Efficient and Error-Correcting Data Structures for Membership and Polynomial Evaluation

Victor Chen, Elena Grigorescu, Ronald de Wolf

Symposium on Theoretical Aspects of Computer Science 2010 (Nancy, France), pp. 203-214
www.stacs-conf.org

EFFICIENT AND ERROR-CORRECTING DATA STRUCTURES FOR MEMBERSHIP AND POLYNOMIAL EVALUATION

VICTOR CHEN [1] AND ELENA GRIGORESCU [2] AND RONALD DE WOLF [3]

[1] Tsinghua University ITCS and MIT CSAIL
E-mail address: victor.vc@gmail.com

[2] MIT CSAIL
E-mail address: elena_g@mit.edu

[3] CWI. Science Park 123. 1098XG Amsterdam. The Netherlands
E-mail address: rdewolf@cwi.nl

ABSTRACT. We construct efficient data structures that are resilient against a constant fraction of adversarial noise. Our model requires that the decoder answers *most* queries correctly with high probability and for the remaining queries, the decoder with high probability either answers correctly or declares "don't know." Furthermore, if there is no noise on the data structure, it answers *all* queries correctly with high probability. Our model is the common generalization of an error-correcting data structure model proposed recently by de Wolf, and the notion of "relaxed locally decodable codes" developed in the PCP literature.

We measure the efficiency of a data structure in terms of its *length* (the number of bits in its representation), and query-answering time, measured by the number of *bit-probes* to the (possibly corrupted) representation. We obtain results for the following two data structure problems:

- (Membership) Store a subset S of size at most s from a universe of size n such that membership queries can be answered efficiently, i.e., decide if a given element from the universe is in S.
  We construct an error-correcting data structure for this problem with length nearly linear in $s \log n$ that answers membership queries with $O(1)$ bit-probes. This nearly matches the asymptotically optimal parameters for the noiseless case: length $O(s \log n)$ and one bit-probe, due to Buhrman, Miltersen, Radhakrishnan, and Venkatesh.
- (Univariate polynomial evaluation) Store a univariate polynomial g of degree $\deg(g) \leq s$ over the integers modulo n such that evaluation queries can be answered efficiently, i.e., we can evaluate the output of g on a given integer modulo n.
  We construct an error-correcting data structure for this problem with length nearly linear in $s \log n$ that answers evaluation queries with $\mathrm{polylog}\, s \cdot \log^{1+o(1)} n$ bit-probes. This nearly matches the parameters of the best-known noiseless construction, due to Kedlaya and Umans.

1998 ACM Subject Classification: E1, E4.

Key words and phrases: Data Structures, Error-Correcting Codes, Membership, Polynomial Evaluation.

This work was done when the author was a student at MIT. Supported by NSF award CCF-0829672 and National Natural Science Foundation of China Grant 60553001, the National Basic Research Program of China Grant 2007CB807900, 2007CB807901.

This work started when this author was visiting CWI in Summer 2008. Supported by NSF award CCF-0829672.

Supported by a Vidi grant from the Netherlands Organization for Scientific Research (NWO).

1. Introduction

The area of data structures is one of the oldest and most fundamental parts of computer science, in theory as well as in practice. The underlying question is a time-space tradeoff: we are given a piece of data, and we would like to store it in a short, space-efficient data structure that allows us to quickly answer specific queries about the stored data. On one extreme, we can store the data as just a list of the correct answers to all possible queries. This is extremely time-efficient (one can immediately look up the correct answer without doing any computation) but usually takes significantly more space than the information-theoretic minimum. At the other extreme, we can store a maximally compressed version of the data. This method is extremely space-efficient but not very time-efficient since one usually has to undo the whole compression first. A good data structure sits somewhere in the middle: it does not use much more space than the information-theoretic minimum, but it also stores the data in a structured way that enables efficient query-answering.

It is reasonable to assume that most practical implementations of data storage are susceptible to *noise*: over time some of the information in the data structure may be corrupted or erased by various accidental or malicious causes. This buildup of errors may cause the data structure to deteriorate so that most queries are not answered correctly anymore. Accordingly, it is a natural task to design data structures that are not only efficient in space and time but also resilient against a certain amount of *adversarial* noise, where the noise can be placed in positions that make decoding as difficult as possible.

Ways to protect information and computation against noise have been well studied in the theory of error-correcting codes and of fault-tolerant computation. In the data structure literature, constructions under often incomparable models have been designed to cope with noise. We mention a few of these models here. First, Aumann and Bender [1] studied pointer-based data structures such as linked lists, stacks, and binary search trees. In this model, errors (adversarial but detectable) occur whenever all the pointers from a node are lost. They studied the dependence between the number of errors and the number of nodes that become irretrievable, and designed a number of efficient data structures where this dependence is reasonable.

Another model for studying data structures with noise is the faulty-memory RAM model, introduced by Finocchi and Italiano [10]. In a faulty-memory RAM, there are $O(1)$ memory cells that cannot be corrupted by noise. Elsewhere, errors (adversarial and undetectable) may occur at any time, even during the decoding procedure. Many data structure problems have been examined in this model, such as sorting [8], searching [9], priority queues [13] and dictionaries [4]. However, the number of errors that can be tolerated is typically less than a linear portion of the size of the input. Furthermore, correctness can only be guaranteed for keys that are not affected by noise. For instance, for the problem of comparison-sorting on n keys, the authors of [8] designed a resilient sorting algorithm that tolerates $\sqrt{n \log n}$ keys being corrupted and ensures that the set of uncorrupted keys remains sorted.

Recently, de Wolf [19] considered another model of resilient data structures. The representation of the data structure is viewed as a bit-string, from which a decoding procedure can read any particular set of bits to answer a data query. The representation must be able to tolerate a constant fraction δ of adversarial noise in the bit-string[1] (but not inside the decoding procedure). His model generalizes the usual noise-free data structures (where $\delta = 0$) as well as the so-called "locally decodable codes" (LDCs) [14]. Informally, an LDC is an encoding that is tolerant of noise and allows

[1]We only consider bit-flip-errors here, not erasures. Since erasures are easier to deal with than bit-flips, it suffices to design a data structure dealing with bit-flip-errors.

fast decoding so that each message symbol can be retrieved correctly with high probability. Using LDCs as building blocks, de Wolf constructed data structures for several problems.

