The Cell Probe Complexity of Succinct Data Structures

Anna Gál[1] and Peter Bro Miltersen[2]

[1] Dept. of Computer Science, University of Texas at Austin. panni@cs.utexas.edu
[2] Dept. of Computer Science, University of Aarhus. bromille@brics.dk

Abstract. We show lower bounds in the cell probe model for the redundancy/query time tradeoff of solutions to static data structure problems.

1 Introduction

In the cell probe model (e.g., [1, 3, 4, 6, 7, 9, 18–21]), a boolean static data structure problem is given by a map $f : \{0,1\}^n \times \{0,1\}^m \rightarrow \{0,1\}$, where $\{0,1\}^n$ is a set of possible data to be stored, $\{0,1\}^m$ is a set of possible queries and $f(x,y)$ is the answer to question y about data x. For natural problems, we have $m \ll n$: the question we pose to the database is much shorter than the database itself. Examples of natural data structuring problems include:

Substring Search: Given a string x in $\{0,1\}^n$ we want to store it in a data structure so that given a query string y of length m, we can tell whether y is a substring of x by inspecting the data structure. This problem is modeled by the function f defined by $f(x,y) = 1$ iff y is a substring of x.

Prefix Sum: Given a bit vector $x \in \{0,1\}^n$, store it in a data structure so that queries "What is $(\sum_{i=1}^{k} x_i) \bmod 2$?" can be answered. This problem is modeled by the function f defined by $f(x,y) = (\sum_{i=1}^{v_y} x_i) \bmod 2$ where y is the binary representation of the integer v_y.

For Substring Search, both the data to be stored and the query are bit strings, as our framework requires. The only reason for this requirement is that to make our discussion about current lower bound techniques and their limitations clear, we want the parameter n to always refer to the number of bits of the data to be stored, the parameter m to always refer to the number of bits of a query and the output of the query to be a single bit. In general, we don't necessarily expect the data we want to store to be bit strings, but an arbitrary encoding as bit strings may take care of this, as in the following example.

Membership: Given a set S of k binary strings each of length m, store S as a data structure so that given a query $y \in \{0,1\}^m$, we can tell whether $y \in S$. To make this problem fit into the framework above, the function f would be defined by letting $n = \lceil \log_2 \binom{2^m}{k} \rceil$ and fixing, in some arbitrary way, a compact encoding of k-sets as n-bit strings and letting $f(S,y) = 1$ iff $y \in S$.

The framework captures not only the classical "storage and retrieval" static data structure problems but also more general problems of dealing with preprocessed information, such as the classical algebraic problem of polynomial evaluation with preprocessing of coefficients ([15, pp. 470-479], see also [19]):

Polynomial Evaluation: Store $g \in \mathbf{F}[x]$, $|\mathbf{F}| = 2^k$, g of degree $\leq d$ as a memory image so that queries "What is $g(x)$?" can be answered for any $x \in \mathbf{F}$. This problem is non-boolean, but can be modeled as a boolean problem by letting $n = (d+1)k$, $m = k + \log k$, fixing an arbitrary compact encoding of polynomials and field elements as bit strings and letting $f(g, x \cdot y) = v_y$'th bit of $g(x)$, where y is the binary notation of v_y and $\cdot$ denotes concatenation.

In the cell probe model with word size 1 (the bit probe model), a *solution* with space bound s and time bound t to a problem f is given by a storage scheme $\phi : \{0,1\}^n \to \{0,1\}^s$, and a query algorithm q so that $q(\phi(x), y) = f(x, y)$. The time t of the query algorithm is its bit probe complexity, i.e., the worst case number of bits it reads in $\phi(x)$.

Every problem possesses two trivial solutions: The solution of explicitly storing the answer to every query (this solution has space $s = 2^m$ and time $t = 1$) and the solution of storing the data verbatim and reading the entire data when answering queries (this solution has space $s = n$ and time $t = n$, as we only charge for reading bits in $\phi(x)$, not for computation). The study of cell probe complexity concerns itself with the tradeoff between s and t that may be obtained by solutions somewhere between the two extremes defined by the trivial solutions. Such solutions may be quite non-trivial and depend strongly on the problem considered. A *polynomial* solution satisfies $s = n^{O(1)}$ and $t = m^{O(1)}$. For instance, perfect hashing schemes form solutions to Membership with $s = O(n)$ and $t = O(m)$ [11] and even $s = n + o(n)$ and $t = O(m)$ [5, 25]. Substring Search also admits an $s = O(n)$, $t = O(m)$ solution [12] and very recently a solution with $s = n + o(n)$ and $t = m^{O(1)}$ was constructed [13] but no solution with $s = n + o(n)$ and $t = O(m)$ is known. For a problem such as Polynomial Evaluation (and many natural data structure problems, such as partial match type problems [3, 4, 7]), we know of no solution with $s = n^{O(1)}, t = m^{O(1)}$. Thus, a main concern is to prove that such solutions do not exist.

