Submitted to the Symposium on Theoretical Aspects of Computer Science
www.stacs-conf.org

ERROR-CORRECTING DATA STRUCTURES

RONALD DE WOLF [1]

[1] CWI. Kruislaan 413. 1098SJ Amsterdam. The Netherlands. E-mail address: rdewolf@cwi.nl

Abstract. We study data structures in the presence of adversarial noise. We want to encode a given object in a succinct data structure that enables us to efficiently answer specific queries about the object, even if the data structure has been corrupted by a constant fraction of errors. This new model is the common generalization of (static) data structures and locally decodable error-correcting codes. The main issue is the tradeoff between the space used by the data structure and the time (number of probes) needed to answer a query about the encoded object. We prove a number of upper and lower bounds on various natural error-correcting data structure problems. In particular, we show that the optimal length of error-correcting data structures for the Membership problem (where we want to store subsets of size s from a universe of size n) is closely related to the optimal length of locally decodable codes for s-bit strings.

1. Introduction

Data structures deal with one of the most fundamental questions of computer science: how can we store certain objects in a way that is both space-efficient and that enables us to efficiently answer questions about the object? Thus, for instance, it makes sense to store a set as an ordered list or as a heap-structure, because this is space-efficient and allows us to determine quickly (in time logarithmic in the size of the set) whether a certain element is in the set or not. From a complexity-theoretic point of view, the aim is usually to study the tradeoff between the two main resources of the data structure: the length/size of the data structure (storage space) and the efficiency with which we can answer specific queries about the stored object. To make this precise, we measure the length of the data structure in bits, and measure the efficiency of query-answering in the number of *probes*, i.e., the number of bit-positions in the data structure that we look at in order to answer a query. The following is adapted from Miltersen's survey [Mil99]:

Definition 1.1. Let D be a set of data items, Q be a set of queries, A be a set of answers, and $f : D \times Q \to A$. A (p, ε)-*data structure* for f of length N is a map $\phi : D \to \{0,1\}^N$ for which there is a randomized algorithm $\mathcal{A}$ that makes at most p probes to its oracle and satisfies $\Pr[\mathcal{A}^{\phi(x)}(q) = f(x,q)] \geq 1 - \varepsilon$ for every $q \in Q$ and $x \in D$.

1998 ACM Subject Classification: E1, E4.

Key words and phrases: data structures, error-correcting codes, locally decodable codes, membership.

Partially supported by a Veni grant from the Netherlands Organization for Scientific Research (NWO), and by the European Commission under Integrated Project QAP, IST 015848.

Usually we will study the case $D \subseteq \{0,1\}^n$ and $A = \{0,1\}$. Most standard data structures taught in undergraduate computer science are deterministic, and hence have error probability $\varepsilon = 0$. As mentioned, the main complexity issue here is the tradeoff between N and p. Some data structure problems that we will consider are the following:

- EQUALITY. $D = Q = \{0,1\}^n$, and $f(x,y) = 1$ if $x = y$, $f(x,y) = 0$ if $x \neq y$. This is not a terribly interesting data structure problem in itself, since for every x there is only one query y for which the answer is '1'; we merely mention this data structure problem here because it will be used to illustrate some definitions later on.
- MEMBERSHIP. $D = \{x \in \{0,1\}^n : \text{Hamming weight } |x| \leq s\}$, $Q = [n] := \{1,\ldots,n\}$, and $f(x,i) = x_i$. In other words, x corresponds to a set of size at most s from a universe of size n, and we want to store the set in a way that easily allows us to make membership queries. This is probably the most basic and widely-studied data structure problem of them all [FKS84, Yao81, BMRV00, RSV02]. Note that for $s = 1$ this is EQUALITY on $\log n$ bits, while for $s = n$ it is the general MEMBERSHIP problem without constraints on the set.
- SUBSTRING. $D = \{0,1\}^n$, $Q = \{y \in \{0,1\}^n : |y| \leq r\}$, $f(x,y) = x_y$, where x_y is the $|y|$-bit substring of x indexed by the 1-bits of y (e.g., $1010_{0110} = 01$). For $r = 1$ it is MEMBERSHIP.
- INNER PRODUCT ($\mathrm{IP}_{n,r}$). $D = \{0,1\}^n$, $Q = \{y \in \{0,1\}^n : |y| \leq r\}$ and $f(x,y) = x \cdot y \bmod 2$. This problem is among the hardest Boolean problems where the answer depends on at most r bits of x (again, for $r = 1$ it is MEMBERSHIP).

More complicated data structure problems such as RANK, PREDECESSOR, NEAREST NEIGHBOR have also been studied a lot, but we will not consider them here.

One issue that the above definition ignores, is the issue of *noise*. Memory and storage devices are not perfect: the world is full of cosmic rays, small earthquakes, random (quantum) events, bypassing trams, etc., that can cause a few errors here and there. Another potential source of noise is transmission of the data structure over some noisy channel. Of course, better hardware can partly mitigate these effects, but in many situations it is realistic to expect a small fraction of the bits in the storage space to become corrupted over time. Our goal in this paper is to study *error-correcting* data structures. These still enable efficient computation of $f(x,q)$ from the stored data structure $\phi(x)$, even if the latter has been corrupted by a constant fraction of errors. In analogy with the usual setting for error-correcting codes [MS77, vL98], we will take a pessimistic, adversarial view of errors here: we want to be able to deal with a constant fraction of errors *no matter where they are placed*. Formally, we define error-correcting data structures as follows.