Unfortunately, de Wolf's model has the drawback that the optimal time-space tradeoffs are much worse than in the noise-free model. The reason is that all known constructions of LDCs that make $O(1)$ bit-probes [21, 7] have very poor encoding length (super-polynomial in the message length). In fact, this encoding length provably must be super-linear in the message length [14, 16, 20]. As his model is a generalization of LDCs, data structures cannot have a succinct representation that has length proportional to the information-theoretic bound.

We thus ask: what is a clean model of data structures that allows efficient representations *and* has error-correcting capabilities? Compared with the pointer-based model and the faulty-memory RAM, de Wolf's model imposes a rather stringent requirement on decoding: *every* query must be answered correctly with high probability from the possibly corrupted encoding. While this requirement is crucial in the definition of LDCs due to their connection to complexity theory and cryptography, for data structures it seems somewhat restrictive.

In this paper we consider a broader, more relaxed notion of error-correction for data structures. In our model, for most queries, the decoder has to return the correct answer with high probability. However, for the few remaining queries, the decoder may claim ignorance, i.e., declare the data item unrecoverable from the (corrupted) data structure. Still, for *every* query, the answer is incorrect only with small probability. In fact, just as de Wolf's model is a generalization of LDCs, our model in this paper is a generalization of the "relaxed" locally decodable codes (RLDCs) introduced by Ben-Sasson, Goldreich, Harsha, Sudan, and Vadhan [3]. They relax the usual definition of an LDC by requiring the decoder to return the correct answer on *most* rather than all queries. For the remaining queries it is allowed to claim ignorance, i.e., to output a special symbol '$\perp$' interpreted as "don't know" or "unrecoverable." As shown in [3], relaxing the LDC-definition like this allows for constructions of RLDCs with $O(1)$ bit-probes of *nearly linear* length.

Using RLDCs as building blocks, we construct error-correcting data structures that are very efficient in terms of time as well as space. Before we describe our results, let us define our model formally. First, a *data structure problem* is specified by a set D of *data items*, a set Q of *queries*, a set A of *answers*, and a function $f : D \times Q \to A$ which specifies the correct answer $f(x, q)$ of query q to data item x. A data structure for f is specified by four parameters: t the number bit-probes, δ the fraction of noise, ε an upper bound on the error probability for each query, and λ an upper bound on the fraction of queries in Q that are not answered correctly with high probability (the 'λ' stands for "lost").

Definition 1.1. Let $f : D \times Q \to A$ be a data structure problem. Let $t > 0$ be an integer, $\delta \in [0, 1]$, $\varepsilon \in [0, 1/2]$, and $\lambda \in [0, 1]$. We say that f has a $(t, \delta, \varepsilon, \lambda)$-*data structure* of length N if there exist an encoder $\mathcal{E} : D \to \{0,1\}^N$ and a (randomized) decoder $\mathcal{D}$ with the following properties: for every $x \in D$ and every $w \in \{0,1\}^N$ at Hamming distance $\Delta(w, \mathcal{E}(x)) \leq \delta N$,

(1) $\mathcal{D}$ makes at most t bit-probes to w,
(2) $\Pr[\mathcal{D}^w(q) \in \{f(x, q), \perp\}] \geq 1 - \varepsilon$ for every $q \in Q$,
(3) the set $G = \{q : \Pr[\mathcal{D}^w(q) = f(x, q)] \geq 1 - \varepsilon\}$ has size at least $(1 - \lambda)|Q|$ ('G' stands for "good"),
(4) if $w = \mathcal{E}(x)$, then $G = Q$.

Here $\mathcal{D}^w(q)$ denotes the random variable which is the decoder's output on inputs w and q. The notation indicates that it accesses the two inputs in different ways: while it has full access to the query q, it only has bit-probe access (or "oracle access") to the string w.

We say that a $(t, \delta, \varepsilon, \lambda)$-data structure is *error-correcting*, or an *error-correcting data structure*, if $\delta > 0$. Setting $\lambda = 0$ recovers the original notion of error-correction in de Wolf's model [19]. A $(t, \delta, \varepsilon, \lambda)$-*relaxed locally decodable code (RLDC)*, defined in [3], is an error-correcting data structure for the membership function $f : \{0,1\}^n \times [n] \to \{0,1\}$, where $f(x, i) = x_i$. A (t, δ, ε)-*locally decodable code (LDC)*, defined by Katz and Trevisan [14], is an RLDC with $\lambda = 0$.

Remark 1.2. For the data structure problems considered in this paper, our decoding procedures make only *non-adaptive* probes, i.e., the positions of the probes are determined all at once and sent simultaneously to the oracle. For other data structure problems it may be natural for decoding procedures to be adaptive. Thus, we do not require $\mathcal{D}$ to be non-adaptive in Condition 1 of Definition 1.1.

1.1. Our results

We obtain efficient error-correcting data structures for the following two data structure problems.

MEMBERSHIP: Consider a universe $[n] = \{1, \ldots, n\}$ and some nonnegative integer $s \leq n$. Given a set $S \subseteq [n]$ with at most s elements, one would like to store S in a compact representation that can answer "membership queries" efficiently, i.e., given an index $i \in [n]$, determine whether or not $i \in S$. Formally $D = \{S : S \subseteq [n], |S| \leq s\}$, $Q = [n]$, and $A = \{0,1\}$. The function $\text{MEM}_{n,s}(S, i)$ is 1 if $i \in S$ and 0 otherwise.

