Effect of Operators on Straight Line Complexity

(Extended Abstract)

Dan Boneh
dabo@bellcore.com

Richard J. Lipton*
rjl@cs.princeton.edu

Abstract

We show that certain explicit operators must vastly increase the straight line complexity of certain polynomials.

1 Introduction

This paper concerns lower bounds on the straight line complexity of multi-variate and univariate polynomials. We obtain two conditional results. Each result is of the form: if certain polynomials have low straight line complexity, then certain complexity results are true. In one case we show that if certain linear operators roughly preserve the complexity of polynomials than, co-NP is in AM. In the other case we show if the resultant of certain simple univariate polynomials has low complexity, then factoring of integers is easy.

We believe that these results are useful in two different ways. First, suppose that one believes, for example, that co-NP is unlikely to have short proofs, i.e. that co-NP is not a subset of AM. Then, our results can be viewed as proving consequences of this assumption. It seems interesting to us that an assumption about SAT has consequences in the pure algebraic complexity model. This is part of a larger program that we have worked on in the last few years. See [1, Ch. 1] for details about many other similar results.

Second, suppose that one believes that while it is unlikely that co-NP has short proofs, it is not impossible that such a result could be true. Then, our results present a novel approach to the problem. Just prove the existence of certain short computations for certain polynomials. This will yield the desired proof system.

Note, while such results are unlikely our methods would still yield interesting results if the size of the straight line programs were not polynomial but just better than naive bounds. Thus, it is entirely possible that the polynomials we consider can be proved to have straight line complexity that is small enough so that we can conclude that co-NP is in say 2^{n^ϵ}. The exact relationship will be presented in the full paper.

One of the most remarkable theorems concerning the straight line complexity of multi-variate polynomials is the theorem of Baur-Strassen [12]. This theorem asserts that the time it takes to compute all first order partial derivatives of a polynomial is only larger by a constant factor than the time to compute the polynomial itself. This theorem plays an important role in obtaining lower bound results. In particular, a lower bound on computing n functions sometimes can yield a lower bound on computing one function.

We study a natural generalization of this theorem. We note that Baur-Strassen bounds the complexity of the polynomial $T(f)$ in terms of the complexity of f where T is a particular linear operator. Our generalization is to consider other operators T. What we can show is that if there are similar bounds, then there are efficient proof systems for co-NP. Our proof systems make essential use of randomness and so we conclude that co-NP is a subset of AM. By a result of Boppana, Hastad, and Zachos [2] this implies that the polynomial hierarchy collapses to the second level.

The recent interest in proof systems and lower bounds for them is related directly to our results. The proof system that we construct is complete and sound. The latter is in the probabilistic sense since we use random sampling. Thus, another way to view our results is that unlike other proof systems proving non-trivial lower bounds on our system is going to be very difficult. This follows since any lower bound on the proof lengths of our system yields lower bounds on the size of straight line computations.

Finally, we construct new polynomials that are difficult to evaluate provided that factoring integers is hard. Constructing univariate polynomials with high straight line complexity is a classic problem. Our con-

*Supported in part by NSF CCR–9304718.

struction takes a few random low degree polynomials and combines them to produce a polynomial which must be difficult to evaluate (assuming factoring is hard). Consequently our technique gives a large collection of explicit hard polynomials.

1.1 Straight line programs

Let K be a field and $f \in K[X_1, \ldots, X_n]$ be a multivariate polynomial with coefficients in K. A straight line computation for f is a sequence of rational functions $g_1, \ldots, g_k \in K(X_1, \ldots, X_n)$ where each g_i satisfies one of the following: (1) g_i is the constant in K or is a variable X_i. (2) $g_i = g_j \circ g_k$ where $\circ$ is one of $+ - * \div$ and $j, k < i$. We say that n is the length of the computation. The *straight line complexity* of the polynomial f, denoted by $L(f)$ is the length of the shortest straight line computation for f.

