# Research Report 267

## On the Cell Probe Complexity of Polynomial Evaluation

Peter Bro Miltersen

RR267

We consider the cell probe complexity of the polynomial evaluation problem with preprocessing of coefficients, for polynomials of degree at most $n$ over a finite field $K$. We show that the trivial cell probe algorithm for the problem is optimal if $K$ is sufficiently large compared to $n$. As an application, we give a new proof of the fact that $P \neq ncr\text{-}TIME\ (o(\log n / \log \log n))$.

Department of Computer Science
University of Warwick
Coventry CV4 7AL
United Kingdom

April 1994

# On the cell probe complexity of polynomial evaluation * (note)

Peter Bro Miltersen †

Aarhus University
and
University of Warwick

**Abstract**

We consider the cell probe complexity of the polynomial evaluation problem with preprocessing of coefficients, for polynomials of degree at most $n$ over a finite field $K$. We show that the trivial cell probe algorithm for the problem is optimal if $K$ is sufficiently large compared to $n$. As an application, we give a new proof of the fact that $P \neq incr\text{-}TIME(o(\log n / \log\log n))$.

## 1 Introduction

Let $K$ be a field. We consider the *polynomial evaluation problem* with preprocessing of coefficients. This problem is as follows: Given a polynomial $f(X) \in K[X]$, preprocess it, so that later, for any field element $a \in K$, $f(a)$ can be computed efficiently. It is a classical problem in the theory of algebraic complexity and has been intensively investigated in the model of *arithmetic straight line programs*. In this model, a solution for the polynomials of degree at most $n$ is given by two objects:

- A map $o$ from the set of polynomials of degree at most $n$ into $K^s$, where $s$ is any integer, called the *preprocessing* map. This maps associates with each polynomial $f(X)$ a vector $(y_1, y_2, \ldots, y_s)$, the result of preprocessing $f(X)$.
- An arithmetic straight line program $P$ with inputs $x, y_1, y_2, \ldots, y_s$, i.e. a sequence of instructions

$$\{v_i := u_i \ \circ_i \ w_i\}_{i \in \{1,\ldots,t\}},$$

*This work was supported by a grant from the Danish Natural Science Research Council. It was partially supported by the ESPRIT II Basic Research Actions Program of the European Community under contract No. 7141 (project ALCOM II).

†Correspondence to: Peter Bro Miltersen, University of Warwick, Department of Computer Science, Coventry CV4 7AL, U.K. Email: pbmilter@dcs.warwick.ac.uk or pbmiltersen@daimi.aau.dk

where $u_i, w_i \in K \cup \{x, y_1, y_s, \ldots, y_s, v_1, v_2, \ldots v_{i-1}\}$ and $\circ_i \in \{+, -, \cdot\}$.

The program $P$ defines a function $P : K^{1+s} \to K$ in the natural way (the value of the function being the value of the variable $v_t$). The solution $(o, P)$ is a correct solution for the polynomial evaluation problem if $P(a, o(f(X))) = f(a)$ for all $f(X)$ and $a$. The complexity of the solution is $t$, the number of steps in $P$.

An obvious solution is to let $o$ be defined by

$$o(a_0 + a_1X + a_2X^2 + \ldots + a_nX^n) = (a_0, a_1, \ldots, a_n)$$

i.e. refrain from preprocessing, and let $P$ evaluate the polynomial using *Horner's rule*, i.e.

$$\begin{aligned} v_1 &:= y_n \cdot x \\ v_2 &:= v_1 + y_{n-1} \\ v_3 &:= v_2 \cdot x \\ v_4 &:= v_3 + y_{n-2} \\ &\ldots \\ v_{2n} &:= v_{2n-1} + y_0 \end{aligned}$$

with a complexity of $2n$. This is not optimal for $K = \mathbf{R}$: Pan [11] gives a scheme with complexity $\lfloor \frac{3n}{2} \rfloor + 2$ for $K = \mathbf{R}$. His scheme is almost optimal. Belaga [4] shows that any correct scheme for $K = \mathbf{R}$ has complexity at least $\lfloor \frac{3n}{2} \rfloor + 1$. For a survey of these, and similar, more recent, results, see Knuth [7, pp. 470-479].

In this paper we consider the problem for $K$ being a *finite* field. Since the lower bounds above are proved in the context of algebraic independence theory, there is no way to extend them to this situation. If $K$ is a finite field we might also note that the arithmetic straight line program model seems unreasonable weak, since we in that case can represent the elements of $K$ by small integers and use the full power of a random access machine, e.g. branching, indirect addressing and an extended instruction set, to solve the problem.