Since there are at least $\binom{n}{s}$ subsets of the universe of size at most s, each subset requiring a different instantiation of the data structure, the information-theoretic lower bound on the space of any data structure is at least $\log \binom{n}{s} \approx s \log n$ bits.[2] An easy way to achieve this is to store S in sorted order. If each number is stored in its own $\log n$-bit "cell," this data structure takes s cells, which is $s \log n$ bits. To answer a membership query, one can do a binary search on the list to determine whether $i \in S$ using about $\log s$ "cell-probes," or $\log s \cdot \log n$ bit-probes. The length of this data structure is essentially optimal, but its number of probes is not. Fredman, Komlós, and Szemerédi [11] developed a famous hashing-based data structure that has length $O(s)$ cells (which is $O(s \log n)$ bits) and only needs a *constant* number of cell-probes (which is $O(\log n)$ bit-probes). Buhrman, Miltersen, Radhakrishnan, and Venkatesh [5] improved upon this by designing a data structure of length $O(s \log n)$ bits that answers queries with *only one bit-probe* and a small error probability. This is simultaneously optimal in terms of time (clearly one bit-probe cannot be improved upon) and space (up to a constant factor).

None of the aforementioned data structures can tolerate a constant fraction of noise. To protect against noise for this problem, de Wolf [19] constructed an error-correcting data structure with $\lambda = 0$ using a locally decodable code (LDC). That construction answers membership queries in t bit-probes and has length roughly $L(s, t) \log n$, where $L(s, t)$ is the shortest length of an LDC encoding s bits with bit-probe complexity t. Currently, all known LDCs with $t = O(1)$ have $L(s, t)$ super-polynomial in s [2, 21, 7]. In fact, $L(s, t)$ must be super-linear for all constant t, see e.g. [14, 16, 20].

Under our present model of error-correction, we can construct much more efficient data structures with error-correcting capability. First, it is not hard to show that by composing the BMRV data structure [5] with the error-correcting data structure for $\text{MEM}_{n,n}$ (equivalently, an RLDC) [3], one

[2] Our logs are always to base 2.

can already obtain an error-correcting data structure of length $O((s \log n)^{1+\eta})$, where η is an arbitrarily small constant. However, following an approach taken in [19], we obtain a data structure of length $O(s^{1+\eta} \log n)$, which is much shorter than the aforementioned construction if $s = o(\log n)$.

Theorem 1.3. *For every $\varepsilon, \eta \in (0,1)$, there exist an integer $t > 0$ and real $\tau > 0$, such that for all s and n, and every $\delta \le \tau$, $\text{MEM}_{n,s}$ has a $(t, \delta, \varepsilon, \frac{s}{2n})$-data structure of length $O(s^{1+\eta} \log n)$.*

We will prove Theorem 1.3 in Section 2. Note that the size of the good set G is at least $n - \frac{s}{2}$. Hence corrupting a δ-fraction of the bits of the data structure may cause a decoding failure for at most half of the queries $i \in S$ but not all. One may replace this factor $\frac{1}{2}$ easily by another constant (though the parameters t and τ will then change).

POLYNOMIAL EVALUATION: Let $\mathbb{Z}_n$ denote the set of integers modulo n and $s \le n$ be some nonnegative integer. Given a univariate polynomial $g \in \mathbb{Z}_n[X]$ of degree at most s, we would like to store g in a compact representation so that for each evaluation query $a \in \mathbb{Z}_n$, $g(a)$ can be computed efficiently. Formally, $D = \{g : g \in \mathbb{Z}_n[X], \deg(g) \le s\}$, $Q = \mathbb{Z}_n$, and $A = \mathbb{Z}_n$, and the function is $\text{POLYEVAL}_{n,s}(g, a) = g(a)$.

Since there are n^{s+1} polynomials of degree at most s, with each polynomial requiring a different instantiation of the data structure, the information-theoretic lower bound on the space of any data structure for this problem is at least $\log(n^{s+1}) \approx s \log n$ bits. Since each answer is an element of $\mathbb{Z}_n$ and must be represented by $\lfloor \log n \rfloor + 1$ bits, $\lfloor \log n \rfloor + 1$ is the information-theoretic lower bound on the bit-probe complexity.

Consider the following two naive solutions. On one hand, one can simply record the evaluations of g in a table with n entries, each with $\lfloor \log n \rfloor + 1$ bits. The length of this data structure is $O(n \log n)$ and each query requires reading only $\lfloor \log n \rfloor + 1$ bits. On the other hand, g can be stored as a table of its $s + 1$ coefficients. This gives a data structure of length and bit-probe complexity $(s + 1)(\lfloor \log n \rfloor + 1)$.

A natural question is whether one can construct a data structure that is optimal both in terms of space and time, i.e., has length $O(s \log n)$ and answers queries with $O(\log n)$ bit-probes. No such constructions are known to exist. However, some lower bounds are known in the weaker cell-probe model, where each cell is a sequence of $\lfloor \log n \rfloor + 1$ bits. For instance, as noted in [18], any data structure for POLYNOMIAL EVALUATION that stores $O(s^2)$ cells ($O(s^2 \log n)$ bits) requires reading at least $\Omega(s)$ cells. Moreover, by [17], if $\log n \gg s \log s$ and the data structure is constrained to store $s^{O(1)}$ cells, then its query complexity is $\Omega(s)$ cells. This implies that the second trivial construction described above is essentially optimal in the cell-probe model.

Recently, Kedlaya and Umans [15] obtained a data structure of length $s^{1+\eta} \log^{1+o(1)} n$ (where η is an arbitrarily small constant) that answers evaluation queries with $O(\text{polylog}\, s \cdot \log^{1+o(1)} n)$ bit-probes. These parameters exhibit the best tradeoff between s and n so far. When $s = n^{\eta}$ for some $0 < \eta < 1$, the data structure of Kedlaya and Umans [15] is much superior to the trivial solution: its length is nearly optimal, and the query complexity drops from $\text{poly}\, n$ to only $\text{polylog}\, n$ bit-probes.