Definition 1.2. Let D be a set of data items, Q be a set of queries, A be a set of answers, and $f : D \times Q \to A$. A (p,δ,ε)-*error-correcting data structure* for f of length N is a map $\phi : D \to \{0,1\}^N$ for which there is a randomized algorithm $\mathcal{A}$ that makes at most p probes to its oracle and satisfies $\Pr[\mathcal{A}^y(q) = f(x,q)] \geq 1 - \varepsilon$ for every $q \in Q$, every $x \in D$, and every $y \in \{0,1\}^N$ at distance $\Delta(y,\phi(x)) \leq \delta N$.

Definition 1.1 is the special case of Definition 1.2 where $\delta = 0$.[1] Note that if $\delta > 0$ then the adversary can always set the errors in a way that gives the decoder $\mathcal{A}$ a non-zero error probability. Hence the setting with bounded error probability is the natural one for error-correcting data structures. This contrasts with the standard noiseless setting, where one usually considers deterministic structures.

A simple example of an efficient error-correcting data structure is for EQUALITY: encode x with a good error-correcting code $\phi(x)$. Then $N = O(n)$, and we can decode by one probe: given y, probe $\phi(x)_j$ for uniformly chosen $j \in [N]$, compare it with $\phi(y)_j$, and output 1 iff these two bits are equal. If up to a δ-fraction of the bits in $\phi(x)$ are corrupted, then we will give the correct answer with probability $1-\delta$ in the case $x = y$. If the distance between any two codewords is close to $N/2$ (which is true for instance for a random linear code), then we will give the correct answer with probability about $1/2-\delta$ in the case $x \neq y$. These two probabilities can be balanced to 2-sided error $\varepsilon = 1/3 + 2\delta/3$. The error can be reduced further by allowing more than one probe.

We only deal with so-called *static* data structures here: we do not worry about updating the x that we are encoding. What about *dynamic* data structures, which allow efficient updates as well as efficient queries to the encoded object? Note that if data-items x and x' are distinguishable in the sense that $f(x,q) \neq f(x',q)$ for at least one query $q \in Q$, then their respective error-correcting encodings $\phi(x)$ and $\phi(x')$ will have distance $\Omega(N)$.[2] Hence updating the encoded data from x to x' will require $\Omega(N)$ changes in the data structure, which shows that a dynamical version of our model of error-correcting data structures with efficient updates is not possible.

Error-correcting data structures not only generalize the standard (static) data structures (Definition 1.1), but they also generalize *locally decodable codes*, defined as:

Definition 1.3. A (p,δ,ε)-*locally decodable code* (LDC) of length N is a map $\phi : \{0,1\}^n \to \{0,1\}^N$ for which there is a randomized algorithm $\mathcal{A}$ that makes at most p probes to its oracle and satisfies $\Pr[\mathcal{A}^y(i) = x_i] \geq 1-\varepsilon$ for every $i \in [n]$, every $x \in \{0,1\}^n$, and every $y \in \{0,1\}^N$ at distance $\Delta(y,\phi(x)) \leq \delta N$.

Note that a (p,δ,ε)-error-correcting data structure for MEMBERSHIP (with $s = n$) is exactly a (p,δ,ε)-locally decodable code. Much work has been done on LDCs, but their length-vs-probes tradeoff is still largely unknown for $p \geq 3$. We refer to [Tre04] and the references therein.

LDCs address only a very simple type of data structure problem: we have an n-bit "database" and want to be able to retrieve individual bits from it. In practice, databases have more structure and complexity, and one usually asks more complicated queries, such as retrieving all records within a certain range. Our more general notion of error-correcting data structures enables a study of such more practical data structure problems in the presence of adversarial noise.

[1] As [BMRV00, end of Section 1.1] notes, a data structure can be viewed as locally decodable source code. With this information-theoretic point of view, an *error-correcting* data structure is a locally decodable combined source-*channel* code, and our results for MEMBERSHIP show that one can sometimes do better than combining the best source code with the best channel code. We thank one of the anonymous referees for pointing this out.

[2] Hence if all pairs $x, x' \in D$ are distinguishable (which is usually the case), ϕ is an error-correcting code.

Comment on terminology. The terminologies used in the data-structure and LDC-literature conflict at various points, and we needed to reconcile them somehow. To avoid confusion, let us repeat here the choices we made. We reserve the term "query" for the question q one asks about the encoded data x, while accesses to bits of the data structure are called "probes" (in contrast, these are usually called "queries" in the LDC-literature). The number of probes is denoted by p. We use n for the number of bits of the data item x (in contrast with the literature about MEMBERSHIP, which mostly uses m for the size of the universe and n for the size of the set). We use N for the length of the data structure (while the LDC-literature mostly uses m, except for Yekhanin [Yek07] who uses N as we do). We use the term "decoder" for the algorithm $\mathcal{A}$. Another issue is that ε is sometimes used as the error probability (in which case one wants $\varepsilon \approx 0$), and sometimes as the bias away from 1/2 (in which case one wants $\varepsilon \approx 1/2$). We use the former.