For $s = O(n)$, lower bounds of the form $t = \Omega(m)$ may be obtained for explicit and natural problems by simple counting arguments [6]. For $s = n^{O(1)}$, we can do almost as good: Lower bounds of the form $t = \Omega(m/\log n)$ can be obtained using communication complexity [20]. But no very good (i.e., $\omega(m)$) lower bounds are known on t for *any* explicit problem f for the case of $s = O(n)$ or $s = n^{O(1)}$ even though counting arguments prove the existence of (non-explicit) problems f with lower bounds of the form $t = \Omega(n)$, even for $m \approx (\log n)^2$ [18]. Thus, it is consistent with our current knowledge that solutions with $s = O(n)$ and $t = O(m)$ exist for *all* explicit (e.g., all exponential time computable) problems, though it is certainly a generally believed conjecture that this is not the case!

Given our lack of tools strong enough to show statements such as $s = O(n) \Rightarrow t = \omega(m)$ for explicit problems, it seems appropriate to lower our ambitions slightly and try to show such lower bounds for t for *any* non-trivial value of s. Achieving such goals is well in line with the current trend in the theoretical as well as practical studies of data structures (e.g., [17, 5, 25, 13]) of focusing on *succinct* data structures where $s = n + r$ for some *redundancy* $r \ll n$, i.e., on structures whose space requirement is close to the information theoretic minimum. Restricting our attention to such succinct structures by no means trivializes obtaining the lower bounds we want to show. For instance, it is open (and remains open, also after this work) whether a solution with $r = 0$ and $t = O(m)$ exists for the Membership problem. However, in this paper we show that for certain explicit (polynomial computable) problems it *is* possible to show lower bounds of the form $t = \omega(m)$ and even $t = \Omega(n)$ for structures with a sufficiently strong upper bound on r:

Theorem 1. *Let k, d be integers larger than 0 so that $d < 2^k/3$. Let $\mathbf{F} = \mathrm{GF}(2^k)$ and let $n = (d+1)k$. Let a storage scheme $\phi : \{f | f \in \mathbf{F}[x], \mathrm{degree}(f) \leq d\} \to \{0,1\}^{n+r}$ and associated query scheme for "What is $f(x)$?", $x \in \mathbf{F}$ with bit probe complexity t be given. Then, $(r+1)t \geq n/3$.*

In particular, for very small redundancies, we get an almost optimal lower bound stating that the query algorithm has to inspect almost the entire data structure. The theorem is for the (more natural) non-boolean version of the polynomial evaluation problem. A lower bound of $(r+1)t \geq n/3k$ for the boolean version of polynomial evaluation we defined previously immediately follows.

The proof of Theorem 1 (presented in Section 2) is based on the fact that the problem of polynomial evaluation hides an error correcting code: The strings of query answers for each possible data (i.e., each polynomial) form the Reed-Solomon code. We can generalize Theorem 1 to any problem hiding an error correcting code in a similar way (see Theorems 4 and 5 in Section 2). However, not many natural data structuring problems contain an error correcting code in this way. In Section 2, we introduce a parameter of data structuring problems called *balance* and, using the sunflower lemma of Erdős and Rado show that for problems having constant balance, we get a lower bound of the form $t(r+1)^2 \geq \Omega(n)$ (Theorem 6). A problem hiding a good error correcting code in the way described above has constant balance, but the converse statement is not necessarily true. Hence Theorem 6 has the potential to prove lower bounds on a wider range of problems than Theorems 4 and 5, though we do not have any natural data structuring problems as examples of this at the moment.

The results above are based on combinatorial properties of a coding theoretic flavor of the problems f to be solved. We don't know how to prove similar lower bounds for natural storage and retrieval problems such as Substring Search. However, we get a natural restriction of the cell probe model by looking at the case of *systematic* or *index* structures. These are storage schemes ϕ satisfying $\phi(x) = x \cdot \phi^*(x)$ for some map ϕ^*, i.e, we require that the original data is kept "verbatim" in the data structure. We refer to $\phi^*(x)$ as the *index part* of $\phi(x)$. The restriction only makes sense if there is a canonical way to interpret the data to be stored as a bit-string. It is practically motivated: The data to be encoded may be in read-only memory or belong to someone else or it may be necessary to keep it around for reasons unrelated to answering the queries defined by f. For more discussion, see, e.g. Manber and Wu [17]. In the systematic model, we prove a tight lower bound for Prefix Sum (in fact, we show that the lower bound is implicit in work of Nisan, Rudich and Saks [24]) and a lower bound for Substring Search.

Theorem 2. *$\Theta(n/(r+1))$ bit probes are necessary and sufficient for answering queries in a systematic structure for Prefix Sum with r bit redundancy.*

Theorem 3. *Consider Substring Search with parameters n, m so that $2\log_2 n + 5 \leq m \leq 5\log_2 n$. For any systematic scheme solving it with redundancy r and bit probe complexity t, we have $(r+1)t \geq \frac{1}{800}n/\log n$.*

Both proofs are presented in Section 3. We are aware of one paper previous to this one where lower bounds of the form $t = \omega(m)$ were established for succinct, systematic data structures: Demaine and Lopez-Ortiz [8] show such a lower bound for a variation of the Substring Search problem. In their variation, a query does not just return a boolean value but an index of an occurrence of the substring if it does indeed occur in the string. For this variation, they prove the following lower bound for a value of m which is $\Theta(\log n)$ as in our bound: $t = o(m^2/\log m) \Rightarrow (r+1)t = \Omega(n \log n)$. Thus, they give a lower bound on the query time even with *linear* redundancy which our method cannot. On the other hand, their method cannot give lower bounds on the query time better than $\Omega(m^2/\log m)$ even for very small redundancies which our method can. Furthermore, our lower bound applies to the boolean version of the problem.