Here we construct an error-correcting data structure for the polynomial evaluation problem that works even in the presence of adversarial noise, with length nearly linear in $s \log n$ and bit-probe complexity $O(\text{polylog}\, s \cdot \log^{1+o(1)} n)$. Formally:

Theorem 1.4. *For every $\varepsilon, \lambda, \eta \in (0,1)$, there exists $\tau \in (0,1)$ such that for all positive integers $s \le n$, for all $\delta \le \tau$, the data structure problem $\text{POLYEVAL}_{n,s}$ has a $(O(\text{polylog}\, s \cdot \log^{1+o(1)} n), \delta, \varepsilon, \lambda)$-data structure of length $O((s \log n)^{1+\eta})$.*

Remark 1.5. We note that Theorem 1.4 easily holds when $s = (\log n)^{o(1)}$. As we discussed previously, one can just store a table of the $s + 1$ coefficients of g. To make this error-correcting, encode the entire table by a standard error-correcting code. This has length and bit-probe complexity $O(s \log n) = O(\log^{1+o(1)} n)$.

1.2. Our techniques

At a high level, for both data structure problems we build our constructions by composing a relaxed locally decodable code with an appropriate noiseless data structure. If the underlying probe-accessing scheme in a noiseless data structure is "pseudorandom," then the noiseless data structure can be made error-correcting by appropriate compositions with other data structures. By pseudorandom, we mean that if a query is chosen uniformly at random from Q, then the positions of the probes selected also "behave" as if they are chosen uniformly at random. Such property allows us to analyze the error-tolerance of our constructions.

More specifically, for the MEMBERSHIP problem we build upon the noiseless data structure of Buhrman et al. [5]. While de Wolf [19] combined this with LDCs to get a rather long data structure with $\lambda = 0$, we will combine it here with RLDCs to get nearly optimal length with small (but non-zero) λ. In order to bound λ in our new construction, we make use of the fact that the [5]-construction is a bipartite *expander graph*, as explained below after Theorem 2.2. This property wasn't needed in [19]. The left side of the expander represents the set of queries, and a neighborhood of a query (a left node) represents the set of possible bit-probes that can be chosen to answer this query. The expansion property of the graph essentially implies that for a random query, the distribution of a bit-probe chosen to answer this query is close to uniform.[3] This property allows us to construct an efficient, error-correcting data structure for this problem.

For the polynomial evaluation problem, we rely upon the noiseless data structure of Kedlaya and Umans [15], which has a decoding procedure that uses the reconstruction algorithm from the Chinese Remainder Theorem. The property that we need is the simple fact that if a is chosen uniformly at random from $\mathbb{Z}_n$, then for any $m \leq n$, a modulo m is uniformly distributed in $\mathbb{Z}_m$. This implies that for a random evaluation point a, the distribution of certain tuples of cell-probes used to answer this evaluation point is close to uniform. This observation allows us to construct an efficient, error-correcting data structure for polynomial evaluation. Our construction follows the non-error-correcting one of [15] fairly closely; the main new ingredient is to add redundancy to their Chinese Remainder-based reconstruction by using more primes, which gives us the error-correcting features we need.

Time-complexity of decoding and encoding. So far we have used the number of bit-probes as a proxy for the actual time the decoder needs for query-answering. This is fairly standard, and usually justified by the fact that the actual time complexity of decoding is not much worse than its number of bit-probes. This is also the case for our constructions. For MEMBERSHIP, it can be shown that the decoder uses $O(1)$ probes and $\mathrm{polylog}(n)$ time (as do the RLDCs of [3]). For POLYNOMIAL EVALUATION, the decoder uses $\mathrm{polylog}(s) \log^{1+o(1)}(n)$ probes and $\mathrm{polylog}(sn)$ time.

The efficiency of *encoding*, i.e., the "pre-processing" of the data into the form of a data structure, for both our error-correcting data structures MEMBERSHIP and POLYNOMIAL EVALUATION

[3] We remark that this is different from the notion of smooth decoding in the LDC literature, which requires that for every *fixed* query, each bit-probe by itself is chosen with probability close to uniform (though not independent of the other bit-probes).

depends on the efficiency of encoding of the RLDC constructions in [3]. This is not addressed explicitly there, and needs further study.

2. The MEMBERSHIP problem

In this section we construct a data structure for the membership problem $\text{MEM}_{n,s}$. First we describe some of the building blocks that we need to prove Theorem 1.3. Our first basic building block is the relaxed locally decodable code of Ben-Sasson et al. [3] with nearly linear length. Using our terminology, we can restate their result as follows:

Theorem 2.1 (BGHSV [3])**.** *For every $\varepsilon \in (0, 1/2)$ and $\eta > 0$, there exist an integer $t > 0$ and reals $c > 0$ and $\tau > 0$, such that for every n and every $\delta \leq \tau$, the membership problem $\text{MEM}_{n,n}$ has a $(t, \delta, \varepsilon, c\delta)$-data structure for $\text{MEM}_{n,n}$ of length $O(n^{1+\eta})$.*

Note that by picking the error-rate δ a sufficiently small constant, one can set $\lambda = c\delta$ (the fraction of unrecoverable queries) to be very close to 0.

The other building block that we need is the following one-probe data structure of Buhrman et al. [5].

Theorem 2.2 (BMRV [5])**.** *For every $\varepsilon \in (0, 1/2)$ and for every positive integers $s \leq n$, there is an $(1, 0, \varepsilon, 0)$-data structure for $\text{MEM}_{n,s}$ of length $m = \frac{100}{\varepsilon^2} s \log n$ bits.*

Properties of the BMRV encoding: The encoding can be represented as a bipartite graph $\mathcal{G} = (L, R, E)$ with $|L| = n$ left vertices and $|R| = m$ right vertices, and regular left degree $d = \frac{\log n}{\varepsilon}$. This $\mathcal{G}$ is an *expander graph*: for each set $S \subseteq L$ with $|S| \leq 2s$, its neighborhood $\Gamma(S)$ satisfies $|\Gamma(S)| \geq \left(1 - \frac{\varepsilon}{2}\right)|S|d$. For each assignment of bits to the left vertices with at most s ones, the encoding specifies an assignment of bits to the right vertices. In other words, each $x \in \{0,1\}^n$ of weight $|x| \leq s$ corresponds to an assignment to the left vertices, and the m-bit encoding of x corresponds to an assignment to the right vertices.