2 Hard operators

Let K be a field. An operator is a map sending one multivariate polynomial over K to another. Here by multivariate polynomial we mean an element of the ring $K[X_1, X_2, \ldots]$ (the polynomial depends on a finite number of variables). Let T be such an operator. We say that T has *polynomial complexity* if there exists a constant c such that for all multivariate polynomials:

$$L(T(f)) \leq L(f)^c$$

A classic example of an operator with polynomial complexity is the differentiation operator. One can easily see that all polynomials $f(x_1, \ldots, x_n)$ satisfy $L(\frac{\partial f}{\partial x_i}) \leq 3L(f)$. In fact, an elegant result due to Strassen [12] shows that *all* partial derivatives can be simultaneously computed using a straight line program of length $3L(f)$. See [9] for a simple proof of this fact.

We say that an operator is *hard* if it does not have polynomial complexity. Hard operators increase the complexity of evaluating some polynomials by a super-polynomial amount. In this section we construct several hard operators. Our hardness results rely on a standard complexity assumption, namely that $co - NP \not\subseteq AM$.

Let $f(x_1, \ldots, x_n)$ be a multivariate polynomial. One can write it as

$$\begin{aligned} f(x_1, \ldots, x_n) &= g_m(x_2, \ldots, x_n)x_1^m + \\ & g_{m-1}(x_2, \ldots, x_n)x_1^{m-1} + \ldots + g_0(x_2, \ldots, x_n) \end{aligned}$$

Define the *smash* operator $S_1(f)$ to be the operator mapping f to the polynomial

$$S_1(f) \stackrel{def}{=} (g_m + g_{m-1} + \ldots + g_1)x_1 + g_0$$

All high powers of x_1 were "smashed" to a linear power. The operators S_i ; $i = 2, 3, \ldots$ are defined analogously to smash the corresponding variables $x_2, x_3, \ldots$. Notice that all smash operators are linear, i.e. $S_i(f + g) = S_i(f) + S_i(g)$ and $S_i(cf) = cS_i(f)$. Smash operators have been previously used in several contexts. For examples see [11, 3].

Theorem 2.1 *Suppose $co - NP \not\subseteq AM$. Let K be an infinite field and for all m define the operator $T_m = S_1 \circ S_2 \circ \cdots \circ S_m$ over the ring $K[X_1, X_2, \ldots]$. Then for some m the operator T_m is a hard operator.*

The proof of Theorem 2.1 relies on the following NP-complete problem which we refer to as SLP:
Instance: $f(x_1, \ldots, x_n) \in K[X_1, \ldots, X_n]$.
The polynomial f is *quadratic* in each variable x_i and is given as a straight line program containing no constants (each step in the program is either an indeterminate x_i or an arithmetic function of previous steps).
Problem: Do there exist $x_1, \ldots, x_n \in \{0, 1\}^n$ such that $f(x_1, \ldots, x_n) \neq 0$.

Without constraining the polynomial to be quadratic in each variable, SLP is clearly NP-complete. To reduce 3-SAT to it encode an "or" $x_1 \vee x_2$ as the polynomial $x_1 + x_2 - x_1 x_2$. Encode an "and" of clauses as a product of polynomials. Given a 3-CNF formula ψ The resulting polynomial is zero for all $\{0, 1\}^n$ inputs if and only if ψ is not satisfiable. Since 3-SAT is NP-complete even if each variable appears only three times in the formula [4, problem LO1] it immediately follows that SLP is NP-Complete when the polynomial is constraint to be cubic in each variable. It is interesting to note that SAT is in polynomial time if each variable appears at most *twice* in the formula (a simple resolution argument proves this). In contrast, the following lemma shows that SLP is NP-complete when the polynomial is quadratic in each variable.