2 Lower bounds for non-systematic structures

Proof of Theorem 1. Let a storage scheme ϕ with redundancy r and an associated query scheme with bit probe complexity t be given. Let $s = n + r$. Assume to the contrary that the scheme satisfies $(r+1)t < n/3$. As $r \geq 0$ in any valid scheme, we have $t < n/3$. We make a randomized construction of another storage scheme ϕ' by randomly removing $r+1$ bits of the data structures of storage scheme ϕ. That is, we pick $S \subset \{1, .., n+r\}$ of size $r+1$ at random and let $\phi'(x) = \phi(x)$ with bits in positions $i \in S$ removed. Thus, $\phi'(x) \in \{0,1\}^{n-1}$. We make an associated query scheme for ϕ' by simulating the query scheme for ϕ, but whenever a bit has to be read that is no longer there, we immediately answer "Don't know". Clearly, if we use our new storage scheme ϕ' and the associated query scheme, we will on every query, either get the right answer or the answer "Don't know". Now fix a polynomial f and a query x and let us look at the probability that the randomized construction gives us the answer "Don't know" on this particular data/query-pair. The probability is equal to the probability that the random set S intersects the fixed set T of bits that are inspected on query x in structure $\phi(f)$ according to the old scheme. As $|S| = r+1$ and $|T| \leq t$, the probability of no intersection can be bounded as $\Pr[S \cap T = \emptyset] \geq (\frac{s-t}{s})\,(\frac{s-1-t}{s-1}) \ldots (\frac{s-(r+1)+1-t}{s-(r+1)+1})$ $\geq (1 - \frac{t}{n})^{r+1} \geq 1 - \frac{(r+1)t}{n} > 2/3$. This means that if we fix f and count the number of answers that are not "Don't know" among all answers to "What is $f(x)$?", $x \in \mathbf{F}$, the expected number of such valid answers is $> 2|\mathbf{F}|/3$, and the expected number of "Don't know" answers is $< |\mathbf{F}|/3$. Thus, for fixed f, the probability that the number of valid answers for this f is $< |\mathbf{F}|/3$ is $< 1/2$. Define f to be "good" for a particular choice of S if the number of valid answers for f is at least $|\mathbf{F}|/3$. Thus, for random S, the probability that a particular fixed f is good is $> 1/2$, by the above calculation, so if we count among all 2^n possible f's the number of good f's, the expectation of this number is $> 2^n/2$. Thus, we can fix a value of S so that the number of good f's is $> 2^n/2$. Let the set of good f's relative to this choice of S be called G. We now argue that the map $\phi' : G \to \{0,1\}^{n-1}$ is a 1-1 map: Given the value $\phi'(f)$ for a particular $f \in G$, we can run the query algorithm for $f(x)$ for all $x \in \mathbf{F}$ and retrieve a valid answer in at least $|\mathbf{F}|/3$ cases - in the other cases we get the answer "Don't know". Since the degree of f is less than $|\mathbf{F}|/3$, the information we retrieve is sufficient to reconstruct f. Thus, we have constructed a 1-1 map from G with $|G| > 2^n/2$ to the set $\{0,1\}^{n-1}$ which has size $2^n/2$. This violates the pigeonhole principle, and we conclude that our assumption $(r+1)t < n/3$ was in fact wrong. This completes the proof of Theorem 1.

Theorem 1 can be generalized to any problem based on some error correcting code. Consider an arbitrary boolean static data structure problem, given by a map $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$. Let $N = 2^n$, and $M = 2^m$. Then the problem can be represented by an $N \times M$ Boolean matrix A_f, with the entry at the row indexed by x and the column indexed by y being equal to $f(x, y)$.

Theorem 4. *Let A_f be the N by M $(N = 2^n)$ matrix of a data structure problem such that the rows of A_f have pairwise distance at least δM. If the problem can be solved with redundancy r and query time t, then $t(r+1) \geq \delta n/2$.*

The argument can also be extended to problems where the minimum distance may not be large, but instead we require that within any ball of radius ρM there are at most L codewords (i.e., codes with certain *list decoding* properties). In fact, the even weaker property of having only few codewords in every *subcube* of dimension ρM is sufficient for our purposes. (Note that this property corresponds to the problem of list decoding from erasures, rather than from errors.)

Let $\alpha_{i_1}, \ldots, \alpha_{i_{M-d}}$ be an arbitrary 0/1 assignment to $M - d$ coordinates. The set $S \subseteq \{0,1\}^M$ of size $|S| = 2^d$ formed by all possible vectors from $\{0,1\}^M$ agreeing with $\alpha_{i_1}, \ldots, \alpha_{i_{M-d}}$ and arbitrary in the remaining coordinates is called a *subcube of dimension d*.