For each $i \in [n]$ we write $\Gamma_i := \Gamma(\{i\})$ to denote the set of d neighbors of i. A crucial property of the encoding function $\mathcal{E}_{bmrv}$ is that for every x of weight $|x| \leq s$, for each $i \in [n]$, if $y = \mathcal{E}_{bmrv}(x) \in \{0,1\}^m$ then $\Pr_{j \in \Gamma_i}[x_i = y_j] \geq 1 - \varepsilon$. Hence the decoder for this data structure can just probe a random index $j \in \Gamma_i$ and return the resulting bit y_j. Note that this construction is not error-correcting at all, since $|\Gamma_i|$ errors in the data structure suffice to erase all information about the i-th bit of the encoded x. ∎

As we mentioned in the Section 1.1, by combining the BMRV encoding with the data structure for $\text{MEM}_{n,n}$ from Theorem 2.1, one easily obtains an $(O(1), \delta, \varepsilon, O(\delta))$-data structure for $\text{MEM}_{n,s}$ of length $O((s \log n)^{1+\eta})$. However, we can give an even more efficient, error-correcting data structure of length $O(s^{1+\eta} \log n)$. Our improvement follows an approach taken in de Wolf [19], which we now describe. For a vector $x \in \{0,1\}^n$ with $|x| \leq s$, consider a BMRV structure encoding $20n$ bits into m bits. The following "balls and bins estimate" is known:

Proposition 2.3 (From Section 2.3 of [19])**.** *For every positive integers $s \leq n$, the BMRV bipartite graph $\mathcal{G} = ([20n], [m], E)$ for $\text{MEM}_{20n,s}$ with error parameter $\frac{1}{10}$ and $m = 10^4 s \log(20n)$ has the following property: there exists a partition of $[m]$ into $b = 10 \log(20n)$ disjoint sets $B_1, \ldots, B_b$ of $10^3 s$ vertices each, such that for each $i \in [n]$, there are at least $\frac{b}{4}$ sets B_k satisfying $|\Gamma_i \cap B_k| = 1$.*

Proposition 2.3 suggests the following encoding and decoding procedures. To encode x, we rearrange the m bits of $\mathcal{E}_{bmrv}(x)$ into $\Theta(\log n)$ disjoint blocks of $\Theta(s)$ bits each, according to the

partition guaranteed by Proposition 2.3. Then for each block, encode these bits with the error-correcting data structure (RLDC) from Theorem 2.1. Given a received word w, to decode $i \in [n]$, pick a block B_k at random. With probability at least $\frac{1}{4}$, $\Gamma_i \cap B_k = \{j\}$ for some j. Run the RLDC decoder to decode the j-th bit of the k-th block of w. Since most blocks don't have much higher error-rate than the average (which is at most δ), with high probability we recover $\mathcal{E}_{bmrv}(x)_j$, which equals x_i with high probability. Finally, we can argue that most queries do not receive a blank symbol $\perp$ as an answer, using the expansion property of the BMRV encoding structure. Due to space limitation, we give only a proof sketch of Theorem 1.3 here.

Proof of Theorem 1.3. We only construct an error-correcting data structure with error probability 0.49. By a standard amplification technique we can reduce the error probability to any other positive constant (i.e., repeat the decoder $O(\log(1/\varepsilon))$ times).

By Theorem 2.2, there exists an encoder $\mathcal{E}_{bmrv}$ for an $(1, 0, \frac{1}{10}, 0)$-data structure for the membership problem $\text{MEM}_{20n,s}$ of length $m = 10^4 s \log(20n)$. Let $s' = 10^3 s$. By Theorem 2.1, for every $\eta > 0$, for some $t = O(1)$, and sufficiently small δ, $\text{MEM}_{s',s'}$ has a $(t, 10^5\delta, \frac{1}{100}, O(\delta))$-data structure of length $s'' = O(s'^{1+\eta})$. Let $\mathcal{E}_{bghsv}$ and $\mathcal{D}_{bghsv}$ be its encoder and decoder, respectively.

Encoding. Let $B_1, \ldots, B_b$ be a partition of $[m]$ as guaranteed by Proposition 2.3. For a string $w \in \{0,1\}^m$, we abuse notation and write $w = w_{B_1} \cdots w_{B_b}$ to denote the string obtained from w by applying the permutation on $[m]$ according to the partition $B_1, \ldots, B_b$. In other words, w_{B_k} is the concatenation of w_i where $i \in B_k$. We now describe the encoding process.

Encoder $\mathcal{E}$: on input $x \in \{0,1\}^n$, $|x| \le s$,

(1) Let $y = \mathcal{E}_{bmrv}\left(x0^{19n}\right)$ and write $y = y_{B_1} \ldots y_{B_b}$.
(2) Output the concatenation $\mathcal{E}(x) = \mathcal{E}_{bghsv}\left(y_{B_1}\right) \ldots \mathcal{E}_{bghsv}\left(y_{B_b}\right)$.

The length of $\mathcal{E}(x)$ is $N = b \cdot O(s'^{1+\eta}) = O(s^{1+\eta} \log n)$.

Decoding. Given a string $w \in \{0,1\}^N$, we write $w = w^{(1)} \ldots w^{(b)}$, where for $k \in [b]$, $w^{(k)}$ denotes the s''-bit string $w_{s''\cdot(k-1)+1} \ldots w_{s''\cdot k}$.

Decoder $\mathcal{D}$: on input i and with oracle access to a string $w \in \{0,1\}^N$,

(1) Pick a random $k \in [b]$.
(2) If $|\Gamma_i \cap B_k| \neq 1$, then output a random bit.
Else, let $\Gamma_i \cap B_k = \{j\}$. Run and output the answer given by the decoder $\mathcal{D}_{bghsv}(j)$, with oracle access to the s''-bit string $w^{(k)}$.