Lemma 2.2 *SLP is NP-complete.*

Proof We reduce 3-Read 3-SAT to SLP (3-Read refers to the fact that each variable appears at most three times in the 3-CNF formula). Let ψ be a 3-Read 3-CNF formula using variables $x_1, \ldots, x_n$. Replace the three occurrences of a variable x_i by three new variables $x_i^{(1)}, x_i^{(2)}, x_i^{(3)}$. The resulting formula ψ' now uses each variable exactly once. Convert ψ' to a polynomial $p()$ as described above, i.e. $y \vee z$ becomes $y + z - yz$ and an "and" becomes a product. The resulting polynomial $p()$ is now multilinear, but we still need to enforce $x_i^{(1)} = x_i^{(2)} = x_i^{(3)}$ for all i. This is done using the polynomial

$$q(y, z, w) = 1 - y - z - w + yz + yw + zw$$

One can verify that for $y, z, w \in \{0,1\}$ we have $q(y,z,w) = 1$ when $y = z = w$ and $q(y,z,w) = 0$ otherwise. Define $p'()$ to be the polynomial:

$$p'() \quad = \quad p() \cdot q\left(x_1^{(1)}, x_1^{(2)}, x_1^{(3)}\right) \cdots q\left(x_n^{(1)}, x_n^{(2)}, x_n^{(3)}\right)$$

Since the polynomial $q()$ is multilinear, the polynomial $p'()$ is at most quadratic in each variable. Furthermore, there exists an input $X \in \{0,1\}^{3n}$ for which $p'(X) \neq 0$ if and only if ψ' has a satisfying assignment with

$$x_i^{(1)} = x_i^{(2)} = x_i^{(3)} \text{ for all } i = 1, \ldots, n$$

This completes the proof of the lemma. □

Proof of Theorem 2.1:
Suppose that for all m the operator T_m has polynomial complexity. We show that co-SLP is in AM. That is, we give a proof system, verifiable in probabilistic polynomial time, proving that a polynomial $f(x_1, \ldots, x_n)$ as above is identically zero on all $\{0,1\}^n$. The proof consists of the shortest straight line programs for the polynomials

$$T_1(f), T_2(f), T_3(f), \ldots, T_n(f).$$

By assumption, all operators $T_i(f)$ have polynomial complexity and therefore the total length of a valid proof is polynomial in the length of the input. Showing that this proof is verifiable in probabilistic polynomial time is done in two steps: first, we show how to verify that the given straight line programs indeed compute the correct polynomials. Second, we show how given the program for $T_n(f)$ one can test if $f = 0$ for all $\{0,1\}^n$ inputs.

We begin with the second step. Observe that the polynomial $T_n(f)$ is linear in each variable. Furthermore, inputs in $X \in \{0,1\}^n$ satisfy $f(X) = T_n(f(X))$ since these inputs are unaffected by the reduction in degrees. A multi-linear polynomial $g()$ which is zero on all $\{0,1\}^n$ inputs must be identically zero. This is easily proved by induction on n. Thus, if $f()$ is zero on all $\{0,1\}^n$ inputs then the polynomial $T_n(f)$ is identically zero on all of K^n. This fact can be efficiently tested using a result of Schwartz [10]:

Theorem 2.3 ([10]) *Let $f \in K[X_1, \ldots, X_n]$ be a polynomial of total degree d which is not identically zero. Let H be any finite subset of K. Then the number of zeros of f in H^n is bounded by $d|H|^{n-1}$.*

To test if $T_n(f)$ is identically zero the verifier picks some subset H of size n^3 of the field K. This is possible since K is infinite. The verifier then picks an element X at random from H^n and tests if $T_n(f)(X) = 0$ using the program for $T_n(f)$. With probability at least $1 - \frac{1}{n^2}$ this test will correctly determine if $T_n(f)$ is identically zero or not.

