Fault-Tolerant Data Structures

Ronald de Wolf*
CWI Amsterdam

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 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 fault-tolerant data structure problems. In particular, we show that the optimal length of fault-tolerant 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 will measure the length of the data structure in bits, and the efficiency of query-answering in the number of *probes*, i.e. the number of bit-positions in the data structure that we need to look at in order to answer a query.

The following definition is adapted from Miltersen's survey [Mil99].

Definition 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 exists a randomized algorithm $\mathcal{A}$ that makes at most p probes to its oracle and that satisfies*

$$\Pr[\mathcal{A}^{\phi(x)}(q) = f(x,q)] \geq 1 - \varepsilon,$$

for every $q \in Q$ and $x \in D$.

*rdewolf@cwi.nl. Partially supported by a Veni grant from the Netherlands Organization for Scientific Research (NWO), and by the European Commission under the Integrated Project Qubit Applications (QAP) funded by the IST directorate as Contract Number 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 $x \neq y$.
- MEMBERSHIP. $D = \{x \in \{0,1\}^n : |x| \leq s\}$, $Q = [n]$, 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] (often this is studied in the *cell-probe* rather than our bit-probe model). Note that for $s = 1$ this is EQUALITY on $\log n$ bits.
- 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 that is indexed by the 1-bits of y (for example, $1010_{0110} = 01$).
- 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 essentially the hardest Boolean problem where the answer depends on at most r bits of x (for $r = 1$ and with additional constraint $|x| \leq s$, this 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. Another issue that we will not deal with in this paper, is the ability to efficiently *update* a data structure (so-called "dynamic" data structures, in contrast to the above "static" type). We refer to [Mil99] for more details.

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 *fault-tolerant* 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, 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 fault-tolerant data structures as follows.

Definition 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)*-fault-tolerant data structure *for f of length N is a map $\phi : D \to \{0,1\}^N$ for which there exists a randomized algorithm $\mathcal{A}$ that makes at most p probes to its oracle and that satisfies*

$$\Pr[\mathcal{A}^y(q) = f(x, q)] \geq 1 - \varepsilon,$$

for every $q \in Q$ and every $y \in \{0,1\}^N$ with Hamming distance $\Delta(y, \phi(x)) \leq \delta N$ for some $x \in D$.

Definition 1 is the special case of Definition 2 where $\delta = 0$. Note that if $\delta > 0$ then the adversary can always set the errors in a way that will give the decoder $\mathcal{A}$ a non-zero error probability. Hence the bounded-error setting is the natural setting for fault-tolerant data structures. This contrasts with the standard noiseless setting, where one usually considers deterministic data structures.

A simple example of an efficient fault-tolerant 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 $\varepsilon \approx 1/3 + \delta$. The error can be reduced further by allowing more than one probe.

Fault-tolerant data structures not only generalize the standard (static) data structures (Definition 1), but they also generalize *locally decodable codes*. These are defined as follows:

Definition 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 exists a randomized algorithm* $\mathcal{A}$ *that makes at most* p *probes to its oracle and that satisfies*

$$\Pr[\mathcal{A}^y(i) = x_i] \geq 1 - \varepsilon,$$

for every $y \in \{0,1\}^N$ *with Hamming distance* $\Delta(y, \phi(x)) \leq \delta N$ *for some* $x \in \{0,1\}^n$.

Note that a (p, δ, ε)-fault-tolerant 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.

1.1 Our results

Despite the fact that our fault-tolerant data structures appear to be a very natural common generalization of both standard data structures and locally decodable codes, to our knowledge they have not been studied before (there has been some related work in different models, see below). In this paper we present a number of initial results that show that the model has some merit.

1.1.1 MEMBERSHIP

The most basic data structure problem is probably the MEMBERSHIP problem. Fortunately, the main positive result we managed to prove for fault-tolerant data structures applies to this problem.

Fix some number of probes p, noise level δ, and allowed error probability ε, and consider the minimal length of p-probe fault-tolerant data structures for s-out-of-n MEMBERSHIP. Let us call this minimal length $\text{MEM}(p, s, n)$. 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 $\text{LDC}(p, s)$ denoting the minimal length among all p-probe LDCs that encode s bits (for our fixed ε, δ), we immediately get the lower bound

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

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

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

What about upper bounds? Something that one can always do to construct fault-tolerant data structures for any problem, is to take the optimal non-fault-tolerant 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-fault-tolerant structure, replacing each of its p_1 probes by p_2 probes to the LDC. This gives a fault-tolerant data structure with a total of $p = p_1 p_2$ probes. In the case of MEMBERSHIP, the optimal non-fault-tolerant 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 fault-tolerant data structures for MEMBERSHIP of length $\mathrm{LDC}(p, O(s \log n))$. For instance for $p = 2$ we can use the Hadamard code for s bits[1], giving upper bound $\mathrm{MEM}(2, s, n) \leq 2^{O(s \log n)}$.

