. Comput. Sci. 288,* 1, 21–43.

FREDMAN, M. AND SAKS, M. 1989. The cell probe complexity of dynamic data structures. In *Proceedings of the ACM Annual Symposium on Theory of Computing*. ACM, New York, 345–354.

HUSFELDT, T. AND RAUHE, T. 1998. Hardness Results for dynamic problems by extensions of fredman and saks' chronogram method. In *Proceedings of the 25th International Colloquium on Automata, Languages and Programming*. Springer, Berlin, 67–78.

INDYK, P., GOODMAN, J., AND O'ROURKE, J. 2004. *Handbook of Discrete and Computational Geometry*. CRC Press, Ch. 39, 877–892.

JAYRAM, T., KHOT, S., KUMAR, R., AND RABANI, Y. 2004. Cell-probe lower bounds for the partial match problem. *J. Comput. Syst. Sci. 69,* 3, 435–447.

KUSHILEVITZ, E. AND NISAN, N. 1997. *Communication Complexity*. Cambridge University Press, Cambridge, UK.

MILTERSEN, P. 1995. On the cell probe complexity of polynomial evaluation. *Theor. Comput. Sci. 143,* 1, 167–174.

MILTERSEN, P. 1999. Cell probe complexity-a survey. In *Pre-Conference Workshop on Advances in Data Structures at the 19th Conference on Foundations of Software Technology and Theoretical Computer Science*.

MILTERSEN, P. 2008. Private communication.

MILTERSEN, P., NISAN, N., SAFRA, S., AND WIGDERSON, A. 1998. On data structures and asymmetric communication complexity. *J. Comput. Syst. Sci. 57,* 1, 37–49.

PĂTRAŞCU, M. 2008. Unifying the landscape of cell-probe lower bounds. In *Proceedings of the IEEE Conference on Foundations of Computer Science*. IEEE, Los Alamitos, CA, 434–443.

PĂTRAŞCU, M. AND DEMAINE, E. 2006. Logarithmic lower bounds in the cell-probe model. *SIAM J. Comput. 35,* 4, 932–963.

PĂTRAŞCU, M. AND THORUP, M. 2006. Time-space trade-offs for predecessor search. In *Proceedings of the ACM Annual Symposium on Theory of Computing*. ACM, New York, 232–240.

YAO, A. 1981. Should tables be sorted? *J. ACM 28,* 3, 615–628.

Received October 2008; revised February 2010; accepted February 2010