Finally we show how the verifier can check the validity of the straight line programs given in the proof. Define $g_i = T_i(f)$ and let $\hat{g}_i$ be the polynomial computed by the program for g_i given in the proof (we set $g_0 = \hat{g}_0 = f$). Suppose inductively the verifier is already convinced that $g_j = \hat{g}_j$ for all $j < i$ (initially $i = 1$). To check that $g_i = \hat{g}_i$ the verifier need only check that $\hat{g}_i = S_i(g_{i-1})$. To do so, the verifier must first check that $\hat{g}_i$ indeed computes a function that depends linearly on x_i. This is equivalent to checking that the second derivative of $\hat{g}_i$ with respect to x_i is identically zero. Using the result of Strassen [12] the verifier constructs a straight line program for $\frac{\partial^2 \hat{g}_i}{\partial x_i^2}$ which is at most 9 times larger than the original program for $\hat{g}_i$. Now we use a result of Ibarra and Moran [5] to verify that the resulting program indeed computes a function that is identically zero. Their test can be made to obtain the correct result with probability at least $1 - \frac{1}{n^2}$.

Now that $\hat{g}_i$ is known to be linearly dependent on x_i it can be written as

$$\hat{g}_i(x_1, \ldots, x_n) = \hat{h}_1()x_i + \hat{h}_0$$

where $\hat{h}_1(), \hat{h}_0()$ may depend on all variables except x_i. Similarly, write

$$g_{i-1}(x_1, \ldots, x_n) = h_2()x_i^2 + h_1()x_i + h_0()$$

Now the verifier need only check that $h_1 + h_2 \equiv \hat{h}_1$ and $h_0 \equiv \hat{h}_0$. In other words, the verifier must check that the polynomials $h_1 + h_2 - \hat{h}_1$ and $h_0 - \hat{h}_0$ are identically zero. To do this we use the result of Ibarra and Moran [5] again. First, we must show that the polynomials $h_1 + h_2 - \hat{h}_1$ and $h_0 - \hat{h}_0$ can be efficiently evaluated. Let $I = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) \in K^{n-1}$ be an $(n-1)$-tuple. To evaluate $\hat{h}_1(I)$ and $\hat{h}_0(I)$ the verifier computes $\hat{g}_i(x_1, \ldots, x_n)$ at two distinct values of x_i. He can then deduce the values of $\hat{h}_1(I)$ and $\hat{h}_0(I)$ by interpolation. Similarly, by evaluating the polynomial $g_{i-1}(x_1, \ldots, x_n)$ at three distinct values of x_i and interpolating the verifier can determine $h_2(I), h_1(I), h_0(I)$. Consequently, the polynomials $h_1 + h_2 - \hat{h}_1$ and $h_0 - \hat{h}_0$ can be efficiently evaluated. We can now use the result of Ibarra and Moran to verify that the resulting programs for $h_1 + h_2 - \hat{h}_1$ and $h_0 - \hat{h}_0$ indeed compute functions that are identically zero.

To summarize, throughout the verification process the verifier made a linear number of probabilistic tests. Each one of these tests can be made to succeeds with probability at least $1 - \frac{1}{n^2}$. It follows that for large enough n the probability that the verifier rejects an incorrect proof is greater than a $\frac{1}{2}$. A correct proof is always accepted. $\square$

An important point is that in the above proof the aggregate smash operators were only applied to polynomials where each variable has at most quadratic degree. It follows that these operators are hard even when applied to multi-quadratic polynomials.

An interesting corollary of Theorem 2.1 is that certain iterated differential operators must also blow up the complexity of some polynomials. Specifically, iterating a weighted version of the Laplacian must vastly increase the straight line complexity of some polynomials.

Corollary 2.4 *Suppose $co-NP \not\subseteq AM$. Let K be a field of characteristic 0 and define the operator*

$$\nabla_k(\alpha) = \sum_{i=1}^{k} \alpha(x_i) \frac{\partial^2 f}{\partial x_i^2}$$

for some univariate polynomial $\alpha(x)$. Then for some polynomial α and integers k, d the operator $\nabla_k^d(\alpha)$ (i.e. iterating $\nabla_k(\alpha)$ d times) must be a hard operator.