1.1. Our results

If one subscribes to the approach to errors taken in the area of error-correcting codes, then our definition of error-correcting data structures seems a very natural one. Yet, to our knowledge, it is new and has not been studied before (see Section 1.2 for other approaches).

1.1.1. MEMBERSHIP. The most basic data structure problem is probably MEMBERSHIP. Fortunately, our main positive result for error-correcting data structures applies to this.

Fix some number of probes p, noise level δ, and allowed error probability ε, and consider the minimal length of p-probe error-correcting data structures for s-out-of-n MEMBERSHIP. Let us call this minimal length $\mathrm{MEM}(p,s,n)$. A first observation is that such a data structure is actually a locally decodable code for s bits: just restrict attention to n-bit strings whose last $n-s$ bits are all 0. Hence, with $\mathrm{LDC}(p,s)$ denoting the minimal length among all p-probe LDCs that encode s bits (for our fixed ε,δ), we immediately get the obvious lower bound

$$\mathrm{LDC}(p,s) \leq \mathrm{MEM}(p,s,n).$$

This bound is close to optimal if $s \approx n$. Another trivial lower bound comes from the observation that our data structure for MEMBERSHIP is a map with domain of size $B(n,s) := \sum_{i=0}^{s} \binom{n}{i}$ and range of size 2^N that has to be injective. Hence we get another obvious bound

$$\Omega(s\log(n/s)) \leq \log B(n,s) \leq \mathrm{MEM}(p,s,n).$$

What about *upper* bounds? Something that one can always do to construct error-correcting data structures for any problem, is to take the optimal non-error-correcting p_1-probe construction and encode it with a p_2-probe LDC. If the error probability of the LDC is much smaller than $1/p_1$, then we can just run the decoder for the non-error-correcting structure, replacing each of its p_1 probes by p_2 probes to the LDC. This gives an error-correcting data structure with $p = p_1p_2$ probes. In the case of MEMBERSHIP, the optimal non-error-correcting data structure of Buhrman et al. [BMRV00] uses only 1 probe and $O(s\log n)$ bits. Encoding this with the best possible p-probe LDC gives error-correcting data structures for MEMBERSHIP of length $\mathrm{LDC}(p, O(s\log n))$. For instance for $p=2$ we can use the Hadamard code[3] for s bits, giving upper bound $\mathrm{MEM}(2,s,n) \leq \exp(O(s\log n))$.

[3]The Hadamard code of $x \in \{0,1\}^s$ is the codeword of length 2^s obtained by concatenating the bits $x\cdot y \pmod 2$ for all $y \in \{0,1\}^s$. It can be decoded by two probes, since for every $y \in \{0,1\}^s$ we have $(x\cdot y) \oplus (x\cdot(y\oplus e_i)) = x_i$. Picking y at random, decoding from a δ-corrupted codeword will be correct with

Our main positive result in Section 2 says that something much better is possible—the max of the above two lower bounds is not far from optimal. Slightly simplifying[4], we prove

$$\mathrm{MEM}(p, s, n) \leq O(\mathrm{LDC}(p, 1000s) \log n).$$

In other words, if we have a decent p-probe LDC for encoding $O(s)$-bit strings, then we can use this to encode sets of size s from a much larger universe $[n]$, at the expense of blowing up our data structure by only a factor of $\log n$. For instance, for $p = 2$ probes we get $\mathrm{MEM}(2, s, n) \leq \exp(O(s)) \log n$ from the Hadamard code, which is much better than the earlier $\exp(O(s \log n))$. For $p = 3$ probes, we get $\mathrm{MEM}(3, s, n) \leq \exp(\exp(\sqrt{\log s})) \log n$ from Efremenko's recent 3-probe LDC [Efr08] (which improved Yekhanin's breakthrough construction [Yek07]). Our construction relies heavily on the MEMBERSHIP construction of [BMRV00]. Note that the near-tightness of the above upper and lower bounds implies that progress (meaning better upper and/or lower bounds) on locally decodable codes for any number of probes is *equivalent* to progress on error-correcting data structures for s-out-of-n MEMBERSHIP.

1.1.2. INNER PRODUCT. In Section 3 we analyze the inner product problem, where we are encoding $x \in \{0,1\}^n$ and want to be able to compute the dot product $x \cdot y \pmod 2$, for any $y \in \{0,1\}^n$ of weight at most r. We first study the non-error-correcting setting, where we can prove nearly matching upper and lower bounds (this is not the error-correcting setting, but provides something to compare it with). Clearly, a trivial 1-probe data structure is to store the answers to all $B(n, r)$ possible queries separately. In Section 3.1 we use a discrepancy argument from communication complexity to prove a lower bound of about $B(n, r)^{1/p}$ on the length of p-probe data structures. This shows that the trivial solution is essentially optimal if $p = 1$. We also construct various p-probe error-correcting data structures for inner product. For small p and large r, their length is not much worse than the best non-error-correcting structures. The upshot is that inner product is a problem where data structures can sometimes be made error-correcting at little extra cost compared to the non-error-correcting case—admittedly, this is mostly because the non-error-correcting solutions for $\mathrm{IP}_{n,r}$ are already very expensive in terms of length.

