Local Verifications in Distributed Data Structures

Yitong Yin*

Abstract

We study distributed data structures, in which the results of queries to the data structure can be verified locally in a distributed network. We consider *locally checkable data structures*, which are the cell-probe schemes in which the results of queries can be checked locally. We introduce *cell-probe proofs (CPP)*, an interactive proof system for cell-probe model, and discuss the connection between CPP and locality in data structures. We present a combinatorial characterization of CPP. With these novel tools, we prove two lower bounds for the locally checkable data structures:

- There do not exist locally checkable data structures for the exact nearest neighbor search (NNS) problem or the partial match problem in high dimensional Hamming space, under the assumption that there are $\mathrm{Poly}(n)$ nodes, each of which contains $n^{o(1)}$ bits, and each node has at most $n^{o(1)}$ local nodes, where n is the number of points in the data set for NNS or partial match problem.
- For any locally checkable data structure for the polynomial evaluation problem, where the data structure stores a polynomial and each query is the input to the polynomial and the result to the query is the corresponding outcome, either the total storage of local nodes is close to the size of the whole polynomial, or the total storage size of the data structure is close to the naive upper bound that stores results for all possible queries.

*Department of Computer Science, Yale University. Supported by a Yale University Fellowship. Email: yitong.yin@yale.edu.

1 Introduction

A fundamental concern for distributed systems is *locality*. The theory of local computations started over a decade ago by [17, 20] laid a ground work for understanding the question of how locality may affect the computational power of a distributed system. Previous works in local computations [10, 16–18, 20] study the question of what kind of labeling of graphs can be computed based on local information only. Motivated by the exciting progress in the practice of distributed data structures in recent years (e.g. [2, 21, 22, 24]), we alternatively study the impact of locality on the power of *data structures*. This paper initiates the studies to the phenomenon of *local verifications* in data structures, and presents the first lower bounds for the data structures with the property of local verifiability.

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 from every pair of query $x \in X$ and data instance $y \in Y$ to an answer. The following are some examples.

> **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.
>
> **Partial match:** $X = \{0, 1, *\}^d$, $Y = \binom{\{0,1\}^d}{n}$, and $f(x, y) \in \{0, 1\}$ that for every $x \in X$ and $y \in Y$, $f(x, y) = 1$ if and only if there exists $z \in y$, 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 data structures is cell-probe model [23]. For each data instance y, a table of cells is constructed to store y. Upon a query x, an all-powerful algorithm tries to answer to a data structure problem $f(x, y)$, based on adaptive random access (probes) to the cells.

For distributed data structures, each cell corresponds to a node in the distributed system. For each data instance y, an underlying network connecting the nodes is constructed, so that each node has a set of local nodes in the current network, which we call it the **neighborhood** of the node. Intuitively, the neighborhood of a node contains the nodes that it may quickly access. For different systems, the definition of neighborhoods may vary: the neighborhood of a node u may contain the nodes adjacent to u, or the nodes within distance ℓ from u for some threshold ℓ. In any case, we parameterize the system by the maximum size of neighborhoods, denoted as δ, which characterizes the degree of locality.

We consider data structures that satisfy the following requirement of locality: for every data instance y, a structure of neighborhoods is imposed on cells as described above, such that for every query x to data y, the value of the answer $f(x, y)$ can be decided within *some* neighborhood. We call the data structure with such property as a **locally checkable data structure**. The performance of such data structures is measured by the number of cells, the number of bits in each cell, and the maximum size δ of the neighborhoods.

The above formulation of locality emerges from realistic distributed systems. A popular example for distributed data structure is "distributed object location" [3], the basic service provided by distributed hash tables (DHTs), where a set of objects is distributed among nodes, and upon a query of an object, the node that contains the queried object is located. In order to deal with the

churn of nodes, the above functionality is always reduced to the nearest neighbor search problem by mapping all nodes and objects into a metric space, and allocating each object to its closest nodes. Most existing systems (e.g. [2, 21, 22, 24]) assume a one-dimensional metric, and there are also distributed data structures that assume metrics with bounded growth rate[1] ([12, 15]).