Proof We show how the operators T_i from Theorem 2.1 can be efficiently implemented using the ∇ operator. Define the operator:

$$R_i(f) = f + \frac{x_i - x_i^2}{2} \cdot \frac{\partial^2 f}{dx_i^2}$$

If $f(x_1, \ldots, x_n)$ is a polynomial in which x_i has degree at most two then R_i implements the smash operator, namely $R_i(f) = S_i(f)$. To see this write $f = a()x_i^2 + b()x_i + c()$ where $a(), b(), c()$ depend on all variable but x_i. Then $R_i(f) = \big(a() + b()\big)x_i + c() = S_i(f)$.

It follows that for multi-quadratic polynomials the operator T_k from Theorem 2.1 satisfies

$$T_k = R_1 \circ R_2 \circ \cdots \circ R_k$$

Define L_i to be the operator

$$L_i = \frac{x_i - x_i^2}{2} \cdot \frac{\partial^2 f}{dx_i^2}$$

Then $R_i = I + L_i$ where I is the identity operator. Hence,

$$\begin{aligned} T_k &= (I + L_1)(I + L_2) \cdots (I + L_k) = \\ & I + \sum_{1 \le i \le k} L_i + \sum_{1 \le i < j \le k} L_i L_j + \ldots + (L_1 L_2 \cdots L_k) = \\ & I + \nabla_k(\alpha) + \nabla_k^2(\alpha) + \ldots + \nabla_k^k(\alpha) \end{aligned}$$

where $\alpha = \frac{x - x^2}{2}$. The last equality follows from the fact that all multi-quadratic polynomials $f(x_1, \ldots, x_n)$ satisfy $L_i(L_i(f)) = 0$.

For multi-quadratic polynomials we just saw that all T_k operators can be efficiently implemented using iterations of the $\nabla_k(\frac{x-x^2}{2})$ operator. It follows from Theorem 2.1 that one of these operators must be hard. $\square$

3 Products of Galois polynomials

In this section we show that assuming the hardness of factoring one can construct explicit families of univariate polynomials that are hard to evaluate. A folk theorem states that the polynomials

$$f_n(x) = (x-1)(x-2)\cdots(x-2^n)$$

must have straight line complexity $L(f_n) > n^c$ for any c. Otherwise, one can efficiently factor integers. This argument was later generalized by Lipton [8] to show that any polynomial having "many" "small" rational roots must be hard to evaluate.

We extend these ideas one step further. We observe that if a polynomial $f(x) \in \mathbb{Z}[x]$ has "many" roots when reduced modulo "many" small primes then the polynomial must be hard to evaluate (assuming factoring is hard). We then construct families of such polynomials. Our construction is based on an operator that maps a small collection of *random* low degree polynomials to one hard polynomial. In what follows we sketch our results and leave the full details for the final version of the paper.

We begin by stating precisely what we mean by the hardness of factoring. The hardness of factoring states that there is no polynomial size circuit for factoring a non-negligible fraction of RSA-integers. More precisely, denote by $\mathbb{Z}_{(2)}(n)$ the set of n-bit integers which are a product of two large primes:

$$\mathbb{Z}_{(2)}(n) = \left\{ N \;\middle|\; \begin{array}{c} N < 2^n \;,\; N = pq \;, \\ p > q > 2^{n/4} \;,\; p, q \, \text{prime} \end{array} \right\}$$

Hardness of factoring assumption For any constant c and any family of polynomial size circuits $\{\mathcal{C}_n\}_{n=1}^{\infty}$ we have that for large enough n

$$\Pr_x \left[1 < \gcd\big(\mathcal{C}_n(x), x\big) < N\right] < \frac{1}{n^c}$$

where x is chosen uniformly at random from $\mathbb{Z}_{(2)}(n)$. Here $\mathcal{C}_n(x)$ is the output of the circuit $\mathcal{C}_n$ on input x. The output of the circuit is regarded as the binary representation of an integer. Notice that this is a non-uniform assumption.