1.2. Related work

Much work has of course been done on locally decodable codes, a.k.a. error-correcting data structures for the MEMBERSHIP problem without constraints on the set size [Tre04]. However, the error-correcting version of s-out-of-n MEMBERSHIP ("storing sparse tables") or of other possible data structure problems has not been studied before.[5] Here we briefly describe some other approaches to data structures in the presence of memory errors. There is also much work on data structures with faulty *processors*, but we won't discuss that.

probability at least $1 - 2\delta$, because both probes y and $y \oplus e_i$ are individually random and hence probe a corrupted entry with probability at most δ. This exponential length is optimal for 2-probe LDCs [KW04].

[4]Our actual result, Theorem 2.2, is a bit dirtier, with some deterioration in the error and noise parameters.

[5]Using the connection between information-theoretical private information retrieval and locally decodable codes, one may derive some error-correcting data structures from the PIR results of [CIK+01]. However, the resulting structures seem fairly weak.

Fault-tolerant pointer-based data structures. Aumann and Bender [AB96] study fault-tolerant versions of *pointer-based* data structures. They define a pointer-based data structure as a directed graph where the edges are pointers, and the nodes come in two types: information nodes carry real data, while auxiliary nodes carry auxiliary or structural data. An *error* is the destruction of a node and its outgoing edges. They assume such an error is detected when accessing the node. Even a few errors may be very harmful to pointer-based data structures: for instance, losing one pointer halfway a standard linked list means we lose the second half of the list. They call a data structure (d, g)-*fault-tolerant* (where d is an integer that upper bounds the number of errors, and g is a function) if $f \leq d$ errors cause at most $g(f)$ information nodes to be lost.

Aumann and Bender present fault-tolerant *stacks* with $g(f) = O(f)$, and fault-tolerant *linked lists* and *binary search trees* with $g(f) = O(f \log d)$, with only a constant-factor overhead in the size of the data structure, and small computational overhead. Note, however, that their error-correcting demands are weaker than ours: we require that *no* part of the data is lost (every query should be answered with high success probability), even in the presence of a constant fraction of errors. Of course, we pay for that in terms of length.

Faulty-memory RAM model. An alternative model of error-correcting data structures is the "faulty-memory RAM model", introduced by Finocchi and Italiano [FI04]. In this model, one assumes there are $O(1)$ incorruptible memory cells available. This is justified by the fact that CPU registers are much more robust than other kinds of memory. On the other hand, all other memory cells can be faulty—including the ones used by the algorithm that is answering queries (something our model does not consider). The model assumes an upper bound Δ on the number of errors.

Finocchi, Grandoni, and Italiano described essentially optimal resilient algorithms for *sorting* that work in $O(n \log n + \Delta^2)$ time with Δ up to about $\sqrt{n}$; and for *searching* in $\Theta(\log n + \Delta)$ time. There is a lot of recent work in this model: Jørgenson et al. [JMM07] study resilient *priority queues*, Finocchi et al. [FGI07] study resilient *search trees*, and Brodal et al. [BFF+07] study resilient *dictionaries*. This interesting model allows for more efficient data structures than our model, but its disadvantages are also clear: it assumes a small number of incorruptible cells, which may not be available in many practical situations (for instance when the whole data structure is stored on a hard disk), and the constructions mentioned above cannot deal well with a constant noise rate.

2. The Membership problem

2.1. Noiseless case: the BMRV data structure for Membership

Our error-correcting data structures for Membership rely heavily on the construction of Buhrman et al. [BMRV00], whose relevant properties we sketch here. Their structure is obtained using the probabilistic method. Explicit but slightly less efficient structures were subsequently given by Ta-Shma [TS02].

The BMRV-structure maps $x \in \{0,1\}^n$ (of weight $\leq s$) to a string $y := y(x) \in \{0,1\}^{n'}$ of length $n' = \frac{100}{\varepsilon^2} s \log n$ that can be decoded with one probe if $\delta = 0$. More precisely, for every $i \in [n]$ there is a set $P_i \subseteq [n']$ of size $\log(n)/\varepsilon$, such that for every x of weight $\leq s$:

$$\Pr_{j \in P_i}[y_j = x_i] \geq 1 - \varepsilon, \tag{2.1}$$

where the probability is over a uniform index $j \in P_i$. For fixed ε, the length $n' = O(s \log n)$ is optimal up to a constant factor, because clearly $\log \binom{n}{s}$ is a lower bound.