Analysis. We defer the analysis to the full version [6]. ∎

3. The POLYNOMIAL EVALUATION problem

In this section we prove Theorem 1.4. Given a polynomial g of degree s over $\mathbb{Z}_n$, our goal is to write down a data structure of length roughly linear in $s \log n$ so that for each $a \in \mathbb{Z}_n$, $g(a)$ can be computed with roughly $\text{polylog}\, s \cdot \log n$ bit-probes. Our data structure is built on the work of Kedlaya and Umans [15]. Since we cannot quite use their construction as a black-box, we first give a high-level overview of our proof, motivating each of the proof ingredients that we need.

Encoding based on reduced polynomials: The most naive construction, by recording $g(a)$ for each $a \in \mathbb{Z}_n$, has length $n \log n$ and answers an evaluation query with $\log n$ bit-probes. As explained in [15], one can reduce the length by using the Chinese Remainder Theorem (CRT): If P_1 is a collection of distinct primes, then a nonnegative integer $m < \prod_{p \in P_1} p$ is uniquely specified by (and can be reconstructed efficiently from) the values $[m]_p$ for each $p \in P_1$, where $[m]_p$ denotes $m \mod p$.

Consider the value $g(a)$ over $\mathbb{Z}$, which can be bounded above by n^{s+2}, for $a \in \mathbb{Z}_n$. Let P_1 consist of the first $\log(n^{s+2})$ primes. For each $p \in P_1$, compute the reduced polynomial $g_p := g \mod p$ and write down $g_p(b)$ for each $b \in \mathbb{Z}_p$. Consider the data structure that simply concatenates the evaluation table of every reduced polynomial. This data structure has length $|P_1|(\max_{p \in P_1} p)^{1+o(1)}$, which is $s^{2+o(1)} \log^{2+o(1)} n$ by the Prime Number Theorem. Note that $g(a) < \prod_{p \in P_1} p$. So to compute $[g(a)]_n$, it suffices to apply CRT to reconstruct $g(a)$ over $\mathbb{Z}$ from the values $[g(a)]_p = g_p([a]_p)$ for each $p \in P_1$. The number of bit-probes is $|P_1| \log(\max_{p \in P_1} p)$, which is $s^{1+o(1)} \log^{1+o(1)} n$.

Error-correction with reduced polynomials: The above CRT-based construction has terrible parameters, but it serves as an important building block from which we can obtain a data structure with better parameters. For now, we explain how the above CRT-based encoding can be made error-correcting. One can protect the bits of the evaluation tables of each reduced polynomial by an RLDC as provided by Theorem 2.1. However, the evaluation tables can have non-binary alphabets, and a bit-flip in just one "entry" of an evaluation table can destroy the decoding process. To remedy this, one can first encode each entry by a standard error-correcting code and then encode the concatenation of all the tables by an RLDC. This is encapsulated in Lemma 3.1, which can be viewed as a version of Theorem 2.1 over non-binary alphabet. We defer this proof to the full version of this paper [6].

Lemma 3.1. *Let $f : D \times Q \to \{0,1\}^\ell$ be a data structure problem. For every $\varepsilon, \eta, \lambda \in (0,1)$, there exists $\tau \in (0,1)$ such that for every $\delta \le \tau$, f has an $(O(\ell), \delta, \varepsilon, \lambda)$-data structure of length $O((\ell|Q|)^{1+\eta})$.*

To apply Lemma 3.1, let D be the set of degree-s polynomials over $\mathbb{Z}_n$, Q be the set of all evaluation points of all the reduced polynomials of g (each $q \in Q$ specified by a pair (a,p) of an evaluation point a and a prime modulus p), and the data structure problem f outputs evaluations of some reduced polynomial of g.

By itself, Lemma 3.1 cannot guarantee resilience against noise. In order to apply the CRT to reconstruct $g(a)$, all the values $\{[g(a)]_p : p \in P_1\}$ must be correct, which is not guaranteed by Lemma 3.1. To fix this, we add redundancy, taking a larger set of primes than necessary so that the reconstruction via CRT can be made error-correcting. Specifically, we apply a Chinese Remainder Code, or CRT code for short, to the encoding process.

Definition 3.2 (CRT code)**.** Let $p_1 < p_2 < \ldots < p_N$ be distinct primes, $K < N$, and $T = \prod_{i=1}^{K} p_i$. The *Chinese Remainder Code (CRT code)* with basis $p_1, \ldots, p_N$ and rate $\frac{K}{N}$ over message space $\mathbb{Z}_T$ encodes $m \in \mathbb{Z}_T$ as $\langle [m]_{p_1}, [m]_{p_2}, \ldots, [m]_{p_N} \rangle$.

Remark 3.3. By CRT, for distinct $m_1, m_2 \in \mathbb{Z}_T$, their encodings agree on at most $K-1$ coordinates. Hence the Chinese Remainder Code with basis $p_1 < \ldots < p_N$ and rate $\frac{K}{N}$ has distance $N - K + 1$.

It is known that good families of CRT code exist and that unique decoding algorithms for CRT codes can correct up to almost half of the distance of the code (see e.g., [12]). The following statement can be easily derived from known facts, and we defer its proof to the full version [6].

Theorem 3.4. *For every positive integer T, there exists a set P consisting of distinct primes, with (1) $|P| = O(\log T)$, and (2) $\forall p \in P$, $\log T < p < 500 \log T$, such that a CRT code with basis P and message space $\mathbb{Z}_T$ has rate $\frac{1}{2}$, relative distance $\frac{1}{2}$, and can correct up to a $(\frac{1}{4} - O(\frac{1}{\log \log T}))$-fraction of errors.*

We apply Theorem 3.4 to a message space of size n^{s+2} to obtain a set of primes P_1 with the properties described above. Note that these primes are all within a constant factor of one another, and in particular, the evaluation table of each reduced polynomial has the same length, up to a constant factor. This fact and Lemma 3.1 will ensure that our CRT-based encoding is error-correcting.