In the above examples, in comparison with the cell-probe mode, there exist several levels of locality: first, for each instance of data, a communication network does exist, such that each node is adjacent to bounded number of other nodes; second, probes to the cells have to be adjacent to some of the previous probes in the underlying network; and third, the most important, every query result can be verified locally by some node. In this paper, we only consider the locality of the third kind, because (1) it shows a more data structural favor than graph theoretical favor, and (2) as we will show, it is fundamental to the power of the data structures. The locally checkable data structures are the data structures satisfying this kind of locality requirement.

It is easy to understand the concept of local verifications in data structures by thinking about the query algorithm in a distributed data structure as walking in a network of nodes, looking for the certificate for the query, namely, the information used by the algorithm to decide the query result. The locally checkable data structure requires that for every query there is a certificate locally accessible from some node.

Such a notion of locality is essential for the distributed systems that the status of the system constantly changes, while the query results has to be consistent with the current status of the system. This is the situation in almost all large-scale distributed systems. In such systems, in order to guarantee that the computation of the query result is not disrupted by the status updates, a certificate of the query should be quickly accessible from some node, i.e. should be within the neighborhood of that node.

The locality of the certificates has been widely assumed in the existing upper bounds for distributed data structures. In some cases the assumption is implicitly dealt with, such as in [2, 21, 22, 24], where the problem is NNS in a ring, therefore for each query, its predecessor and successor in the ring serve as a local certificate. For the case of NNS in a growth-restricted metric, in the distributed implementation of [15] by Chord [22], the bounded size finger lists serve as local certificates as argued in [15]. The term of certificate was explicitly mentioned in [12], and in their distributed implementation, Tapestry [24], the locality of certificates is also guaranteed.

For some hard problems, such as NNS and partial match in a high dimensional Hamming cube [13,14], or polynomial evaluation [19], it is still widely open whether there exist data structures that have local certificates, even if we allow the running time of the query algorithm to be unbounded. For high dimensional NNS, this problem is especially interesting because a recent result [1] shows a tradeoff between load balance and the dimension of the metric space when allocating objects to closest nodes and high dimensional Hamming cube indeed has optimal load balance. However, existing lower bounds for these problems [4, 6, 7, 14, 19] do not apply to our model because none of them addresses the requirement of locality.

Motivated by the exciting progress in the upper bounds of distributed data structures, which all use local verifications effectively to meet realistic concerns, we are also curious about lower bound side of the story. In particular, we want to explore the power of locally checkable data structures, we want to characterize the problems that can be solved efficiently by such data structures, and finally, we want to know for the open problems such as NNS in high dimension, whether there exists an efficient locally checkable data structure.

[1]Formally, the metric space has bounded KR-dimension, a counting measure of doubling dimension.

1.1 Our results

We introduce a notion of locality into the cell-probe model by introducing a concept of locally checkable data structures, which addresses the locality of verifications in distributed data structures.

We introduce cell-probe proofs, an interactive proof system in the cell-probe model. This notion of proofs formulates verifications instead of computations in the cell-probe model. We discuss the connection between cell-probe proofs and locally checkable data structures. Unlike the fully adaptive computations in the cell-probe model, the formulation of cell-probe proofs shows a combinatorial simplicity. We introduce a combinatorial structure that fully characterizes the problems that have cell-probe proofs with certain parameters.

With these novel tools, we prove two lower bounds on the data structures due to the requirement of locality:

- There do not exist locally checkable data structures for exact nearest neighbor search (NNS) problem or partial match problem in high dimensional Hamming space, under the assumption that there are $\mathrm{Poly}(n)$ nodes, each of which contains $n^{o(1)}$ bits, and each node has a $n^{o(1)}$-size neighborhood, where n is the number of points in the data set for NNS or partial match problem.
- There do not exist locally checkable data structures for polynomial evaluation problem defined on $(d-1)$-degree polynomials over finite filed 2^k, under the assumption that there are s nodes, each of which contains b bits, and each node has a δ-size neighborhood, such that $\delta(b+\log s) < k(d-1) - \log k$ and $s\delta(b+\log s) < k2^k$.