2.2. Noisy case: 1 probe

For the noiseless case, the BMRV data structure has information-theoretically optimal length $O(s \log n)$ and decodes with the minimal number of probes (one). This can also be achieved in the error-correcting case if $s = 1$: then we just have the EQUALITY problem, for which see the remark following Definition 1.2. For larger s, one can observe that the BMRV-structure still works with high probability if $\delta \ll 1/s$: in that case the total number of errors is $\delta n' \ll \log n$, so for each i, most bits in the $\Theta(\log n)$-set P_i are uncorrupted.

Theorem 2.1 (BMRV)**.** *There exist* $(1, \Omega(1/s), 1/4)$*-error-correcting data structures for* MEMBERSHIP *of length* $N = O(s \log n)$*.*

This only works if $\delta \ll 1/s$, which is actually close to optimal, as follows. An s-bit LDC can be embedded in an error-correcting data structure for MEMBERSHIP, hence it follows from Katz-Trevisan's [KT00, Theorem 3] that there are no 1-probe error-correcting data structures for MEMBERSHIP if $s > 1/(\delta(1 - H(\varepsilon)))$ (where $H(\cdot)$ denotes binary entropy). In sum, there are 1-probe error-correcting data structures for MEMBERSHIP of information-theoretically optimal length if $\delta \ll 1/s$. In contrast, if $\delta \gg 1/s$ then there are no 1-probe error-correcting data structures at all, not even of exponential length.

2.3. Noisy case: $p > 1$ probes

As we argued in the introduction, for fixed ε and δ there is an easy lower bound on the length N of p-probe error-correcting data structures for s-out-of-n MEMBERSHIP:

$$N \geq \max\left(\mathrm{LDC}(p, s), \log \sum_{i=0}^{s} \binom{n}{i}\right).$$

Our nearly matching upper bound, below, uses the ε-error data structure of [BMRV00] for some small fixed ε. A simple way to obtain a p-probe error-correcting data structure is just to encode their $O(s \log n)$-bit string y with the optimal p-probe LDC (with error ε', say), which gives length $\mathrm{LDC}(p, O(s \log n))$. The one probe to y is replaced by p probes to the LDC. By the union bound, the error probability of the overall construction is at most $\varepsilon + \varepsilon'$. This, however, achieves more than we need: this structure enables us to recover y_j for every j, whereas it would suffice to recover y_j for most $j \in P_i$ (for each $i \in [n]$).

Definition of the data structure and decoder. To construct a shorter error-correcting data structure, we proceed as follows. Let δ be a small constant (e.g. 1/10000); this is the noise level we want our final data structure for MEMBERSHIP to protect against. Consider the BMRV-structure for s-out-of-n MEMBERSHIP, with error probability at most 1/10. Then $n' = 10000 s \log n$ is its length, and $b = 10 \log n$ is the size of each of the sets P_i. Apply now a random permutation π to y (we show below that π can be fixed to a specific permutation). View the resulting n'-bit string as made up of $b = 10 \log n$ consecutive blocks of $1000s$ bits each. We encode each block with the optimal $(p, 100\delta, 1/100)$-LDC that encodes $1000s$ bits. Let ℓ be the length of this LDC. This gives overall length $N = 10\ell \log n$. The decoding procedure is as follows. Randomly choose a $k \in [b]$. This picks out one of the blocks. If this kth block contains exactly one $j \in P_i$ then recover y_j from the (possibly corrupted) LDC

for that block, using the p-probe LDC-decoder, and output y_j. If the kth block contains 0 or more than 1 elements from P_i, then output a uniformly random bit.

Analysis. Our goal below is to show that we can fix the permutation π such that for at least $n/20$ of the indices $i \in [n]$, this procedure has good probability of correctly decoding x_i (for all x of weight $\leq s$). The intuition is as follows. Thanks to the random permutation and the fact that $|P_i|$ equals the number of blocks, the expected intersection between P_i and a block is exactly 1. Hence for many $i \in [n]$, many blocks will contain exactly one index $j \in P_i$. Moreover, for most blocks, their LDC-encoding won't have too many errors, hence we can recover y_j using the LDC-decoder for that block. Since $y_j = x_i$ for 90% of the $j \in P_i$, we usually recover x_i.

To make this precise, call $k \in [b]$ "good for i" if block k contains *exactly one* $j \in P_i$, and let X_{ik} be the indicator random variable for this event. Call $i \in [n]$ "good" if at least $b/4$ of the blocks are good for i (i.e., $\sum_{k\in[b]} X_{ik} \geq b/4$), and let X_i be the indicator random variable for this event. The expected value (over uniformly random π) of each X_{ik} is the probability that if we randomly place b balls into ab positions (a is the block-size $1000s$), then there is exactly one ball among the a positions of the first block, and the other $b-1$ balls are in the last $ab-a$ positions. This is

$$\frac{a\binom{ab-a}{b-1}}{\binom{ab}{b}} = \frac{(ab-b)(ab-b-1)\cdots(ab-b-a+2)}{(ab-1)(ab-2)\cdots(ab-a+1)} \geq \left(\frac{ab-b-a+2}{ab-a+1}\right)^{a-1} \geq \left(1-\frac{1}{a-1}\right)^{a-1}$$

The righthand side goes to $1/e \approx 0.37$ with large a, so we can safely lower bound it by $3/10$. Then, using linearity of expectation:

$$\frac{3bn}{10} \leq \mathrm{Exp}\left[\sum_{i\in[n],k\in[b]} X_{ik}\right] \leq b \cdot \mathrm{Exp}\left[\sum_{i\in[n]} X_i\right] + \frac{b}{4}\left(n - \mathrm{Exp}\left[\sum_{i\in[n]} X_i\right]\right),$$

which implies $\mathrm{Exp}\left[\sum_{i\in[n]} X_i\right] \geq \frac{n}{20}$. Hence we can fix one permutation π such that at least $n/20$ of the indices i are good.