Theorem 5. *Let A_f be the N by M $(N = 2^n)$ matrix of a data structure problem such that within any subcube of dimension ρM there are at most L row vectors from A_f. If the problem can be solved with redundancy r and query time t, then $t(r + 1 + \log L) \geq \rho(n - \log L)/2$.*

The proofs of Theorems 4 and 5 are very similar to the proof of Theorem 1 and appear in the full version of this paper.

We next give a general lower bound for any problem whose matrix satisfies certain conditions. Informally, we require that the submatrix formed by any small subset of rows contains a balanced column.

Definition 1. *Let A be a matrix with 0/1 entries. We say that A has balance at least λ for parameter k, if for any k rows of the matrix A there exists a column that contains at least λk 0-s and at least λk 1-s among the entries of the given k rows.*

Lemma 1. *Given a code with N words in $\{0,1\}^l$, let A be the N by l matrix formed by the words as rows. If the minimum distance of the code is δl, then A has balance at least $\delta/8$ for every $1 < k \leq N$.*

Proof. Look at the k by l table formed by k rows of A. Let $\gamma = \delta/8$. Suppose that each column in the table has either $< \gamma k$ 0-s or $< \gamma k$ 1-s. Let a be the number of mostly 1 columns and b be the number of mostly 0 columns. Then $< k/2$ rows have $> 2\gamma a$ 0-s on the mostly 1 part. Restrict the table to the other $k' > k/2$ rows. In this table, the b mostly 0 columns still have $< 2\gamma k'$ 1-s. So, $< k'/2$ rows have $> 4\gamma b$ 1-s on the mostly 0 part. Thus, $> k/4$ rows have both $< 2\gamma a$ 0-s on the mostly 1 part and $< 4\gamma b$ 1's on the mostly 0 part, respectively. The distance of any two of these rows is $< 4\gamma a + 8\gamma b < \delta l$, which is a contradiction.

The proof of Lemma 1 also extends to codes where the minimum distance may not be large, but instead we require that within any ball of certain radius there are not too many words, i.e., to problems satisfying the condition of Theorem 5. We can, however, construct codes that satisfy the property of having large balance for every k, *without* the property of having few codewords in every Hamming ball of a given radius, and even without the weaker property of having few codewords in every subcube of a given dimension. Consider the following example of such construction. Let ρ be any constant, and L any integer, such that $\rho + \frac{1}{L} < 1/20$. We will construct a set of words in $\{0,1\}^M$ with at least L words in some subcube of dimension ρM, such that for any set of rows of the corresponding matrix there is a column with balance $> \rho + \frac{1}{L}$. Start with any family that has balance at least $5(\rho + \frac{1}{L})$. (We know the existence of such families, from the existence of good error correcting codes.) Add L words to this family as follows. Take a code of L words on $c \log L$ coordinates for some constant c, with relative minimum distance $1/4$. (Such code exists for some constant c.) Let the first $c \log L$ coordinates of the extra L words to be words from this code of size L, and let the L words be identical in the remaining $M - c \log L$ coordinates. Unless L is huge (compared to M), we have $c \log L < \rho M$, thus we have L words in a subcube of dimension ρM. It is not hard to see that the corresponding matrix has balance at least $\rho + \frac{1}{L}$ for any k. Thus, the following theorem has the potential of giving lower bounds for a wider range of problems than the theorems of Section 2. Consider an arbitrary boolean static data structure problem, given by a map $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$.

Theorem 6. *Let A_f be the N by M ($N = 2^n$, $M = 2^m$) matrix of f. If A_f has balance at least λ for every $1 < k \leq \log N$. and the problem defined by f can be solved with redundancy r and query time t, then $t(r+1)^2 \geq \lambda n$.*

Proof. A solution to the data structure problem is given by a representation $\phi : \{0,1\}^n \to \{0,1\}^s$ and a query algorithm. We consider a matrix B of size

$N \times s$, such that the row of B indexed by x is the vector $\phi(x)$. We use the following standard observation.

Observation 1. *Given a set $\mathcal{C}$ of $N = 2^{s-r}$ vectors in $\{0,1\}^s$, for every $0 \leq w \leq s$ there is a vector $v \in \{0,1\}^s$, such that there are at least $\binom{s}{w}/2^r$ vectors in $\mathcal{C}$ at distance w from v.*

Proof. Let $\chi(u,v) = 1$ if u and v differ in w coordinates, and $\chi(u,v) = 0$ otherwise. We have $\sum_{u \in \mathcal{C}} \sum_{v \in \{0,1\}^s} \chi(u,v) = |\mathcal{C}| \binom{s}{w}$. On the other hand, $\sum_{v \in \{0,1\}^s} \sum_{u \in \mathcal{C}} \chi(u,v) \leq 2^s \max_{v \in \{0,1\}^s} |\mathcal{C}_{v,w}|$, where $\mathcal{C}_{v,w} = \{z \in \mathcal{C} |\ z$ and v differ in w coordinates $\}$. This completes the proof of Observation 1.