2 Model

A **static data structure problem** or just **data structure problem**, is represented as a boolean function $f : X \times Y \to \{0,1\}$. In order to prove lower bounds, we only consider the decision problems. We refer to each $y \in Y$ as an instance of **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. [9,11,23]), 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. Upon each 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 a distributed data structure, where each cell corresponds to a node in the distributed system, for each data instance y, in addition to the table of cells, an underlying network $G_y(I, E_y)$ is also constructed, which is an undirected graph whose vertex set is the set of cells. We suppose that from any node, the nodes less than ℓ distant from it are its local nodes, so that for each network G_y, a structure of neighborhoods is given as $\Delta_y : I \to 2^I$, that for every node $i \in I$, the set of all local nodes to i is $\Delta_y(i) = \{j \in I \mid d_y(i,j) < \ell\}$ where $d_y(i,j)$ denotes the distance between i and j in G_y. We use the maximum size of a neighborhood $\delta = \max_{y,i} |\Delta_y(i)|$ to parameterize the degree of locality. Intuitively, $\Delta_y(i)$ is the set of nodes that are quickly accessible from node i in

the network G_y, and δ is the upper bound on the number of such nodes. In most realistic systems, it holds that $\delta = s^{o(1)}$ for a system with s nodes.

A **locally checkable data structure** is a cell-probe scheme such that for every instance of data y, there exists a structure of neighborhoods $\{\Delta_y(i)\}_i$ for cells, such that for each query x, the value of $f(x, y)$ is checkable within some neighborhood $\Delta_y(i)$. It is formally defined as follow.

Definition 1 *For a cell-probe scheme, we denote its table structure as $T : Y \times I \to \{0,1\}^b$, and let S_{xy} denote the set of cells probed by the query algorithm upon query x to data y.*

*We say the cell-probe scheme is δ-**local**, if there exists a neighborhood structure $\Delta : Y \times I \to 2^I$, such that $\max_{y,i} |\Delta_y(i)| = \delta$ and for every x and y, there is some subset $P \subseteq S_{xy}$ of probed cells, such that $P \subseteq \Delta_y(i)$ for some $i \in I$, and for all such $y' \in Y$ that $\forall j \in P, T_{y'}(j) = T_y(j)$, it holds that $f(x, y') = f(x, y)$.*

Intuitively, the cells in P contains the necessary information that decides the value of $f(x, y)$, and a δ-local cell-probe scheme requires that such information is always local to some cell.

In order to formally characterize the local verifiability of data structures, we introduce a new concept, cell-probe proofs, which formalizes the proofs and verifications in the cell-probe model. With the help of cell-probe proofs, we have a much simpler definition of locally checkable data structures.

2.1 Cell-probe proofs

For a specific data structure problem f, a cell-probe proof system (CPP) may be subsequently defined for f.

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, who 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:

$$\begin{aligned} &\text{(Completeness)} \quad \exists P \subseteq I: \quad v(x, \{\langle i, T_y(i)\rangle \mid i \in P\}) = f(x, y), \text{ and} \\ &\text{(Soundness)} \qquad \forall P \subseteq I: \quad v(x, \{\langle i, T_y(i)\rangle \mid i \in P\}) = \begin{cases} f(x, y) \\ \perp \end{cases}. \end{aligned}$$

A (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 [23], 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 [5].

Note that although a data structure problem is nothing but a boolean function, CPP is very different from the certificate complexity of boolean functions [8]. In CPP, the prover and the verifier communicate with each other via a table structure, which distinguishes CPP with the 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 quite impossible in the model of certificate complexity.

We can alternatively interpret cell-probe proofs as nondeterministic cell probes. Naturally, for a cell-probe scheme, for any query, the cells probed by the algorithm contains a cell-probe proof, which provides the necessary information to answer the query. For a cell-probe scheme, let S_{xy} denote the set of cells probed by the query algorithm upon query x to data y. We say a cell-probe scheme contain a CPP, if they have the same table structure and for every x and y, the proof $P_{xy} \subseteq S_{xy}$.

We can therefore have an alternative definition for the locally checkable data structures.

Definition 2 *A cell-probe scheme is δ-local if it contains a CPP and there exists a neighborhood structure $\Delta : Y \times I \to 2^I$, such that $\max_{y,i} |\Delta_y(i)| = \delta$ and for every x and y, the proof given by the CPP $P_{xy} \subseteq \Delta_y(i)$ for some i.*

The equivalence of Definition 1 and Definition 2 directly follows the definition of CPP.

According to Definition 2, a δ-local cell-probe scheme with s cells each of b bits implies a (s, b, δ)-CPP. The following lemma further reduce it to 1-cell proofs, which are essential for our analysis.