Definition 3.1 *We say that a sequence of polynomials* $\{f_n\}_{n=1}^{\infty}$ *over* $\mathbb{Z}$ *is c-useful if infinitely often*

1. *The polynomial* $f_n(x)$ *has a root modulo* $1/n^c$ *fraction of the primes less than* 2^n.

2. *For all primes* $p < 2^n$, *if* $f_n(x)$ *has one root modulo* p *then it has at least* p/n^c *distinct roots modulo* p.

Lemma 3.1 *Assuming the hardness of factoring, a c-useful sequence must satisfy* $L(f_n) > n^d$ *infinitely often for any* $d > 0$.

Proof Sketch Suppose a c-useful sequence $\{f_n\}_{n=1}^{\infty}$ exists with $L(f_n) < n^d$. Then for a large enough n one can factor a non-negligible fraction of $\mathbb{Z}_{(2)}(m)$ for some $m \geq 2n$ as follows: given $N = pq \in \mathbb{Z}_{(2)}(m)$ pick a random $x \bmod N$ and evaluate $\gcd(N, f_n(x))$. Notice that $f_n(x) \bmod N$ can be computed in time $O(n^{d+2})$. Now, with non-negligible probability x is a root of $f_n(x)$ modulo one of the primes, but not the other. Consequently, we obtain a non-trivial factor of N. $\square$

This simple lemma shows that to construct explicit polynomials with high straight line complexity it suffices to construct c-useful sequences. Constructing such sequences is the goal of this section. In fact, we describe a family of operators that produce c-useful sequences on random inputs. Our approach constructs c-useful sequences $\{f_n\}_{n=1}^{\infty}$ where each f_n is a product of many high degree Galois polynomials (a Galois polynomial is one which generates a Galois number field). This ensures that property (2) is satisfied, since a Galois polynomial that has a root modulo a prime, splits completely into linear factors when reduced modulo that prime.

From here on we pick a fixed value for n and show how to construct a polynomial $h_n(x)$ that satisfies conditions (1) and (2) for primes less than 2^n and some fixed value c. By a random polynomial of degree d we mean a polynomial whose coefficients are picked independently at random from $-2^n, \ldots, 2^n$. For small values of d one can generate a random Galois polynomial by picking a random polynomial of degree d and adjoining all its conjugates. Usually this results in a Galois polynomial of degree $d!$. When d is small (e.g. 5) this is not a problem.

Let g be a polynomial of degree d and b an integer in $[0, 2^n]$. From here on we regard d as a small constant. Define the operator $R_{g,b} : \mathbb{Z}[X] \to \mathbb{Z}[X]$ by

$$R_{g,b}(h)(y) \stackrel{def}{=} \mathrm{Res}_x\Big(h(x),\ g(y - bx)\Big)$$

Here $\mathrm{Res}_x(f, g)$ stands for the resultant of f and g with respect to the variable x. For monic polynomials the resultant satisfies $\mathrm{Res}(f, g) = \mathrm{Res}(g, f \bmod g)$. Using this relation one can observe that the operator $R_{g,b}()$ does not greatly increase the complexity of a polynomial. More precisely,

$$L\big(R_{g,b}(h)\big) \leq d^2 L(h)$$

Lemma 3.2 *Let*

$$\{f_{i,j} \mid i = 0, \ldots, n\ ;\ j = 1, \ldots, d \log n\}$$

be a collection of random Galois polynomials in $\mathbb{Z}[x]$ *each of degree* d. *Set* $g_i(x) = \prod_{j=1}^{d \log n} f_{i,j}(x)$ *(note,* $g_i(x)$ *is a low degree polynomial, i.e. degree* $n^{O(1)}$*). Define the polynomial* $h_n(x)$ *as*

$$h_n(x) = R_{g_n,b} \circ R_{g_{n-1},b} \circ \cdots \circ R_{g_1,b}(g_0)$$

where $0 \leq b \leq 2^n$ *is an integer chosen at random. Then* $h_n(x)$ *is a polynomial of degree* $d^{2n} \log^n n$. *Furthermore, with probability more than half,* $h_n(x)$ *is a product of* $(d \log n)^n$ *Galois polynomials each of degree* d^n.