Let $w = r+1$ (note that $r+1 \geq 1$), and let $v \in \{0,1\}^s$, guaranteed to exist by the observation, such that there are at least $\binom{s}{r+1}/2^r$ rows of B at distance $r+1$ from v. Let B_v be the matrix obtained from B by adding v to each row of B (taking bitwise XOR). With each vector $u \in \{0,1\}^s$ we associate a set $U \subseteq [s]$, such that $i \in [s]$ belongs to U if and only if the i-th entry of u is 1. Then the matrix B_v specifies a family $\mathcal{B}$ of N sets, such that at least $\binom{s}{r+1}/2^r$ members of $\mathcal{B}$ have cardinality $r+1$.

A family of k sets $S_1, \ldots, S_k$ is called a *sunflower* with k petals and core T, if $S_i \cap S_j = T$ for all $i \neq j$. We also require that the sets $S_i \setminus T$ are nonempty.

Lemma 2 (Erdős and Rado, [10]). *Let $\mathcal{F}$ be a family of sets each with cardinality w. If $|\mathcal{F}| > w!(k-1)^w$, then $\mathcal{F}$ contains a sunflower with k petals.*

Since $\binom{s}{r+1}/2^r > (r+1)!(s/(r+1)^2)^{r+1}$, Lemma 2 implies that $\mathcal{B}$ contains a sunflower with $k = s/(r+1)^2$ petals. Let $S_1, \ldots, S_k$ be the sets of the sunflower, and let T be its core. Then, the sets $S_i \triangle T$ are pairwise disjoint. ($S_i \triangle T$ denotes the symmetric difference of the sets S_i and T.) Let z and $u_1, \ldots, u_k$ be the vectors obtained by adding the vector v to the characteristic vectors of the set T and $S_1, \ldots, S_k$, respectively. Then the vectors $u_1, \ldots, u_k$ are rows of the matrix B, and they have the property that the vectors $z \oplus u_1, \ldots, z \oplus u_k$ have no common 1's, since the set $S_i \triangle T$ is exactly the set of coordinates where the vectors z and u_i differ from each other. Let $x_1, \ldots, x_k$ be the data such that $u_i = \phi(x_i), i = 1, \ldots, k$. Consider now the k rows of A_f indexed by $x_1, \ldots, x_k$. By our assumption on A_f, there is a question y, such that at least λk of the answers $f(x_i, y)$ are 0, and at least λk of the answers $f(x_i, y)$ are 1. We think of the query algorithm as a decision tree, and show that it has large depth. In particular, we show that the path consistent with the vector z has to be at least λk long. (Note that the vector z may not be a row of the matrix B. However, we can assume that the decision tree has been trimmed, so that there are no long paths that can be cut off without affecting the correctness of the algorithm. This implies that there is at least one path corresponding to a vector $\phi(x)$ that the algorithm may actually have to follow, and is at least λk long.) Assume that the query algorithm reads at most $t < \lambda k$ bits on any input when trying to answer the question y, and assume that the bits read are consistent with the vector z. Since the sets of coordinates where z differs from u_i for $i = 1, \ldots, k$ are pairwise disjoint, after asking at most t questions, the algorithm can rule out at most t of the data $x_1, \ldots, x_k$, and the remaining $k - t$ are still possible. If $t < \lambda k$, then among the data that are still not ruled out, both the answer 0 and the answer 1 is possible, and the algorithm cannot determine the answer to the given question y. This completes the proof of Theorem 6.

It is not hard to find examples of matrices with large balance for $k \leq \log N$, if we are not worried about the number of rows N being large enough compared to the number of columns M. We should mention that there are well known constructions (e.g. [2, 14, 22, 23, 26]) for the much stronger property requiring that all possible 2^k patterns appear in the submatrix formed by arbitrary k rows. However, in such examples, $N \leq M$ or $2^k \leq M$ must trivially hold. Error correcting codes provide examples where N can be very large compared to M. Let $n(k, \lambda, M)$ denote the largest possible number n, such that 2^n by M 0/1 matrices exist with balance at least λ for k. Lower bounds on the largest achievable rate of error-correcting codes or list decodable codes provide lower bounds on $n(k, \lambda, M)$. For example, the Gilbert-Varshamov bound (see e.g. [16]) together with Lemma 1 implies $n(k, \lambda, M) \geq (1 - H(8\lambda))M$, for every $k > 1$. Note that while error correcting codes give large balance for every $k > 1$, for our purposes matrices that have large balance for only certain values of k may already be useful. It would be interesting to know if $n(k, \lambda, M)$ can be significantly larger (for certain values of k) than what is achievable by error-correcting or list decodable codes. If this is the case, then our techniques might help to achieve lower bounds for the Membership problem.