Lemma 3 *For any data structure problem f, if there exists a δ-local cell-probe scheme with s cells each of b bits, then there exists a $(s, \delta(b + \log s), 1)$-CPP for f.*

Proof: Assuming that for the δ-local cell-probe scheme, the structure of neighborhood for each y is $\{\Delta_y(i)\}_i$, and the table structure is $T : Y \times [s] \to \{0, 1\}^b$, we define a new table structure $T' : Y \times [s] \to \{0, 1\}^{\delta(b+\log s)}$ by letting $T'_y(i) = \{\langle j, T_y(j)\rangle \mid j \in \Delta_y(i)\}$, i.e. we store all the cells in each $\Delta_y(i)$ along with their indices as a new cell. Because the new table T'_y store both the contents and indices of cells of the old table T_y, probes to T_y can be simulated by probes to T'_y. According to the definition of δ-local cell-probe schemes, for every x and y, a cell-probe proof is contained by some $\Delta_y(i)$ in the old table T_y, therefore in the new table T'_y, for every x and y, a cell-probe proof is contained in a cell, which implies a $(s, \delta(b + \log s), 1)$-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 4 *We say a set system $\mathcal{F} \subseteq 2^Y$ is a $s \times k$-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 existence of cell-probe proofs. We start with the case for 1-cell proofs.

Theorem 5 *There is a $(s, b, 1)$-CPP for $f : X \times Y \to \{0,1\}$, if and only if there exists a $s \times 2^b$-partition $\mathcal{F}$ of Y, such that for every $x \in X$ and every $y \in Y$, there is a $F \in \mathcal{F}(y)$ 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 A such that $A_{ij} = \{y \in Y \mid T_y(i) = j\}$, i.e. A_{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 A 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 A is a $s \times 2^b$-partition $\mathcal{F}$ of Y, where each A_{ij} is a $F \in \mathcal{F}$.

If there is a $(s, b, 1)$-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 A, A_{ij} contains all y' such that $T_{y'}(i) = j$, thus $|f(x, A_{ij})| = 1$.

($\Longleftarrow$) Assuming that $\mathcal{F}$ is a $s \times 2^b$-partition of Y such that for every x and every y there is a $F \in \mathcal{F}(y)$ that $|f(x, F)| = 1$, we rewrite $\mathcal{F}$ in the form of a $s \times 2^b$ matrix A that A_{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 $y \in A_{ij}$. Because each row of A is a partition of Y, such T is well-defined.

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

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 a $s \times 2^b$-partition 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 5, 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 $\mathcal{F}$ reaches 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 $\{0,1\}^d$, that $X = \{0,1\}^d$, $Y = \binom{\{0,1\}^d}{n}$ and $f(x, y) \in \{0,1\}$ answers whether there exists a point in y within distance λ from the x. As in [4, 7], 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 exists $(\text{Poly}(n), n^{o(1)}, 1)$-CPP for λ-NN problem, which according to Lemma 3, implies that there does not exist practical locally checkable data structures for NNS in high dimensional space. To show this, we show the same lower bound for the partial match problem [13, 14], which is an instantiation of the λ-NN problem as shown in [7].

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 $X = \{0,1,*\}^d$. 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 6 *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 5, 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 probabilistic method. With some distribution of x and y, we show that for all $s \times 2^b$ 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 7 *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,

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

∎

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 8 *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 by 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 "1"s. Therefore,

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

We then prove that for all $s \times 2^b$ partition $\mathcal{F}$ 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,

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

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

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

for some partition $\mathcal{P}$ of Y where $|\mathcal{P}| \leq 2^b$. 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 $|\mathcal{P}(y)| \leq \binom{(1-2^{-k})2^d}{n}$ and the case that $|\mathcal{P}(y)| > \binom{(1-2^{-k})2^d}{n}$. According to Lemma 7, for any partition $\mathcal{P}$ of Y with $|\mathcal{P}| \leq 2^b$, the probability that $|\mathcal{P}(y)| \leq \binom{(1-2^{-k})2^d}{n} = \binom{2^d}{n}/2^{n^{\Omega(1)}}$ is at most $n^{-\omega(1)}$.