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[2], 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 fault-tolerantly 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 $\mathrm{MEM}(2, s, n) \leq 2^{O(s)} \log n$ from the Hadamard code, which is much better than the earlier $2^{O(s \log n)}$. For $p = 3$ probes, we get $\mathrm{MEM}(3, s, n) \leq 2^{O(s^{1/t})} \log n$ for any Mersenne prime $2^t - 1$ from Yekhanin's recent 3-probe LDC [Yek07].

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-fault-tolerant setting. 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 fault-tolerant data structures for inner product. For small p and large r, their length is not much worse than the best non-fault-tolerant structures. The upshot is that inner product is a problem where data structures can sometimes be made fault-tolerant at little extra cost compared to the non-fault-tolerant case—admittedly, this is mostly because the non-fault-tolerant solutions for $\mathrm{IP}_{n,r}$ are already very expensive in terms of their length.

1.2 Related work

Much work has of course been done on fault-tolerant data structures for the MEMBERSHIP problem without constraints on the set size (a.k.a. general locally decodable codes). However, to our knowledge, the general fault-tolerant version of MEMBERSHIP or of other possible data structure problems has not been studier before. Using the connection between information-theoretical private

[1] The Hadamard code of $x \in \{0, 1\}^s$ is the code 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 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].

[2] Our actual result, Theorem 2, is a bit dirtier, with some deterioration in the error and noise parameters.

information retrieval and locally decodable codes [KT00], one may derive some fault-tolerant data structures from the PIR results of [CIK$^+$01]. However, the resulting structures seem fairly weak.

An alternative model of fault-tolerant 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. NB: here δ is the total number of errors, not a fraction as in our earlier definitions.

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. Jørgenson, Moruz, and Mølhave [JMM07] constructed a resilient *priority queue* that uses an optimal $O(n)$ space to store n elements, and allows insertion and deletion in amortized time $O(\log n + \delta)$. This interesting model allows for more efficient data structures than the model proposed here, 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.

Comment on terminology. The terminologies used in the data-structure and LDC-literature conflict at various points, and we needed to reconcile them somehow. Our choice is as follows. 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 thing 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.

2 The MEMBERSHIP problem

2.1 Noiseless case: the BMRV data structure for MEMBERSHIP

Our fault-tolerant 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 $S_i \subseteq [n']$ of size $S_i = \log(n)/\varepsilon$ such that for every x of weight $\leq s$:

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

where the probability is taken over a uniform index $j \in S_i$. For fixed ε, the length $n' = O(s \log n)$ of the BMRV-structure 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 fault-tolerant case if $s = 1$: then we just have the EQUALITY problem, for which see the remark following Definition 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 S_i are uncorrupted.

Theorem 1 (BMRV) *There exist* $(1, \Omega(1/s), 1/4)$*-fault-tolerant 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 a fault-tolerant data structure for MEMBERSHIP, hence it follows from Katz-Trevisan's [KT00, Theorem 3] that there are no 1-probe fault-tolerant data structures for MEMBERSHIP if $s > 1/(\delta(1 - H(\varepsilon)))$.

In sum, there are one-probe fault-tolerant 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 one-probe fault-tolerant 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 fault-tolerant 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 uses the ε-error data structure of [BMRV00] for some small fixed ε. A simple way to obtain a p-probe fault-tolerant 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 if we were able to recover y_j for most $j \in S_i$.

Definition of the data structure and decoder. To construct a shorter fault-tolerant 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 S_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 $j \in S_i$ 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 S_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 $|S_i|$ equals the number of blocks, the expected intersection between S_i and a block is exactly 1. Hence for many $i \in [n]$, many blocks will contain exactly one index $j \in S_i$. 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 $j \in S_i$, we usually recover x_i.

To make this precise, call $k \in [b]$ "good" for i if block k contains *exactly one* $j \in S_i$, 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:

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

which implies

$$\mathrm{Exp}\left[\sum_{i=1}^n X_i\right] \geq \frac{n}{20}.$$

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 $j \in S_i$ satisfy $y_j = x_i$. Hence for a good index i, with probability at least $1/4 - 1/10$ we will pick a k such that the kth block is good for i *and* the unique $j \in S_i$ in the kth block satisfies $y_j = x_i$. By Markov's inequality, the probability that the block that we picked 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 value 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 a fault-tolerant data structure for MEMBERSHIP for a universe of size $n/20$ instead of n elements, put we can fix this by starting with the BMRV-structure for $20n$ bits.