For every index i, at least 90% of all $j \in P_i$ satisfy $y_j = x_i$. Hence for a good index i, with probability at least $1/4 - 1/10$ we pick a k such that the kth block is good for i *and* the unique $j \in P_i$ in the kth block satisfies $y_j = x_i$. By Markov's inequality, the probability that the picked block has more than a 100δ-fraction of errors, is less than $1/100$. If the fraction of errors is at most 100δ, then our LDC-decoder recovers the relevant bit y_j with probability $99/100$. Hence the overall probability of outputting the correct x_i is at least

$$\frac{3}{4}\cdot\frac{1}{2} + \left(\frac{1}{4} - \frac{1}{10} - \frac{1}{100}\right)\cdot\frac{99}{100} > \frac{51}{100}.$$

We end up with an error-correcting data structure for Membership for a universe of size $n/20$ instead of n elements, but we can fix this by starting with the BMRV-structure for $20n$ bits.

We summarize this construction in a theorem:

Theorem 2.2. *If there exists a $(p, 100\delta, 1/100)$-LDC of length ℓ that encodes $1000s$ bits, then there exists a $(p, \delta, 49/100)$-error-correcting data structure of length $O(\ell \log n)$ for the s-out-of-n Membership problem.*

The error and noise parameters of this new structure are not great, but they can be improved by more careful analysis. We here sketch a better solution without giving all technical details. Suppose we change the decoding procedure for x_i as follows: pick $j \in P_i$ uniformly at random, decode y_j from the LDC of the block where y_j sits, and output the result. There are three sources of error here: (1) the BMRV-structure makes a mistake (i.e., j happens to be such that $y_j \neq x_i$), (2) the LDC-decoder fails because there is too much noise on the LDC that we are decoding from, (3) the LDC-decoder fails even though there is not too much noise on it. The 2nd kind is hardest to analyze. The adversary will do best if he puts just a bit more than the tolerable noise-level on the encodings of blocks that contain the most $j \in P_i$, thereby "destroying" those encodings.

For a random permutation, we expect that about $b/(e \cdot m!)$ of the b blocks contain m elements of P_i. Hence about $1/65$ of all blocks have 4 or more elements of P_i. If the LDC is designed to protect against a 65δ-fraction of errors within one encoded block, then with overall error-fraction δ, the adversary has exactly enough noise to "destroy" all blocks containing 4 or more elements of P_i. The probability that our uniformly random j sits in such a "destroyed" block is about

$$\sum_{m \geq 4} \frac{m}{b} \frac{b}{e \cdot m!} = \frac{1}{e} \left(\frac{1}{3!} + \frac{1}{4!} + \cdots \right) \approx 0.08.$$

Hence if we set the error of the BMRV-structure to $1/10$ and the error of the LDC to $1/100$ (as above), then the total error probability for decoding x_i is less than 0.2 (of course we need to show that we can fix a π such that good decoding occurs for a good fraction of all $i \in [n]$). Another parameter that may be adjusted is the block size, which we here took to be $1000s$. Clearly, different tradeoffs between codelength, tolerable noise-level, and error probability are possible.

3. The Inner product problem

3.1. Noiseless case

Here we show bounds for Inner product, first for the noiseless case ($\delta = 0$).

Upper bound. Consider all strings z of weight at most $\lceil r/p \rceil$. The number of such z is $B(n, \lceil r/p \rceil) = \sum_{i=0}^{\lceil r/p \rceil} \binom{n}{i} \leq (epn/r)^{r/p}$. We define our codeword by writing down, for all z in lexicographic order, the inner product $x \cdot z$ mod 2. If we want to recover the inner product $x \cdot y$ for some y of weight at most r, we write $y = z_1 + \cdots + z_p$ for z_j's of weight at most $\lceil r/p \rceil$ and recover $x \cdot z_j$ for each $j \in [p]$, using one probe for each. Summing the results of the p probes gives $x \cdot y$ (mod 2). For $p = 1$ probes, the length is $B(n, r)$.

Lower bound. To prove a nearly-matching lower bound, we use Miltersen's technique of relating a data structure to a two-party communication game [Mil94]. We refer to [KN97] for a general introduction to communication complexity. Suppose Alice gets string $x \in \{0,1\}^n$, Bob gets string $y \in \{0,1\}^n$ of weight $\leq r$, and they need to compute $x \cdot y$ (mod 2) with bounded error probability and minimal communication between them. Call this communication problem $\mathrm{IP}_{n,r}$. Let $B(n, r) = \sum_{i=0}^{r} \binom{n}{i}$ be the size of Q, i.e., the number of possible queries y. For reasons of space we skip the proof of the following communication complexity lower bound, which may be found in the full version of this paper.