We let $A = \bigcup \mathcal{P}(y) = \bigcup_{y' \in \mathcal{P}(y)} y'$. Note that $A \subseteq \{0,1\}^d$, and $f(x,\mathcal{P}(y)) = \{0\}$ implies that $f(x,A) = 0$. For such $\mathcal{P}(y)$ that $|\mathcal{P}(y)| > \binom{(1-2^{-k})2^d}{n}$, by pigeonhole principle, it holds that $|A| \geq (1-2^{-k})2^d$. Due to Lemma 8, $f(x,A) = 0$ with prohibitively small probability. Putting them together, it holds for arbitrary partition $\mathcal{P}$ of Y with $|\mathcal{P}| \leq 2^b$ that

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

Combining with (1), we have that

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

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

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

It follows that for any $s \times 2^b$ partition $\mathcal{F}$ of Y, where $s = \text{Poly}(n)$ 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 5, there is no $(s,b,1)$-CPP for f with the above range of s and b. ∎

In [7], 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 9 *There does not exist $(Poly(n), n^{o(1)}, 1)$-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 3, the following lower bound holds for locally checkable data structures.

Corollary 10 *There does not exist an $n^{o(1)}$-local cell-probe scheme with $\mathrm{Poly}(n)$ cells each of $n^{o(1)}$ bits, for the nearest neighbor search or 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 $\leq (d-1)$ 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: for every query $(x, z) \in X$ and every data instance $g \in Y$, $f((x, z), g) = 1$ if $g(x) = z$ and $f((x, z), g) = 0$ if otherwise, i.e. a polynomial g is preprocessed and stored as a data structure, so that on 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 reveal 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. 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 $|P| \leq 1$, $\tau(P) = 2^k$.

Lemma 11 *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 $x_\tau, x_{\tau+1}, \ldots, x_d$. 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 $i = \tau, \tau+1, \ldots, d$, since if otherwise, $g_1 \equiv g_2$ by interpolation. Recall that g is a polynomial of 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 $x_\tau, x_{\tau+1}, \ldots, x_d$, it must hold that $2^{k(d-\tau)} \geq |P|$, i.e. $\tau(P) \leq d - \frac{\log |P|}{k}$. ∎

Lemma 12 *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 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 that $|\mathcal{P}| \leq 2^b$ and $b < k(d-1) - \log k$, it holds that*

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

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 11,

$$E\{\tau(\mathcal{P}(g))\} = \sum_{i=1}^{2^b} \frac{q_i}{2^{kd}} \tau(P_i) \leq (m - m_0) \cdot \frac{2^k}{2^{kd}} + \sum_{i=1}^{2^b - m} \frac{q_i}{2^{kd}} (d - \frac{\log q_i}{k}),$$

which due to Lagrange multiplier, is $\leq \frac{m-m_0}{2^{k(d-1)}} + \frac{2^{kd}-m}{2^{kd}}(d - \frac{\log(2^{kd}-m) - \log(2^b - m)}{k})$, which is maximized when $m = m_0 = 0$ if $b < k(d-1) - \log k$, i.e. $E\{\tau(\mathcal{P}(g))\} \leq \frac{b}{k}$. ∎

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

Theorem 13 *For any $(s, b, 1)$-CPP for the polynomial evaluation problem with parameters k and d where $d \leq 2^k$, either $b \geq k(d-1) - \log k$ or $s \cdot b \geq k \cdot 2^k$.*

Proof: We will prove that there does not exist $(s, b, 1)$-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 $|\mathcal{P}| \leq 2^b$, according to Lemma 12,

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

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

$$\Pr_{x,g}[\exists F \in \mathcal{F}(g) \text{ s.t. } \forall g_1, g_2 \in F, g_1(x) = g_2(x)] \leq s \cdot \Pr_{x,g}[\forall g_1, g_2 \in \mathcal{P}(g), g_1(x) = g_2(x)] \leq \frac{s \cdot b}{k2^k} < 1,$$

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 $g \in Y$ 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 5, we know that there does not exist $(s, b, 1)$-CPP with the given range of s and b. ∎

Due to Lemma 3, the following lower bound holds for locally checkable data structures.

Corollary 14 *There does not exist a δ-local cell-probe scheme with s cells, each of which contains b bits, for the polynomial evaluation problem with parameter k and d where $d \leq 2^k$, if $(b + \log s) < k(d-1) - \log k$ and $s\delta(b + \log s) < k2^k$.*

6 Acknowledgment

I would like to thank James Aspnes for the helpful discussion, and thank Dana Angluin for her comments on the early version of the paper.