This changes the problem somewhat. For instance, in order to get a non-trivial problem we must put a bound on $s$, the number of indices in $o(f(X))$, since we could otherwise, in the case where e.g. $K = \mathbf{Z}_p$, define

$$o(f(X)) = (f(0), f(1), f(2), \ldots, f(p-1))$$

and "compute" $f(a)$ for any value of $a$ with a single table look up, using indirect addressing.

The precise model in which we consider the problem in this paper is the *cell probe* model, which for our purposes can be regarded as a strong, non-uniform version of the random access machine model. Previously, the cell probe model has been studied mainly for *set* problems, such at the problems of storing a set $S \subseteq \{1, \ldots, m\}$, using few (e.g. $O(|S|)$) cells, each containing an element from $\{1, \ldots, m\}$, so that *membership* queries "Is $i \in S$?" [13, 5] or *rank* queries "What is $|S \cap \{1, \ldots, i\}|$?" [2, 1, 9] can be answered efficiently.

In the cell probe model, a solution with size bound $s$ for the polynomial evaluation problem for polynomials of degree at most $n$ is given by the following objects:

- As in the straight line model, a preprocessing map $o$ from the set of polynomials of degree at most $n$ into $K^s$.
- For each $a \in K$ a *decision tree* $T_a$ over $K^s$. This is a rooted tree with each internal node having $|K|$ sons. Each node is labeled with an integer between 1 and $s$. The out-going edges from each node are labeled with elements of $K$, each possible value appearing exactly once. The leaves are labeled with elements from $K$.

Given a vector $y \in K^s$, we can compute a value $T_a(y)$ by the following procedure: We start in the root of $T_a$ and read its label $i$. We proceed to a new node by following the edge with label $y_i$, and read the label of this node, etc. We continue this until reaching a leaf, the value read there is $T_a(y)$. A cell probe algorithm is correct if $T_a(o(f(X)) = f(a)$ for all $f(X)$ and $a$. Its complexity is the depth of the deepest tree.

Note that the cell probe model only makes sense for $K$ a finite field, since if $K$ is infinite, we can let $\phi$ be an injection $K^n \rightarrow K$, giving a complexity of 1.

If $s \geq n+1$, an upper bound on the complexity of the problem is $n+1$, by the algorithm which stores a polynomial as its coefficients and reads them all when evaluating. We are interested in knowing if this is optimal.

We prove the following result.

**Theorem 1** *Let $K$ be a finite field. Any size bound $s$ cell probe algorithm solving the problem for polynomials of degree at most $n$ has complexity at least*

$$\min(n+1, \frac{\log|K| - \log n}{\log s})$$

Thus, if $s$ is reasonably small (e.g. $s = n^{O(1)}$), and $K$ is sufficiently large compared to $n$ ($\log|K| >> n \log n$), the trivial cell probe algorithm is optimal.

We don't have any lower bounds for smaller values of $|K|$, but neither do we know of any scheme beating the trivial upper bound for any value of $|K|$ and $n$ with $n, s = o(|K|)$. We conjecture that the lower bound holds for smaller values of $|K|$ as well, i.e. that polynomial evaluation in general is *access infeasible* [8].

As an application, we consider lower bounds for *dynamic language membership* problems. The class of dynamic language membership problems is a general class of dynamic problems, considered by Miltersen, Subramanian, Vitter and Tamassia [10]. A problem in this class is given by a language $L \subseteq \{0,1\}^*$. We are supposed to implement a data type $L$-MEMBER containing a string $x \in \{0,1\}^*$ with three kinds of operations:

- `init`$(n)$. This operation initializes $x$ to $0^n$.
- `change`$(i,a)$. This operation changes the $i$'th component of $x$ to $a$.
- `query`. This operation returns `true` if $x \in L$, `false` otherwise.

Many naturally occurring problems, for instance dynamic graph problems, can be phrased as dynamic language membership problems. For a time bound $t(n)$,

the complexity class *incr-TIME*$(t(n))$ is the class of languages $L$ for which $L$-MEMBER has an implementation on a random access computer [3], i.e. a unit cost random access machine where each machine word stores an integer, polynomially bounded in $n$, so that `init`$(n)$ can be done in time $n^{O(1)}$ and **change** and **query** can be done in time $t(n)$ (with $n = |x|$).