Reducing the bit-probe complexity: We now explain how to reduce the bit-probe complexity of the CRT-based encoding, using an idea from [15]. Write $s = d^m$, where $d = \log^C s$, $m = \frac{\log s}{C \log \log s}$, and $C > 1$ is a sufficiently large constant. Consider the following multilinear extension map $\psi_{d,m} : \mathbb{Z}_n[X] \to \mathbb{Z}_n[X_0, \ldots, X_{m-1}]$ that sends a univariate polynomial of degree at most s to an m-variate polynomial of degree less than d in each variable. For every $i \in [s]$, write $i = \sum_{j=0}^{m-1} i_j d^j$ in base d. Define $\psi_{d,m}$ which sends X^i to $X_1^{i_0} \cdots X_m^{i_{m-1}}$ and extends multilinearly to $\mathbb{Z}_n[X]$.

To simplify our notation, we write $\tilde{g}$ to denote the multivariate polynomial $\psi_{d,m}(g)$. For every $a \in \mathbb{Z}_n$, define $\tilde{a} \in \mathbb{Z}_n^m$ to be $([a]_n, [a^d]_n, [a^{d^2}]_n, \ldots, [a^{d^{m-1}}]_n)$. Note that for every $a \in \mathbb{Z}_n$, $g(a) = \tilde{g}(\tilde{a}) \pmod n$. Now the trick is to observe that the total degree of the multilinear polynomial $\tilde{g}$ is less than the degree of the univariate polynomial g, and hence its maximal value over the integers is much reduced. In particular, for every $a \in \mathbb{Z}_n^m$, the value $\psi_{d,m}(g)(a)$ over the integers is bounded above by $d^m n^{dm+1}$.

We now work with the reduced polynomials of $\tilde{g}$ for our encoding. Let P_1 be the collection of primes guaranteed by Theorem 3.4 when $T_1 = d^m n^{dm+1}$. For $p \in P_1$, let $\tilde{g}_p$ denote $\tilde{g} \mod p$ and $\tilde{a}_p$ denote the point $([a]_p, [a^d]_p, \ldots, [a^{d^{m-1}}]_p)$. Consider the data structure that concatenates the evaluation table of $\tilde{g}_p$ for each $p \in P_1$. For each $a \in \mathbb{Z}_n$, to compute $g(a)$, it suffices to compute $\tilde{g}(\tilde{a})$ over $\mathbb{Z}$, which by Theorem 3.4 can be reconstructed (even with noise) from the set $\{\tilde{g}_p(\tilde{a}_p) : p \in P_1\}$.

Since the maximum value of $\tilde{g}$ is at most $T_1 = d^m n^{dm+1}$ (whereas the maximum value of g is at most $d^m n^{d^m+1}$), the number of primes we now use is significantly less. This effectively reduces the bit-probe complexity. In particular, each evaluation query can be answered with $|P_1| \cdot \max_{p \in P_1} \log p = (dm \log n)^{1+o(1)}$ bit-probes, which by our choice of d and m is equal to $\operatorname{polylog} s \cdot \log^{1+o(1)} n$. However, the *length* of this encoding is still far from the information-theoretically optimal $s \log n$ bits. We shall explain how to reduce the length, but since encoding with multilinear reduced polynomials introduces potential complications in error-correction, we first explain how to circumvent these complications.

Error-correction with reduced multivariate polynomials: There are two complications that arise from encoding with reduced multivariate polynomials. The first is that not all the points in the evaluation tables are used in the reconstructive CRT algorithm. Lemma 3.1 only guarantees that most of the entries of the table are decoded correctly with high probability, but not all of them (even if the fraction of errors in the table is low, a λ-fraction of queries may be answered by $\perp$). So if the entries that are used in the reconstruction via CRT are not decoded by Lemma 3.1, then the whole decoding procedure fails.

More specifically, to reconstruct $\tilde{g}(\tilde{a})$ over $\mathbb{Z}_n$, it suffices to query the point $\tilde{a}_p$ in the evaluation table of $\tilde{g}_p$ for each $p \in P_1$. Typically the set $\{\tilde{a}_p : a \in \mathbb{Z}_n\}$ will be much smaller than $\mathbb{Z}_p^m$, so not all the points in $\mathbb{Z}_p^m$ are used. To circumvent this issue, we only store the query points that are used in the CRT reconstruction. Let $B^p = \{\tilde{a}_p : a \in \mathbb{Z}_n\}$. For each $p \in P_1$, the encoding only stores the evaluation of $\tilde{g}_p$ at the points B^p instead of the entire domain $\mathbb{Z}_p^m$. The disadvantage of computing the evaluation at the points in B^p is that the encoding stage takes time proportional to n. We thus give up on encoding efficiency (which was one of the main goals of Kedlaya and Umans) in order to guarantee error-correction.

The second complication is that the sizes of the evaluation tables may no longer be within a constant factor of each other. (This is true even if the evaluation points come from all of $\mathbb{Z}_p^m$.) If one of the tables has length significantly longer than the others, then a constant fraction of noise may completely corrupt the entries of all the other small tables, rendering decoding via CRT impossible. This potential problem is easy to fix; we apply a repetition code to each evaluation table so that all the tables have equal length.

Reducing the length: Now we explain how to reduce the length of the data structure to nearly $s \log n$, along the lines of Kedlaya and Umans [15]. To reduce the length, we need to reduce the magnitude of the primes used by the CRT reconstruction. We can effectively achieve that by applying the CRT twice. Instead of storing the evaluation table of $\tilde{g}_p$, we apply CRT again and store evaluation tables of the reduced polynomials of $\tilde{g}_p$ instead. Whenever an entry of $\tilde{g}_p$ is needed, we can apply the CRT reconstruction to the reduced polynomials of $\tilde{g}_p$.

Note that for $p_1 \in P_1$, the maximum value of $\tilde{g}_{p_1}$ (over the integers rather than mod n) is at most $T_2 = d^m p_1^{dm+1}$. Now apply Theorem 3.4 with T_2 the size of the message space to obtain a collection of primes P_2. Recall that each $p_1 \in P_1$ is at most $O(dm \log n)$. So each $p_2 \in P_2$ is at most $O((dm)^{1+o(1)} \log\log n)$, which also bounds the cardinality of P_2 from above.