Theorem 3.1. *Every communication protocol for* $\mathrm{IP}_{n,r}$ *with worst-case (or even average-case) success probability* $\geq 1/2+\beta$ *needs at least* $\log(B(n,r)) - 2\log(1/2\beta)$ *bits of communication.*

Armed with this communication bound we lower bound data structure length:

Theorem 3.2. *Every* (p,ε)*-data structure for* $\mathrm{IP}_{n,r}$ *needs* $N \geq \frac{1}{2}2^{(\log(B(n,r))-2\log(1/(1-2\varepsilon))-1)/p}$.

Proof. We will use the data structure to obtain a communication protocol for $\mathrm{IP}_{n,r}$ that uses $p(\log(N)+1)+1$ bits of communication, and then invoke Theorem 3.1 to obtain the lower bound. Alice holds x, and hence $\phi(x)$, while Bob simulates the decoder. Bob starts the communication. He picks his first probe to the data structure and sends it over in $\log N$ bits. Alice sends back the 1-bit answer. After p rounds of communication, all p probes have been simulated and Bob can give the same output as the decoder would have given. Bob's output will be the last bit of the communication. Theorem 3.1 now implies $p(\log(N)+1)+1 \geq \log(B(n,r)) - 2\log(1/(1-2\varepsilon))$. Rearranging gives the bound on N. ∎

For fixed ε, the lower bound is $N = \Omega\left(B(n,r)^{1/p}\right)$. This is $\Omega((n/r)^{r/p})$, which (at least for small p) is not too far from the upper bound of approximately $(epn/r)^{r/p}$ mentioned above. Note that in general our bound on N is superpolynomial in n whenever $p = o(r)$. For instance, when $r = \alpha n$ for some constant $\alpha \in (0,1/2)$ then $N = \Omega(2^{nH(\alpha)/p})$, which is non-trivial whenever $p = o(n)$. Finally, note that the proof technique also works if Alice's messages are longer than 1 bit (i.e., if the code is over a larger-than-binary alphabet).

3.2. Noisy case

3.2.1. *Constructions for* Substring. One can easily construct error-correcting data structures for Substring, which also suffice for Inner product. Note that since we are recovering r bits, and each probe gives at most one bit of information, by information theory we need at least about r probes to the data structure. Our solutions below will use $O(r\log r)$ probes. View x as a concatenation $x = x^{(1)}\dots x^{(r)}$ of r strings of n/r bits each (we ignore rounding for simplicity), and define $\phi(x)$ as the concatenation of the Hadamard codes of these r pieces. Then $\phi(x)$ has length $N = r\cdot 2^{n/r}$.

If $\delta \geq 1/4r$ then the adversary could corrupt one of the r Hadamard codes by 25% noise, ensuring that some of the bits of x are irrevocably lost even when we allow the full N probes. However, if $\delta \ll 1/r$ then we can recover each bit x_i with small constant error probability by 2 probes in the Hadamard codeword where i sits, and with error probability $\ll 1/r$ using $O(\log r)$ probes. Hence we can compute $f(x,y) = x_y$ with error close to 0 using $p = O(r\log r)$ probes (or with $2r$ probes if $\delta \ll 1/r^2$). This also implies that *any* data structure problem where $f(x,q)$ depends on at most some fixed constant r bits of x, has an error-correcting data structure of length $N = r\cdot 2^{n/r}$, $p = O(r\log r)$, and that works if $\delta \ll 1/r$. Alternatively, we can take Efremenko's [Efr08] or Yekhanin's 3-probe LDC [Yek07], and just decode each of the r bits separately. Using $O(\log r)$ probes to recover a bit with error probability $\ll 1/r$, we recover the r-bit string x_y using $p = O(r\log r)$ probes even if δ is a constant independent of r.

3.2.2. *Constructions for* INNER PRODUCT. Going through the proof of [Yek07], it is easy to see that it allows us to compute the parity of any set of r bits from x using at most $3r$ probes with error ε, if the noise rate δ is at most $\varepsilon/(3r)$ (just add the results of the 3 probes one would make for each bit in the parity). To get error-correcting data structures even for small constant p (independent of r), we can adapt the polynomial schemes from [BIK05] to get the following theorem. The details are given in the full version of this paper.

Theorem 3.3. *For every $p \geq 2$, there exists a $(p, \delta, p\delta)$-error-correcting data structure for $\mathrm{IP}_{n,r}$ of length $N \leq p \cdot 2^{r(p-1)^2 n^{1/(p-1)}}$.*

For the $p = 2$ case, we get something simpler and better from the Hadamard code. This code, of length 2^n, actually allows us to compute $x \cdot y \pmod 2$ for any $y \in \{0,1\}^n$ of our choice, with 2 probes and error probability at most 2δ (just probe z and $y \oplus z$ for uniformly random $z \in \{0,1\}^n$ and observe that $(x \cdot z) \oplus (x \cdot (y \oplus z)) = x \cdot y$). Note that for $r = \Theta(n)$ and $p = O(1)$, even non-error-correcting data structures need length $2^{\Theta(n)}$ (Theorem 3.2). This is an example where error-correcting data structures are not significantly longer than the non-error-correcting kind.