Because polynomial initialization time is allowed and no restrictions on the amount of memory used by the implementation is made, the definition is robust against reasonable changes in the instruction set of the random access computer, since we can make tables of required instructions during initialization.

Clearly, for any time bound $t(n)$ bounded by a polynomial, *incr-TIME*$(t(n))$ is included in $P$, the class of languages which can be recognized in polynomial time, but it is an open problem if $P =$ *incr-TIME*$(O(\log n/\log\log n))$. It follows from a lower bound on dynamic prefix problems by Fredman and Saks [6], using the *time stamp* method, that $P \neq$ *incr-TIME*$(o(\log n/\log\log n))$. We give a completely different (and somewhat easier) proof of this fact, by giving a lower bound for a polynomial time problem related to polynomial evaluation.

# 2 The proof

The proof, which is not difficult, uses the technique of reduction from communication problems (first used implicitly by Ajtai [1], made explicit by Miltersen [8, 9]), together with standard techniques in communication complexity [12], modified to non-binary protocols.

In the following, $K$ is a fixed finite field with $|K| = k$. Consider the following communication game between two players, Alice and Bob.

- Alice is given $a \in K$.
- Bob is given a polynomial $f(X) \in K[X]$ of degree at most $n$.

The object of the game is to let Alice determine the value of $f(a)$ through communication with Bob. The communication is structured in the following way: Alice chooses her messages from $\{1, \ldots, s\}$, Bob chooses his messages from $\{1, \ldots, k\}$ and the communication is strictly alternating, with Alice sending the first message. The complexity is the worst case number of rounds required in an optimal protocol before Alice is able to give the correct answer.

**Lemma 2** *If there is a cell probe algorithm with size bound $s$ and complexity $t$ for the polynomial evaluation problem, then the complexity of the communication game is at most $t$.*

**Proof** We construct a communication protocol using the cell probe algorithm.

Suppose Alice is given $a$ and Bob is given $f(X)$. Bob computes $o(f(X)) \in K^s$, but does not send anything yet.

Then Alice simulates the decision tree $T_a$ by sending Bob requests for the cells she wants to read in $o(f(X))$. Bob sends the content of the cell in question back. This is repeated until a leaf in the decision tree is reached and Alice knows the answer, i.e. for at most $t$ rounds.

□

We now show a lower bound for the communication problem. In the following lemma, we consider a general communication problem $h : A \times B \rightarrow C$, where Alice is given $a \in A$, Bob is given $b \in B$ and the objective of the game is to let Alice find $h(a, b)$. Again, Alice chooses her messages from $\{1, \ldots, s\}$ and Bob chooses his from $\{1, \ldots, k\}$.

**Lemma 3** *If a communication problem $h : A \times B \rightarrow C$ has a $t$ round protocol, then there is $A' \subseteq A$ and $B' \subseteq B$ so that $|A'| \geq |A|/s^t$ and $|B'| \geq |B|/k^t$ and so that*

$$\forall\, x \in A' \;\forall\, y, z \in B' : h(x, y) = h(x, z).$$

**Proof** By induction in $t$. The lemma clearly holds for $t = 0$, since if Alice can announce the answer without communicating with Bob, the function can only depend on her input. Now assume that it holds for $t$, and we will show it for $t + 1$. Let a communication problem $h$ with a $t + 1$ protocol $P$ be given. For $\alpha \in \{1, \ldots, s\}$, let $A_\alpha$ be those $x \in A$ for which Alice sends $\alpha$ as a first message when given $x$ as input. Fix $\alpha$, so that $|A_\alpha| \geq |A|/s$. For $\beta \in \{1, \ldots, k\}$, let $B_\beta$ be those $y \in B$ for which Bob send $\beta$ as the first message if $\alpha$ was the first message received. Fix $\beta$, so that $|B_\beta| \geq |B|/k$. The communication problem $h$, restricted to $A_\alpha \times B_\beta$ has a $t$ round protocol $P'$, doing the following: Simulate $P$ from the second round on, pretending that in the first round, Alice sent $\alpha$ to Bob and Bob sent $\beta$ to Alice. By the induction hypothesis, we can find $A'$ and $B'$ of the appropriate size.