We summarize this construction in a theorem:

Theorem 2 *If there exists a $(p, 100\delta, 1/100)$-LDC of length ℓ that encodes $1000s$ bits, then there exists a $(p, \delta, 49/100)$-fault tolerant 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 $j \in S_i$ 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 $j \in S_i$, thereby "destroying" those codes.

For a random permutation, we expect about $b/(e \cdot m!)$ of the b blocks contain m elements of S_i. Hence about $1/65$ of all blocks have 4 or more elements of S_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 S_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 case where there is no noise ($\delta = 0$).

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 code 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). In particular, 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. The proof of our communication complexity lower bound below uses a fairly standard discrepancy argument, but we have not found this specific result anywhere. For completeness we include a proof in Appendix A.

Theorem 3 *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 complexity bound we can lower bound data structure length:

Theorem 4 *Every* (p, ε)*-data structure for* $\mathrm{IP}_{n,r}$ *needs space* $N \geq \frac{1}{2} 2^{(\log(B(n,r)) - 2\log(1/(1-2\varepsilon)) - 1)/p}$

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 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 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 fault-tolerant 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.[3] Our solutions below will use $O(r \log r)$ probes. View x as a concatenation $x = x^{(1)} \ldots x^{(r)}$ 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 code 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 $\delta \ll 1/r^2$).[4] This also implies that *any* data structure problem where $f(x, q)$ depends on at most some fixed constant r bits of x, has a fault-tolerant data structure of length

[3] $d/(\log(N) + 1)$ probes in the case of *quantum* decoders.

[4] It follows from Buhrman et al. [BNRW07] that if we allow a *quantum* decoder, the factor of $\log r$ is not needed.

$N = r \cdot 2^{n/r}$, $p = O(r \log r)$, and that works if $\delta \ll 1/r$. Alternatively, we can take Yekhanin's 3-probe LDC [Yek07], of length $N \approx 2^{n^{1/t}}$ for every Mersenne prime $2^t - 1$, 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 Yekhanin's construction, 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 fault-tolerant 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 Appendix B.

Theorem 5 *For every $p \geq 2$, there exists a $(p, \delta, p\delta)$-fault-tolerant data structure for $\mathrm{IP}_{n,r}$ of length $N \leq p \cdot 2^{r(p-1)^2 n^{1/(p-1)}}$.*

The $p = 2$ case of this construction is essentially the Hadamard code. The Hadamard 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 r and $y \oplus r$ for uniformly random r and observe that $(x \cdot r) \oplus (x \cdot (r \oplus y)) = x \cdot y$). Note that for $r = \Theta(n)$ and $p = O(1)$, even non-fault-tolerant data structures need length $2^{\Theta(n)}$ (Theorem 4). This is an example where fault-tolerant data structures are not significantly more efficient than the regular, non-fault-tolerant kind.

4 Future work

Many questions are opened up by our model of fault-tolerant data structures. We mention a few:

- 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 fault-tolerant versions? The obvious approach is to put the best known LDC on top of the best known non-fault-tolerant data structures. This is not always optimal, though—for instance in the case of Membership one can do significantly better, as we showed here.

- 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, in contrast to the *bit-probe* model we considered here [Mil99]. Probing a cell gives $O(\log n)$ bits at the same time, which can significantly improve the length-vs-probes tradeoff.

- What about *dynamic* data structures, which allow efficient updates as well as efficient queries to the encoded object?

- 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, a fault-tolerant data structure allows honest parties to answer queries about the encoded object while communicating with at most p other 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 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 the faulty-memory RAM model, and to Gabriel Moruz for sending me a copy of that paper.

References

[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, 2000. Earlier version in STOC'00.

[BNRW07] H. Buhrman, I. Newman, H. Röhrig, and R. de Wolf. Robust polynomials and quantum algorithms. *Theory of Computing Systems*, 40(4):379–395, 2007. Special issue on STACS 2005. quant-ph/0309220.

[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 Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 293–304, 2001.

[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 International Workshop on Algorithms and Data Structures (WADS)*, volume 4619 of *Lecture Notes in Computer Science*, 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. Special issue on STOC'03. quant-ph/0208062.

[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.