Proof Sketch Let u_1, u_2 be two irreducible polynomials of degree d. Let K_1, K_2 be the number fields generated by u_1, u_2 respectively. Suppose the smallest number field K containing both K_1 and K_2 has degree d^2. With high probability this will indeed be the case when u_1, u_2 are chosen at random. Then a generating polynomial for the number field K is

$$u(y) = \mathrm{Res}_x\big[u_1(x), u_2(y - bx)\big]$$

To see this observe that if α is a root of u_1 and β is a root of u_2 then $\beta + b\alpha$ is a root of u. The reason is that when $y = \beta + b\alpha$ both $u_1(x)$ and $u_2(y - bx)$ have a common root, namely $x = \alpha$. It is well known [7] that $\beta + b\alpha$ generates K with very high probability. Notice that $u()$ has degree d^2 as required.

When K_1, K_2 are Galois fields K is also a Galois field (since its degree is d^2). It follows that if

$$u_1(x) = f_{1,1} \quad \text{and} \quad u_i(y) = \mathrm{Res}_x\Big[u_{i-1}(x), f_{i,1}(y - bx)\Big]$$

then $u_n(x)$ is a Galois polynomial of degree d^n. This $u_n(x)$ is one factor of $h_n(x)$. The final observation that plays a role here is that

$$\begin{aligned}\mathrm{Res}(u_1 \cdot u_2, u_3 \cdot u_4) &= \\ &\mathrm{Res}(u_1, u_3) \cdot \mathrm{Res}(u_1, u_4) \cdot \mathrm{Res}(u_2, u_3) \cdot \mathrm{Res}(u_2, u_4)\end{aligned}$$

$\square$

The proof shows that $h_n(x)$ satisfies some more properties. Write $h_n(x) = \prod_{k=1}^{(d \log n)^n} u_k(x)$ where the u_k are Galois polynomial of degree d^n. Then the number field generated by a u_k is isomorphic to the number field generated by the polynomials $f_{1,j_1}, \ldots, f_{n,j_n}$ for some sequence $j_1, \ldots, j_n \subseteq \{1, \ldots, d \log n\}$. It follows that $u_k(x)$ has a root modulo a prime p if and only if all of $f_{1,j_1}, \ldots, f_{n,j_n}$ have a root modulo p.

An effective version of the Chebotarev density theorem due to Lagarias and Odlyzko [6] shows that the polynomials $f_{i,j}$ have a root modulo $1/d + o(1/n)$ of the primes less than 2^n (for this we have to use the fact that the degree d is constant, e.g. 5!, while n tends to infinity). Let $\mathcal{A}_{i,j}$ be the event that $f_{i,j}$ has a root modulo a prime p. Since the polynomials $f_{i,j}$ were chosen independently of one another, it follows that the events $\mathcal{A}_{i,j}$ are independent and each has probability $1/d + o(1/n)$.

Lemma 3.3 *For sufficiently large n, the polynomial $h_n(x)$ has a root modulo p with probability at least* $1/4$.