□

**Proof of Theorem 1** Assume the communication game has a $t$ round protocol. Find $A' \subseteq K$ and a subset $B'$ of the polynomials over $K$ with degree at most $n$ with the properties stated in Lemma 3, i.e.

$$|A'| \geq k/s^t,$$

$$|B'| \geq k^{n+1}/k^t = k^{n+1-t}$$

and

$$\forall\, x \in A' \;\forall\, f(X), g(X) \in B' : f(x) = g(x).$$

Since two different polynomials of degree at most $n$ over a field can agree on at most $n$ points, we have that

$$|A'| \leq n \;\vee\; |B'| \leq 1$$

so

$$k/s^t \leq n \;\vee\; k^{n+1-t} \leq 1$$

and

$$t \geq \min(n + 1, \frac{\log k - \log n}{\log s}).$$

By Lemma 2, this is also a lower bound on the cell probe complexity of the original problem.

□

# 3 Application to dynamic problems

We need a slightly modified version of Theorem 1 for the application in this section. For $a \in \mathbf{Z}_m$, the ring of integers modulo $m$, we say that $a$ is *positive* if $a \in \{0, \ldots, \lceil m/2 \rceil - 1\}$. We consider a modified polynomial evaluation problem, where we only have to determine if $f(a)$ is positive.

**Theorem 4** *Let $K = \mathbf{Z}_p$, $p$ a prime. Any size bound $s$ cell probe algorithm solving the modified polynomial evaluation problem for polynomials of degree at most $n$ has complexity at least*

$$\min\left(\frac{n(1-1/p)}{\log p} + 1, \frac{\log p - \log n}{\log s}\right).$$

**Proof** We note that if, for $n$ different points $a_i$, there are $r$ polynomials which on each $a_i$ agree on whether their value is positive or not, then $r \leq \lceil p/2 \rceil^n$. Using this fact, we proceed as in the proof of Theorem 1.

□

We now define a language $L \subseteq \{0,1\}^*$. We let $L \cap \{0,1\}^n = \emptyset$ unless $n = m\lceil \log m \rceil + m$ for some integer $m$. Let $x$ be a string of length $m\lceil \log m \rceil + m$. The first $m\lceil \log m \rceil$ bits of $x$ are interpreted as the binary notation of the coefficients of a degree $m-1$ polynomial $f(X)$ over the ring $\mathbf{Z}_m$. If the last $m$ bits do not have the form $0^a 10^{m-a-1}$, then $x \notin L$. Otherwise $x \in L$ if and only if $f(a)$ is positive. Clearly, $L \in P$.

**Theorem 5** $L \notin \mathit{incr\text{-}TIME}(o(\frac{\log n}{\log \log n}))$.

**Proof** The method is similar to the lower bound proof for the Union-Split-Find problem in [9]. Suppose an implementation on a random access computer of the dynamic language membership problem for $L$ is given, with the complexity of the **change** and **query** operations being $o(\log n / \log \log n)$. Let $p$ be sufficiently large prime. Perform the **init**$(p\lceil \log p \rceil + p)$ operation. The content of any memory location now has to bounded by a polynomial in $p$ until the next **init** operation. These values include the values used for indirect addressing. We will not perform any more **init** operations, so in the rest of the proof, we can identify the set of possible states of the random access memory (the memory images) of the implementation with $\{1, \ldots, m\}^m$, where $m = p^{O(1)}$ (it is convenient for our proof that only a polynomial number of cells can be accessed, but it is possible to modify the proof to not take advantage of this fact. This would give a lower bound in a stronger model, the decision assignment tree model [6, 9]).

We now describe a cell probe algorithm $(o, \{T_a\})$ for the problem of evaluating polynomials in $\mathbf{Z}_p[X]$ of degree at most $\lceil \log p \rceil^2$.

Let $f(X)$ be such a polynomial. Starting from the initialized memory image $M_0 \in \{1, \ldots, m\}^m$, we perform a sequence of $\lceil \log p \rceil^3$ **change** operations, making the first $p\lceil \log p \rceil$ bits of $x$ into a representation of $f(X)$. Let the resulting memory image be $M_f \in \{1, \ldots, m\}^m$. By the assumption on the complexity of the **change** operations, $M_0$ and $M_f$ differ on at most $r = \lceil \log p \rceil^4$ indices. Let these indices be $a_1, a_2, \ldots, a_r$ and let the new content of $a_i$ be $d_i$. We need the following fact, due to Fredman, Komlós and Szemerédi [5]:

- Let $m$ be an integer. There is a scheme for storing sets $S \subset \{1,\ldots,m\} \times \{1,\ldots,m\}$ using $O(|S|)$ cells, each containing an element in $\{1,\ldots,m\}$, so that for each $j$, the query "Is $(j,x)$ in $S$ for some $x$, and if so, return such an $x$" can be answered using $O(1)$ probes.