3 Lower bounds for systematic structures

Proof of Theorem 2. Upper bound: For $r = 0$, the upper bound is obvious. For $r \geq 1$, divide the input vector into r equal sized blocks and let y_i be the parity of the i'th block. Now store for each $j = 1, ..r$, the parity of $y_1, y_2, \ldots, y_j$. Given a prefix sum query, it can be answered by reading a non-systematic bit, that gives the parity of a collection of blocks and XORing it with a number of individual input bits, all found in a single block of size n/r. The bit probe complexity is $O(n/r)$.
Lower bound: Let a scheme of redundancy r be given and suppose the queries can be answered with t bit probes, i.e., we can find $x_1 \oplus \cdots \oplus x_j$ using a decision tree of depth t over the input bits and the index bits. Split the input into $r + 1$ blocks of about equal length, each block containing at least $\lfloor \frac{n}{r+1} \rfloor$ bits. It is possible to determine the parity of one of the blocks by a decision tree of depth $2t$ over the input bits and the index bits. We now apply a theorem of Nisan, Rudich and Saks [24]: Given $l + 1$ instances of computing parity of k bits, with l *help bits* (which can be arbitrary functions of the $(l + 1)k$ input bits), given for free. At least one of the $l + 1$ parity functions has decision tree complexity $\geq k$. We immediately get the desired bound.

Proof of Theorem 3. Since we must have $r \geq 0$ and $t \geq 1$ in a valid scheme, we can assume that $1 \leq t \leq \frac{n}{800 \log n}$ otherwise there is nothing to prove. We need to prove a claim about a certain two-player game. Let $b \geq a \geq 40$ be integers and assume b is even. The game is played with b boxes labeled $0, \ldots, b-1$ and a slips of papers, labeled $0, \ldots, a-1$. Player I colors each slip of paper either red or blue and puts each slip of paper in a box (with no two slips going into one box) without Player II watching. Now Player II can open at most $b/2$ boxes using any adaptive strategy and based on this must make a guess about the color of every slip of paper. Player II wins the game if he correctly announces the color of every slip of paper. Suppose Player I adopts the strategy of coloring each slip of paper uniformly and independently at random and putting them at random into a boxes chosen uniformly at random. We claim that no matter which strategy Player II adopts, the probability that Player II wins the game is at most $2^{-a/20}$. To prove the claim, note that when Player I is playing uniformly at random in the way described, by symmetry the adaptiveness of Player II is useless and the optimal strategy for Player II is to open boxes $1, 2, \ldots, b/2$, announce the colors of the slips of papers found and make an arbitrary guess for the rest. The probability that he finds more than $\frac{9}{10}a$ slips of papers is $\sum_{j > \frac{9}{10}a}^{a} \frac{\binom{b/2}{j}\binom{b/2}{a-j}}{\binom{b}{a}} = \sum_{i=0}^{\lfloor \frac{1}{10}a \rfloor} \frac{\binom{b/2}{i}\binom{b/2}{a-i}}{\binom{b}{a}}$. Since $a \leq b$, for $i \leq \frac{1}{10}a$ we have $\frac{b}{2(b-i)} \leq 5/9$. Then, $\frac{\binom{b/2}{i}\binom{b/2}{a-i}}{\binom{b}{a}} \leq \binom{a}{i}(1/2)^i(\frac{b}{2(b-i)})^{a-i} \leq \binom{a}{i}(5/9)^a$ and $\sum_{i=0}^{\lfloor \frac{1}{10}a \rfloor} \frac{\binom{b/2}{i}\binom{b/2}{a-i}}{\binom{b}{a}} \leq (5/9)^a \sum_{i=0}^{\lfloor \frac{1}{10}a \rfloor} \binom{a}{i} \leq (5/9)^a 2^{H(1/10)a} \leq 2^{(H(1/10) - \log_2(3/2))a}$ $\leq 2^{-0.115a}$. The probability that he guesses the colors of all remaining slips correct, given that at least $a/10$ was not found is at most $2^{-a/10}$. Thus, the probability that Player II correctly guesses the color of every slip of paper is bounded by $2^{-0.115a} + 2^{-a/10} \leq 2^{-a/20}$, as $a \geq 40$. This completes the proof of the claim.