4. Future work

Many questions are opened up by our model of error-correcting data structures, e.g.:

- There are plenty of other natural data structure problems, such as RANK, PREDECESSOR, versions of NEAREST NEIGHBOR etc. [Mil99]. What about the length-vs-probes tradeoffs for their error-correcting versions? The obvious approach is to put the best known LDC on top of the best known non-error-correcting data structures. This is not always optimal, though—for instance in the case of s-out-of-n MEMBERSHIP one can do significantly better, as we showed.
- It is often natural to assume that a memory cell contains not a bit, but some number from, say, a polynomial-size universe. This is called the *cell-probe* model [Yao81], in contrast to the *bit-probe* model we considered here. Probing a cell gives $O(\log n)$ bits at the same time, which can significantly improve the length-vs-probes tradeoff and is worth studying. Still, we view the bit-probe approach taken here as more fundamental than the cell-probe model. A p-probe cell-probe structure is a $O(p \log n)$-probe bit-probe structure, but not vice versa. Also, the way memory is addressed in actual computers in constant chunks of, say, 8 or 16 bits at a time, is closer in spirit to the bit-probe model than to the cell-probe model.
- Zvi Lotker suggested to me the following connection with distributed computing. Suppose the data structure is distributed over N processors, each holding one bit. Interpreted in this setting, an error-correcting data structure allows an honest party to answer queries about the encoded object while communicating with at most p processors. The answer will be correct with probability $1 - \varepsilon$, even if up to a δ-fraction of the N processors are faulty or even malicious (the querier need not know where the faulty/malicious sites are).

Acknowledgments

Thanks to Nitin Saxena for many useful discussions, to Harry Buhrman and Jaikumar Radhakrishnan for discussions about [BMRV00], to Zvi Lotker for the connection with distributed computation mentioned in Section 4, to Peter Bro Miltersen for a pointer to [JMM07], and to Gabriel Moruz for sending me a copy of that paper.

References

[AB96] Y. Aumann and M. Bender. Fault-tolerant data structures. In *Proceedings of 37th IEEE FOCS*, pages 580–589, 1996.

[BFF+07] 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 ESA*, pages 347–358, 2007.

[BIK05] A. Beimel, Y. Ishai, and E. Kushilevitz. General constructions for information-theoretical Private Information Retrieval. *Journal of Computer and System Sciences*, 72(2):247–281, 2005.

[BMRV00] H. Buhrman, P. B. Miltersen, J. Radhakrishnan, and S. Venkatesh. Are bitvectors optimal? *SIAM Journal on Computing*, 31(6):1723–1744, 2002.

[CIK+01] R. Canetti, Y. Ishai, R. Kumar, M. Reiter, R. Rubinfeld, and R. Wright. Selective private function evaluation with applications to private statistics. In *Proceedings of 20th ACM PODC*, pages 293–304, 2001.

[Efr08] K. Efremenko. 3-Query locally decodable codes of subexponential length. ECCC Report TR08–069, 2008.

[FGI07] I. Finocchi, F. Grandoni, and G. Italiano. Resilient search trees. In *Proceedings of 18th ACM-SIAM SODA*, pages 547–553, 2007.

[FI04] 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.

[FKS84] 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.

[JMM07] A. G. Jørgenson, G. Moruz, and T. Mølhave. Resilient priority queues. In *Proceedings of 10th WADS*, volume 4619 of *Lecture Notes in Computer Science*, Springer, 2007.

[KN97] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[KT00] 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.

[KW04] 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.

[Mil94] P. B. Miltersen. Lower bounds for Union-Split-Find related problems on random access machines. In *Proceedings of 26th ACM STOC*, pages 625–634, 1994.

[Mil99] P. B. Miltersen. Cell probe complexity - a survey. Invited paper at *Advances in Data Structures* workshop. Available at Miltersen's homepage, 1999.

[MS77] F. MacWilliams and N. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, 1977.

[RSV02] J. Radhakrishnan, P. Sen, and S. Venkatesh. The quantum complexity of set membership. *Algorithmica*, 34(4):462–479, 2002.

[Tre04] L. Trevisan. Some applications of coding theory in computational complexity. *Quaderni di Matematica*, 13:347–424, 2004.

[TS02] A. Ta-Shma. Storing information with extractors. *Information Processing Letters*, 83(5):267–274, 2002.

[vL98] J. H. van Lint. *Introduction to Coding Theory*. Springer, third edition, 1998.

[Yao81] A. C-C. Yao. Should tables be sorted? *Journal of the ACM*, 28(3):615–628, 1981.

[Yek07] S. Yekhanin. Towards 3-query locally decodable codes of subexponential length. In *Proceedings of 39th ACM STOC*, pages 266–274, 2007.