We store the set $S = \{(a_i, d_i)\}$ using this scheme. The structure uses $O(r)$ cells, each containing an integer between 1 and $m$. Since $m = p^{O(1)}$, we can code the content of each cell as $O(1)$ elements of $\mathbf{Z}_p$ using any code we might like. By concatenating these codes, this gives us a vector in $(\mathbf{Z}_p)^{O(r)}$. We define $\phi(f(X))$ to be this vector. We now show that over this structure, $f(X)$ can be evaluated more efficiently by a family of decision trees than Theorem 4 permits. In order to evaluate $f(a)$, we would like to run the operations $\texttt{change}(p\lceil \log p \rceil + 1 + a, 1), \texttt{query}$ on $M_f$ since this would provide the answer. However, instead of $M_f$, we only have the structure containing the difference between $M_0$ and $M_f$ at our disposal. We simulate performing the operations as follows: Each time we want to write the value $d$ in a memory location $i$, we make a private note that the new content of location $i$ is $d$. If we want to read a memory location $a$, we first see if it appears in our private notes. If it does not, we look it up in the encoding of $S$. If it does not appear there, we know that its content is the same as it was in $M_0$. Changing and examining our private notes do not use $\phi(f(X))$, i.e. these actions can be hardwired into the decision tree, and so can the necessary knowledge about $M_0$. Each lookup requires only a constant number of probes, i.e. the total number of probes required to evaluate $f(X)$ on any point is $o(\log p/\log\log p)$ by the assumption on the complexity of **change** and **query**. But according to Theorem 4, the number of probes required is at least

$$\min\left(\frac{\lceil \log p \rceil^2 (1 - 1/p)}{\log p} + 1, \frac{\log p - \log(\lceil \log p \rceil^2)}{\log O(\lceil \log p \rceil^4)}\right) = \Omega(\log p/\log\log p),$$

a contradiction.

□

## Acknowledgements

I would like to thank Vlado Dančík, Gudmund S. Frandsen, Thore Husfeldt, and Mike Paterson, for helpful discussions.

# References

[1] M. Ajtai, A lower bound for finding predecessors in Yao's cell probe model, *Combinatorica* **8** (1988) 235-247.

[2] M. Ajtai, M. Fredman, J. Komlós, Hash functions for priority queues, in: *Proc. 24th Ann. IEEE Symp. on Foundations of Computer Science* (1983) 299-303.

[3] D. Angluin, L.G. Valiant, Fast probabilistic algorithms for Hamiltonian circuits and matchings, *J. Comput. System Sci.* **18** (1979) 155-193.

[4] E.G. Belaga, Evaluation of polynomials of one variable with preliminary processing of the coefficients, *Problemy Kibernet.* **5** (1961) 7-15.

[5] M.L. Fredman, J. Komlós, E. Szemerédi, Storing a sparse table with $O(1)$ worst case access time, *J. Assoc. Comput. Mach.* **31** (1984) 538-544.

[6] M.L. Fredman, M.E. Saks, The cell probe complexity of dynamic data structures, in: *Proc. 21st Ann. ACM Symp. on Theory of Computing* (1989) 345-354.

[7] D.E. Knuth, *The Art of Computer Programming, Vol. II: Seminumerical Algorithms* (Addison-Wesley, Reading, MA, 2nd ed., 1980).

[8] P.B. Miltersen, The bit probe complexity measure revisited, in: *Proc. 10th Symp. on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science, Vol. 665 (Springer, Berlin, 1993) 662-671.

[9] P.B. Miltersen, Lower bounds for Union-Split-Find related problems on random access machines, to appear in: *Proc. 26th Ann. ACM Symp. on Theory of Computing* (1994).

[10] P.B. Miltersen, S. Subramanian, J.S. Vitter, R. Tamassia, Complexity models for incremental computation, *Theoretical Computer Science*, to appear.

[11] V. Ya. Pan, Methods of computing values of polynomials, *Russian Math. Surveys* **21** (1) (1966) 105-136.

[12] A.C. Yao, Some complexity questions related to distributive computing, in: *Proc. 11th Ann. ACM Symp. on Theory of Computing* (1979) 209-213.

[13] A.C. Yao, Should tables be sorted?, *J. Assoc. Comput. Mach.* **28** (1981) 615-628.