We show that a good scheme for Substring Search leads to a good strategy for Player II in the game. So given a scheme with parameters n, m, r, t, we let $a = \lfloor \frac{n}{4tm} \rfloor$ and $b = 4ta$. Since $t \leq n/(800 \log n)$ and $m \leq 5 \log n$, we have $a \geq 40$. We consider a string of length n as consisting of b concatenated chunks of length m, padded with 0's to make the total length n (note that $bm = 4tam \leq n$). We can now let such a string encode a move of Player I (i.e. a coloring of slips of papers and a distribution of them into boxes) as follows: The content of Box i is encoded in chunk number i. If the box is empty, we make the chunk 000000..000. If the box contains paper slip number j, colored blue, we make the chunk $001 j_1 1 j_2 1 j_3 1 \ldots 1 j_k 0$, padded with zeros to make the total length m, where $j_1 \ldots j_k$ is the binary representation of j with $\lceil \log a \rceil$ binary digits (note that $3 + 2\lceil \log a \rceil \leq 2 \log n + 5 \leq m$). Similarly, if the box contains paper slip number j, colored red, we make the chunk $001 j_1 1 j_2 1 j_3 1 \ldots 1 j_k 1$, padded with zeros. Now consider the set X of strings encoding all legal moves of player I. Each element x of X has some systematic data structure $\phi(x) = x \cdot \phi^*(x)$ where $\phi^*(x) \in \{0,1\}^r$. Pick the most likely setting z of $\phi^*(x)$ of these among elements of X, i.e., if we take a random element x of X, the probability that $\phi^*(x) = z$ is at least 2^{-r}. We now make a strategy for Player II in the game. Player II will pretend to have access to a Substring Search data structure which he will hope encodes the move of Player I. The index part of this data structure will be the string z which is fixed and independent of the move of Player I and hence can be hardwired into the protocol of Player II. Player II shall simulate certain query operations on the pretend data structure. However, he has only access to the index part of the structure (i.e., z). Thus, whenever he needs to read a bit of the non-index bits, he shall open the box corresponding to the chunk of the bit from which he can deduce the bit (assuming that the entire data structure really does encode the move of Player I). In this way, Player II simulates performing query operations "Is $001j_1 1j_2 1j_3 1...1j_k 0$ a substring?" and "Is $001j_1 1j_2 1j_3 1...1j_k 1$ a substring?" with $j = j_1 j_2 \ldots j_k$ being the binary representations of all $y \in \{0, \ldots, a-1\}$, i.e., $2a$ query operations. From the answers to the queries, he gets a coloring of the slips of papers. All answers are correct for those cases where his index part was the correct one, i.e., for those case where $z = \phi^*(x)$ where x is an encoding of the move of Player I, i.e., with probability at least 2^{-r}. Thus, since the total number of boxes opened is at most $t2a \leq b/2$, we have by the claim that $r \geq a/20$, i.e., $20r \geq \lfloor n/4tm \rfloor$, and, since r is an integer and $m \leq 5 \log n$ we have $(r+1)t \geq \frac{1}{400} n/\log n$. This completes the proof of Theorem 3.

We could potentially get a better lower bound by considering a more complicated game taking into account the fact that the different query operations do not communicate. Again we have b boxes labeled $0, \ldots, b-1$ and a slips of paper, labeled $0, \ldots, a-1$. The modified game is played between Player I and a *team* consisting of Player II_0, II_1, …, II_{a-1}. Again, Player I colors each slip of paper either red or blue and puts each slip of paper in a box without Players II_0, II_1, …, II_{a-1} watching. Now Player II_i can look in at most $b/2$ boxes using any adaptive strategy and based on this must make a guess about the color of the slip labeled i. This is done by each player on the team individually without communication or observation between them. The team wins if *every* player in the team correctly announces the color of "his" slip. About this game we can state the following hypothesis.

Hypothesis Let $b \geq 2a$. Suppose Player I adopts the strategy of coloring each slip of paper uniformly at random and independently putting them at random into a boxes chosen uniformly at random. Then no matter which strategy the team adopts, the probability that they win is at most $2^{-\Omega(a)}$.

The intuition for the validity of the hypothesis is the fact that the players of the team are unable to communicate and each will find his own slip of paper with probability $\leq \frac{1}{2}$. If the hypothesis can be verified it will lead to a tradeoff for Substring Search of the form $t = o(n/\log n) \Rightarrow s = \Omega(n/\log n)$. However, Sven Skyum (personal communication) has pointed out that if the hypothesis is true, the parameters under which it is true are somewhat fragile: If $b = a$, the team can win the game with probability bounded from below by a constant (roughly 0.3) for arbitrary large values of a. The catch is that even though each player will find his own slip of paper with probability only $\frac{1}{2}$, one can make these events highly dependent (despite the fact that the players do not communicate). We leave finding Skyum's protocol as an exercise to the reader.

4 Open problems

It is interesting that all our best bounds, both in the non-systematic and in the systematic case, are of the form "$(r+1)t$ must be linear or almost linear in n." We don't see any inherent reason for this and in general do not expect the lower bounds obtained to be tight. Thus, it would be nice to to prove a lower bound of, say, the form, $t < n/\text{polylog}\, n \Rightarrow r > n/\text{polylog}\, n$ for Polynomial Evaluation in the non-systematic case or Substring Search in the systematic case. For the latter result, it would be sufficient to verify the hypothesis about the game defined above. It is also interesting to note that our lower bound for Substring Search and the lower bound of Demaine and Lopez-Ortiz are incomparable. Can the two techniques be combined to yield a better lower bound? We have only been able to prove lower bounds in the non-systematic case for problems satisfying certain coding theoretic properties. It would be very nice to extend the non-systematic lower bounds to more natural search and retrieval problems, such as Substring Search. A prime example of a problem for which we would like better bounds is Membership as defined in the introduction. As the data to be stored has no canonical representation as a bitstring, it only makes sense to consider this problem in the non-systematic model. The lower bound $r = O(n) \Rightarrow t = \Omega(m)$ was shown by Buhrman et al [6]. On the other hand, a variety of low-redundancy dictionaries with $r = o(n)$ and $t = O(m)$ has been constructed [5, 25]. We conjecture that any solution for membership with $t = O(m)$ must have *some* redundancy, i.e., that $t = O(m) \Rightarrow r \geq 1$. It would be very nice to establish this. The main open problem of cell probe complexity remains: Show, for some explicit problem, a tradeoff of the form $r = O(n) \Rightarrow t = \omega(m)$. Clearly, for such tradeoffs the distinction between systematic and non-systematic structures is inconsequential.