[RSV02] J. Radhakrishnan, P. Sen, and S. Venkatesh. The quantum complexity of set membership. *Algorithmica*, 34(4):462–479, 2002. Earlier version in FOCS'00. quant-ph/0007021.

[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.

[Yao77] A. C-C. Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of 18th IEEE FOCS*, pages 222–227, 1977.

[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.

A Proof of Theorem 3

Let μ be the uniform input distribution: each x has probability $1/2^n$ and each y of weight $\leq r$ has probability $1/B(n,r)$. We show a lower bound on the communication c of *deterministic* protocols that compute $\mathrm{IP}_{n,r}$ with μ-probability at least $1/2+\beta$. By Yao's principle [Yao77], this lower bound then also applies to randomized protocols.

Consider a deterministic c-bit protocol. Assume the last bit communicated is the output bit. It is well-known that this partitions the input space into *rectangles* $R_1,\ldots,R_{2^c}$, where $R_i = A_i \times B_i$, and the protocol gives the same output bit a_i for each $(x,y) \in R_i$.[5] The *discrepancy* of rectangle $R = A \times B$ under μ is the difference between the weight of the 0s and the 1s in that rectangle:

$$\delta_\mu(R) = \left|\mu(R \cap \mathrm{IP}_{n,r}^{-1}(1)) - \mu(R \cap \mathrm{IP}_{n,r}^{-1}(0))\right|$$

We can show for every rectangle that its discrepancy is not very large:

Lemma 1 $\delta_\mu(R) \leq \dfrac{\sqrt{|R|}}{\sqrt{2^n}B(n,r)}$.

Proof. Let M be the $2^n \times B(n,r)$ matrix whose (x,y)-entry is $(-1)^{\mathrm{IP}_{n,r}(x,y)} = (-1)^{x\cdot y}$. It is easy to see that $M^TM = 2^nI$, where I is the $B(n,r)\times B(n,r)$ identity matrix. This implies, for any $v \in \mathbb{R}^{B(n,r)}$

$$\| Mv \|^2 = (Mv)^T\cdot(Mv) = v^TM^TMv = 2^nv^Tv = 2^n\| v \|^2.$$

Let $R = A \times B$, $v_A \in \{0,1\}^{2^n}$ and $v_B \in \{0,1\}^{B(n,r)}$ be the characteristic (column) vectors of the sets A and B. Note that $\| v_A \| = \sqrt{|A|}$ and $\| v_B \| = \sqrt{|B|}$. The sum of M-entries in R is $\sum_{a\in A,b\in B} M_{ab} = v_A^TMv_B$. We can bound this using Cauchy-Schwarz:

$$|v_A^TMv_B| \leq \| v_A \|\cdot\| Mv_B \| = \| v_A \|\cdot\sqrt{2^n}\| v_B \| = \sqrt{|A|\cdot|B|\cdot 2^n}.$$

[5] [KN97, Section 1.2]. The number of rectangles may be smaller than 2^c, but we can always add some empty rectangles.

Observing that $\delta_\mu(R) = |v_A^T M v_B|/(2^n B(n,r))$ and $|R| = |A| \cdot |B|$ concludes the proof. $\square$

Define the success and failure probabilities (under μ) of the protocol as

$$P_s = \sum_{i=1}^{2^c} \mu(R_i \cap \mathrm{IP}_{n,r}^{-1}(a_i)) \quad \text{and} \quad P_f = \sum_{i=1}^{2^c} \mu(R_i \cap \mathrm{IP}_{n,r}^{-1}(1-a_i))$$

Then

$$\begin{aligned}
2\beta &\leq P_s - P_f \\
&= \sum_i \mu(R_i \cap \mathrm{IP}_{n,r}^{-1}(a_i)) - \mu(R_i \cap \mathrm{IP}_{n,r}^{-1}(1-a_i)) \\
&\leq \sum_i \left|\mu(R_i \cap \mathrm{IP}_{n,r}^{-1}(a_i)) - \mu(R_i \cap \mathrm{IP}_{n,r}^{-1}(1-a_i))\right| \\
&= \sum_i \delta_\mu(R_i) \\
&\leq \frac{\sum_i \sqrt{|R_i|}}{\sqrt{2^n} B(n,r)} \\
&\leq \frac{\sqrt{2^c}\sqrt{\sum_i |R_i|}}{\sqrt{2^n} B(n,r)} \\
&= \sqrt{2^c/B(n,r)},
\end{aligned}$$

where the last inequality is Cauchy-Schwarz and the last equality holds because $\sum_i |R_i|$ is the total number of inputs, which is $2^n B(n,r)$.