References

[1] J. Aspnes, M. Safra, and Y. Yin. Ranged hash functions and the price of churn. To appear in SODA 2008.

[2] J. Aspnes and G. Shah. Skip graphs. *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms (SODA 2003)*, pages 384–393, 2003.

[3] H. Balakrishnan, M. Kaashoek, D. Karger, R. Morris, and I. Stoica. Looking up data in P2P systems. *Communications of the ACM*, 46(2):43–48, 2003.

[4] O. Barkol and Y. Rabani. Tighter bounds for nearest neighbor search and related problems in the cell probe model. *Proceedings of the thirty-second annual ACM Symposium on Theory of Computing (STOC 1999)*, pages 388–396, 1999.

[5] P. Beame and F. Fich. Optimal bounds for the predecessor problem. *Proceedings of the thirty-first annual ACM Symposium on Theory of Computing (STOC 1999)*, pages 295–304, 1999.

[6] P. Beame and E. Vee. Time-space tradeoffs, multiparty communication complexity, and nearest-neighbor problems. *Proceedings of the 34th annual ACM Symposium on Theory of Computing (STOC 2002)*, pages 688–697, 2002.

[7] A. Borodin, R. Ostrovsky, and Y. Rabani. Lower bounds for high dimensional nearest neighbor search and related problems. *Proceedings of the thirty-first annual ACM Symposium on Theory of Computing (STOC 1999)*, pages 312–321, 1999.

[8] H. Buhrman and R. de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288(1):21–43, 2002.

[9] E. Demaine and M. Pătraşcu. Logarithmic lower bounds in the cell-probe model. *SIAM Journal of Computing*, 35(4):932–963, 2006.

[10] M. Elkin. Unconditional lower bounds on the time-approximation tradeoffs for the distributed minimum spanning tree problem. *Proceedings of the thirty-sixth annual ACM Symposium on Theory of Computing (STOC 2004)*, pages 331–340, 2004.

[11] M. Fredman and M. Saks. The cell probe complexity of dynamic data structures. *Proceedings of the twenty-first annual ACM Symposium on Theory of Computing (STOC 1989)*, pages 345–354, 1989.

[12] K. Hildrum, J. Kubiatowicz, S. Ma, and S. Rao. A note on the nearest neighbor in growth-restricted metrics. *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 560–561, 2004.

[13] P. Indyk, J. Goodman, and J. O'Rourke. Nearest neighbors in high-dimensional spaces. *Handbook of Discrete and Computational Geometry, chapter 39*, 2004.

[14] T. Jayram, S. Khot, R. Kumar, and Y. Rabani. Cell-probe lower bounds for the partial match problem. *Journal of Computer and System Sciences*, 69(3):435–447, 2004.

[15] D. Karger and M. Ruhl. Finding nearest neighbors in growth-restricted metrics. *Proceedings of the thiry-fourth annual ACM Symposium on Theory of Computing (STOC 2002)*, pages 741–750, 2002.

[16] F. Kuhn, T. Moscibroda, and R. Wattenhofer. What cannot be computed locally! *Proceedings of the twenty-third annual ACM Symposium on Principles of Distributed Computing (PODC 2004)*, pages 300–309, 2004.

[17] N. Linial. Locality in Distributed Graph Algorithms. *SIAM Journal on Computing*, 21:193, 1992.

[18] A. Mayer, M. Naor, and L. Stockmeyer. *Local Computations on Static and Dynamic Graphs*. Weizmann Institute of Science, Dept. of Applied Mathematics and Computer Science, 1995.

[19] P. Miltersen. On the cell probe complexity of polynomial evaluation. *Theoretical Computer Science*, 143(1):167–174, 1995.

[20] M. Naor and L. Stockmeyer. What Can be Computed Locally? *SIAM Journal on Computing*, 24(6):1259–1277, 1995.

[21] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, 11:329–350, 2001.

[22] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *Proceedings of the 2001 SIGCOMM conference*, 31(4):149–160, 2001.

[23] A. Yao. Should Tables Be Sorted? *Journal of the ACM*, 28(3):615–628, 1981.

[24] B. Zhao, L. Huang, J. Stribling, S. Rhea, A. Joseph, and J. Kubiatowicz. Tapestry: a resilient global-scale overlay for service deployment. *Selected Areas in Communications, IEEE Journal on*, 22(1):41–53, 2004.