Acknowledgements Anna Gál is supported in part by NSF CAREER Award CCR-9874862 and an Alfred P. Sloan Research Fellowship. Peter Bro Miltersen is supported by BRICS, Basic Research in Computer Science, a centre of the Danish National Research Foundation.

References

1. M. Ajtai. A lower bound for finding predecessors in Yao's cell probe model. *Combinatorica*, 8:235–247, 1988.
2. N. Alon, O. Goldreich, J. Håstad, R. Peralta: Simple constructions of almost k-wise independent random variables. *Random Structures and Algorithms* 3 (1992), 289–304.
3. O. Barkol and Y. Rabani, Tighter bounds for nearest neighbor search and related problems in the cell probe model. In *Proc. 32th Annual ACM Symposium on Theory of Computing (STOC'00)*, pages 388–396.
4. A. Borodin, R. Ostrovsky, Y. Rabani, Lower bounds for high dimensional nearest neighbor search and related problems. In *Proc. 31th Annual ACM Symposium on Theory of Computing (STOC'99)*, pages 312–321.
5. A. Brodnik and J.I. Munro. Membership in constant time and almost-minimum space. *SIAM Journal on Computing*, 28:1627–1640, 1999.

6. H. Buhrman, P.B. Miltersen, J. Radhakrishnan, S. Venkatesh. Are bitvectors optimal? In *Proc. 32th Annual ACM Symposium on Theory of Computing (STOC'00)*, pages 449–458.
7. A. Chakrabarti, B. Chazelle, B. Gum, and A. Lvov. A lower bound on the complexity of approximate nearest-neighbor searching on the Hamming Cube. In *Proc. 31th Annual ACM Symposium on Theory of Computing (STOC'99)*, pages 305–311.
8. E.D. Demaine and A. Lopez-Ortiz. A Linear Lower Bound on Index Size for Text Retrieval. In *Proc. 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'01)*, pages 289–294.
9. P. Elias and R. A. Flower. The complexity of some simple retrieval problems. *Journal of the Association for Computing Machinery*, 22:367-379, 1975.
10. P. Erdős and R. Rado. Intersection theorems for systems of sets. *Journal of London Mathematical Society* 35 (1960), pages 85-90.
11. M. L. Fredman, J. Komlós, and E. Szemerédi. Storing a sparse table with $O(1)$ worst case access time. *Journal of the Association for Computing Machinery*, 31:538–544, 1984.
12. R. Grossi, J.S. Vitter. Compressed suffix arrays and suffix trees with applications to text indexing and string matching. In *Proc. 32th Annual ACM Symp. on Theory of Computing (STOC'00)*, pages 397–406.
13. R. Grossi, A. Gupta, and J.S. Vitter. High-Order Entropy-Compressed Text Indexes. In *Proc. 14th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA'03)*, pages 841–850.
14. D. J. Kleitman and J. Spencer: Families of k-independent sets. *Discrete Math.*6 (1973), pp. 255-262.
15. D.E. Knuth, *The Art of Computer Programming, Vol. II: Seminumerical Algorithms* (Addison-Wesley, Reading, MA, 2nd ed., 1980).
16. F. J. MacWilliams and N. J. A. Sloane. The theory of error correcting codes. Elsevier/North-Holland, Amsterdam, 1981.
17. U. Manber, S. Wu. GLIMPSE - A Tool to Search Through Entire Filesystems. White Paper. Available at `http://glimpse.cs.arizona.edu/`.
18. P.B. Miltersen. The bitprobe complexity measure revisited. In *10th Annual Symposium on Theoretical Aspects of Computer Science (STACS'93)*, pages 662–671, 1993.
19. P.B. Miltersen, On the cell probe complexity of polynomial evaluation, *Theoretical Computer Science*, 143:167–174, 1995.
20. P.B. Miltersen, N. Nisan, S. Safra, and A. Wigderson: On data structures and asymmetric communication complexity, *Journal of Computer and System Sciences*, 57:37–49, 1998.
21. M. Minsky and S. Papert. *Perceptrons.* MIT Press, Cambridge, Mass., 1969.
22. J. Naor and M. Naor: Small-bias probability spaces: efficient constructions and applications. *SIAM J. Comput.*, Vol. 22, No. 4, (1993), pp. 838-856.
23. M. Naor, L. Schulman, A. Srinivasan: Splitters and near optimal derandomization. In *Proc. of 36th IEEE FOCS*, (1995), pp. 182-191.
24. N. Nisan, S. Rudich, and M. Saks. Products and Help Bits in Decision Trees, *SIAM J. Comput.* 28:1035–1050, 1999.
25. R. Pagh. Low redundancy in static dictionaries with $O(1)$ lookup time. In *International Colloquium on Automata Languages and Programming (ICALP'99), Lecture Notes in Computer Science*, Volume 1644, pages 595–604, 1999.
26. G. Seroussi and N. Bshouty: Vector sets for exhaustive testing of logic circuits. *IEEE Trans. Inform. Theory*, 34 (1988), pp. 513-522.