Proof We must show that some $u_k(x)$ has a root modulo p with probability at least $1/4$. Due to the structure of the u_k's this will happen if for all $i = 1, \ldots, n$ there exists a $j = 1, \ldots, d \log n$ such that $f_{i,j}(x)$ has a root modulo p. Thanks to independents of the events $\mathcal{A}_{i,j}$ this happens with probability:

$$\left[1 - \left(1 - \frac{1}{d} + o(\frac{1}{n})\right)^{d \log n}\right]^n \geq \frac{1}{4}$$

$\square$

The lemma shows that the polynomial $h_n(x)$ satisfies condition (1) of definition 3.1. As was stated above, condition (2) is satisfied since the factors of $h_n(x)$ are Galois polynomials. Therefore, if some $u_k(x)$ has a root modulo a prime p then it completely splits into linear factors modulo p. The degree of $u_k(x)$ is d^n implying that it has that many roots modulo p. Some of these roots are multiple roots. However using the fact that the $f_{i,j}(x)$ were chosen randomly and independently it is not difficult to show that the number of distinct roots modulo p is a constant fraction of p. Consequently, a sequence of polynomials $\{h_n\}_{n=1}^{\infty}$ is c-useful with probability at least half.

To summarize, we sketched the proof of the following theorem:

Theorem 3.4 *Define the polynomials $h_n(x)$ as in Lemma 3.2. Then the hardness of factoring implies that with probability at least half, infinitely often the sequence $\{h_n\}_{n=1}^{\infty}$ has complexity $L(h_n) > n^d$ for all constants d.*

In words, we were able to show that n iterations of the operator $R_{g_i,b}$ must increase the complexity of a univariate polynomial by a super polynomial amount. Furthermore, the polynomials g_i are small products of *random* Galois polynomials of degree d. Hence, a combination of n *randomly* chosen operators $R_{g_i,b}$ must be a hard operator, assuming factoring is hard.

4 Summary

We prove lower bounds on straight line computations assuming standard complexity theoretic assumptions. The first result shows that certain natural operators acting on multivariate polynomials must increase the straight line complexity by a super-polynomial amount. The result is based on the assumption that $co-NP \not\subseteq AM$. As a corollary we proved that certain differentiation operators must be hard to evaluate.

The second result shows that certain univariate polynomials must be hard to evaluate, assuming factoring is hard. Our polynomials are constructed by carefully combining a small number of low degree polynomials. The main tool used in the construction is the resultant operator. Hence, our results show that for *random* inputs, iterating the resultant operator the way we do increases the complexity of its operands by a super-polynomial amount.

Acknowledgments

We thank the anonymous referees for their helpful comments.

References

[1] D. Boneh, "Studies in computational number theory with applications to cryptography", Princeton University technical report CS-TR-523-96.

[2] R. Boppana, J. Hastad, S. Zachos, "Does co-NP have short interactive proofs?", Information Processing Letters, Vol. 25, no. 2, 1987, pp. 127–132.

[3] M. Clegg, J. Edmonds, R. Impagliazzo, "Using the Groebner Basis Algorithm to Find Proofs of Unsatisfiability", Proc. of STOC96, 1996, pp. 174–183.

[4] M. Garey, D. Johnson, "Computers and intractability", W. H. Freeman, 1979.

[5] O. Ibarra, S. Moran, "Probabilistic algorithms for deciding equivalence of straight line programs", JACM 30 (1), pp. 217–228, 1983.

[6] J. Lagarias, A. Odlyzko, "Effective versions of the Chebotarev density theorem", Algebraic Number Fields, ed. A. Frohlich, Academic Press, pp. 409–464, 1977.

[7] S. Lang, "Algebraic number fields", Spinger-Verlag.

[8] R. Lipton, "Straight line complexity and integer factorization", Proc. of ANTS, 1994.

[9] J. Morgenstern, "How to compute fast a function and all its derivatives", SIGACT News, 16 (4), pp. 60–62, 1985.

[10] J. Schwartz, "Fast probabilistic algorithms for verification of polynomial identities", JACM 27 (4), pp. 701–717, 1980.

[11] A. Shen, "IP = PSPACE: Simplified Proof", JACM 39 (4), pp. 878–880, 1992.

[12] V. Strassen, "Die berechnungskomplexitat vo elementarsymmetrischen Funktionen und von interpolationskoefizienten", Numer. Math. 20, pp. 238–251, 1973.