Rearranging gives $2^c \geq (2\beta)^2 B(n,r)$, hence $c \geq \log(B(n,r)) - 2\log(1/2\beta)$.

B Proof of Theorem 5

Here our goal is to construct p-probe fault-tolerant data structures for the inner product problem. Let d be an integer to be determined later. Pick $m = \lceil dn^{1/d} \rceil$. Then $\binom{m}{d} \geq n$, so there exist n distinct sets $S_1, \ldots, S_n \subseteq [m]$, each of size d. For each $x \in \{0,1\}^n$, define an m-variate polynomial p_x of degree d over $\mathbb{F}_2$ by

$$p_x(z_1, \ldots, z_m) = \sum_{i=1}^{n} x_i \prod_{j \in S_i} z_j.$$

Note that if we identify S_i with its m-bit characteristic vector, then $p_x(S_i) = x_i$. For $z^{(1)}, \ldots, z^{(r)} \in \{0,1\}^m$, define an rm-variate polynomial $p_{x,r}$ over $\mathbb{F}_2$ by

$$p_{x,r}(z^{(1)}, \ldots, z^{(r)}) = \sum_{j=1}^{r} p_x(z^{(j)}).$$

This polynomial $p_{x,r}(z)$ has rm variables, degree d, and allows us to evaluate parities of any set of r of the variables of x: if $y \in \{0,1\}^n$ (of weight r) has its 1-bits at positions $i_1, \ldots, i_r$, then

$$p_{x,r}(S_{i_1}, \ldots, S_{i_r}) = \sum_{j=1}^{r} x_{i_j} = x \cdot y \pmod 2.$$

To construct a fault-tolerant data structure for $\mathrm{IP}_{n,r}$, it thus suffices to give a structure that enables us to evaluate $p_{x,r}$ at any point w of our choice.[6]

Let $w \in \{0,1\}^{rm}$. Suppose we "secret-share" this into p pieces $w^{(1)}, \ldots, w^{(p)} \in \{0,1\}^{rm}$ which are uniformly random subject to the constraint $w = w^{(1)} + \cdots + w^{(p)}$. Now consider the prm-variate polynomial $q_{x,r}$ defined by

$$q_{x,r}(w^{(1)}, \ldots, w^{(p)}) = p_{x,r}(w^{(1)} + \cdots + w^{(p)}).$$

Each monomial M in this polynomial has at most d variables. If we pick $d = p - 1$, then for every M there will be a $j \in [p]$ such that M does not contain variables from $w^{(j)}$. Assign all such monomials to a new polynomial $q_{x,r}^{(j)}$, which is independent of $w^{(j)}$. This allows us to write

$$q_{x,r}(w^{(1)}, \ldots, w^{(p)}) = q_{x,r}^{(1)}(w^{(2)}, \ldots, w^{(p)}) + \cdots + q_{x,r}^{(p)}(w^{(1)}, \ldots, w^{(p-1)}).$$

Note that each $q_{x,r}^{(j)}$ has domain of size $2^{(p-1)rm}$. The data structure is defined as the concatenation, for all $j \in [p]$, of the values of $q_{x,r}^{(j)}$ on all possible inputs. This has length

$$N = p \cdot 2^{(p-1)rm} = p \cdot 2^{r(p-1)^2 n^{1/(p-1)}}.$$

This length is $2^{O(rn^{1/(p-1)})}$ for $p = O(1)$.

Decoding is as follows: the decoder would like to evaluate $p_{x,r}$ on some point $w \in \{0,1\}^{rm}$. He picks $w^{(1)}, \ldots, w^{(p)}$ as above, and for all $j \in [p]$, probes the point $z^{(1)}, \ldots, z^{(j-1)}, z^{(j+1)}, \ldots, z^{(p)}$ in the jth block of the code. This, if uncorrupted, returns the value of $q_{x,r}^{(j)}$ at that point. The decoder outputs the sum of his p probes (mod 2). If none of the probed bits were corrupted, the output is $p_{x,r}(w)$ Note that the probe within the jth block is uniformly random in that block, so its error probability is exactly the fraction δ_j of errors in the jth block. If the overall fraction of errors in the data structure is at most δ, then we have $\frac{1}{p}\sum_{j=1}^{p} \delta_j \leq \delta$. Hence by the union bound, the total error probability is at most $\sum_{j=1}^{p} \delta_j \leq p\delta$.

[6] If we also want to be able to compute $x \cdot y \pmod 2$ for $|y| < r$, we can just add a dummy 0 as $(n+1)$st variable to x, and use its index $r - |y|$ times as inputs to $p_{x,r}$.