For each query, the number of bit-probes made is at most $|P_1||P_2| \max_{p_2 \in P_2} \log p_2$, which is at most $(dm)^{2+o(1)} \log^{1+o(1)} n$. Recall that by our choice of d and m, $dm = \frac{\log^{C+1} s}{C \log\log s}$. Thus, the bit-probe complexity is $\mathrm{polylog}\, s \cdot \log^{1+o(1)} n$. Now, by Lemma 3.1, the length of the encoding is nearly linear in $|P_1||P_2| \max_{p_2 \in P_2} p_2^m \log p_2$, which is at most $\mathrm{polylog}\, s \cdot \log^{1+o(1)} n \cdot \max_{p_2 \in P_2} p_2^m$. So it suffices to bound $\max_{p_2 \in P_2} p_2^m$ from above. To this end, recall that by the remark following Theorem 1.4, we may assume without loss of generality that $s = \Omega(\log^\zeta n)$ for some $0 < \zeta < 1$. This implies that $\log\log\log n \leq \log\log s - \log\zeta$. Then for each $p_2 \in P_2$,

$$\begin{aligned} p_2^m &\leq \left(O\left((dm)^{1+o(1)} \log\log n\right)\right)^m \\ &\leq (dm)^{(1+o(1))m} \cdot s^{\frac{1}{C}+o(1)}. \end{aligned}$$

It is easy to see that $(dm)^{(1+o(1))m}$ can be bounded above by $s^{(1+o(1))(1+\frac{1}{C}-o(1))}$. Thus, $p_2^m = s^{1+\frac{2}{C}+o(1)}$. Putting everything together, the length of the encoding is nearly linear in $s \log n$. As mentioned, we defer the formal proof to the full version of this paper [6].

Acknowledgments

We thank Madhu Sudan for helpful comments and suggestions on the presentation of this paper.

References

[1] Y. Aumann and M. Bender. Fault-tolerant data structures. In *Proceedings of 37th IEEE FOCS*, pages 580–589, 1996.

[2] A. Beimel, Y. Ishai, E. Kushilevitz, and J. Raymond. Breaking the $O(n^{1/(2k-1)})$ barrier for information-theoretic Private Information Retrieval. In *Proceedings of 43rd IEEE FOCS*, pages 261–270, 2002.

[3] E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan, and S. Vadhan. Robust PCPs of proximity, shorter PCPs and applications to coding. *SIAM Journal on Computing*, 36(4):889–974, 2006. Earlier version in STOC'04.

[4] G. Brodal, R. Fagerberg, I. Finocchi, F. Grandoni, G. Italiano, A. Jørgenson, G. Moruz, and T. Mølhave. Optimal resilient dynamic dictionaries. In *Proceedings of 15th European Symposium on Algorithms (ESA)*, pages 347–358, 2007.

[5] H. Buhrman, P. B. Miltersen, J. Radhakrishnan, and S. Venkatesh. Are bitvectors optimal? *SIAM Journal on Computing*, 31(6):1723–1744, 2002. Earlier version in STOC'00.

[6] V. Chen, E. Grigorescu, and R. de Wolf. Efficient and Error-Correcting Data Structures for Membership and Polynomial Evaluation, 2009. Preprint at http://arxiv.org/abs/0909.3696.

[7] K. Efremenko. 3-query locally decodable codes of subexponential length. In *Proceedings of 41st ACM STOC*, 2009.

[8] I. Finocchi, F. Grandoni, and G. Italiano. Optimal resilient sorting and searching in the presence of memory faults. In *Proceedings of 33rd ICALP*, volume 4051 of *Lecture Notes in Computer Science*, pages 286–298, 2006.

[9] I. Finocchi, F. Grandoni, and G. Italiano. Resilient search trees. In *Proceedings of 18th ACM-SIAM SODA*, pages 547–553, 2007.

[10] I. Finocchi and G. Italiano. Sorting and searching in the presence of memory faults (without redundancy). In *Proceedings of 36th ACM STOC*, pages 101–110, 2004.

[11] M. Fredman, M. Komlós, and E. Szemerédi. Storing a sparse table with $O(1)$ worst case access time. *Journal of the ACM*, 31(3):538–544, 1984.

[12] O. Goldreich, D. Ron, and M. Sudan. Chinese remaindering with errors. *IEEE Transactions on Information Theory*, 46(4):1330–1338, 2000.

[13] A. G. Jørgenson, G. Moruz, and T. Mølhave. Resilient priority queues. In *Proceedings of 10th International Workshop on Algorithms and Data Structures (WADS)*, volume 4619 of *Lecture Notes in Computer Science*, 2007.

[14] J. Katz and L. Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *Proceedings of 32nd ACM STOC*, pages 80–86, 2000.

[15] K. S. Kedlaya and C. Umans. Fast modular composition in any characteristic. In *Proceedings of 49th IEEE FOCS*, pages 146–155, 2008.

[16] I. Kerenidis and R. de Wolf. Exponential lower bound for 2-query locally decodable codes via a quantum argument. *Journal of Computer and System Sciences*, 69(3):395–420, 2004. Earlier version in STOC'03. quant-ph/0208062.

[17] P. B. Miltersen. On the cell probe complexity of polynomial evaluation. *Theor. Comput. Sci.*, 143(1):167–174, 1995.

[18] P. B. Miltersen. Cell probe complexity - a survey. Invited paper at *Advances in Data Structures* workshop. Available at Miltersen's homepage, 1999.

[19] R. de Wolf. Error-correcting data structures. In *Proceedings of 26th Annual Symposium on Theoretical Aspects of Computer Science (STACS'2009)*, pages 313–324, 2009. cs.DS/0802.1471.

[20] D. Woodruff. New lower bounds for general locally decodable codes. Technical report, ECCC Report TR07–006, 2006.

[21] S. Yekhanin. Towards 3-query locally decodable codes of subexponential length. *Journal of the ACM*, 55(1), 2008. Earlier version in STOC'07.