Acta Informatica 6, 153—169 (1976)

Another Polynomial Homomorphism[*]

Robert T. Moenck

Received May 13, 1975

Summary. The current proposals for applying the so called "fast" $O(N \log^a N)$ algorithms to multivariate polynomials is that the univariate methods be applied recursively, much in the way more conventional algorithms are used. Since the size of the problems is rather large for which a "fast" algorithm is more efficient than a classical one, the recursive approach compounds this size completely out of any practical range.

The degree homomorphism is proposed here as an alternative to this recursive approach. It is argued that methods based on the degree homomorphism and a "fast" algorithm may be viable alternatives to more conventional algorithms for certain multivariate problems in the setting of algebraic manipulation. Several such problems are discussed including: polynomial multiplication, powering, division (both exact and with remainder), greatest common divisors and factoring.

1. Introduction

In recent years there has been a considerable interest in devising algorithms for operations on polynomials which are more efficient than the more traditional methods. These new algorithms are generally based on the use of the Fast Fourier Transform (FFT) to perform polynomial multiplication in $O(N \log N)$[1] steps and are generally characterised by an $O(N \log^a N)$, $a \in \{1, 2, \ldots\}$ run time. A partial list of such algorithms is shown in Figure 1.

Univariate polynomial multiplication is itself a good example of such an algorithm. If we assume that the coefficients are drawn from an appropriate finite field so that all coefficient operations are exact and require one step to add

Fig. 1 A list of some fast algebraic algorithms

Polynomial operation	Operation count	Ref.
Multiplication	$O(N \log N)$	[1]
Exact division	$O(N \log N)$	[1]
Powering	$O(dN \log dN)$	[2]
Division with remainder	$O(N \log N)$	[3]
GCD and resultants	$O(N \log^2 N)$	[4]
Multipoint Evaluation	$O(N \log^2 N)$	[5]
Interpolation	$O(N \log^2 N)$	[5]
Factoring	$O(N^2 \log^4 N)$	[6]

[*] This research was supported by NRC Grant A9284.

[1] All logarithms are base 2

Table 1. A comparison of the classical vs. the FFT method of univariate polynomial multiplication. Times are taken fron an Algol-W code running on an IBM 360-75 for multiplication of polynomials with coefficients in the finite field GF (40961). Note that both the theoretical and empirical times indicate a crossover point at degree 31

N degree + 1	Classical method		FFT method	
	Theoretical	Empirical	Theoretical	Empirical
	$2N^2-2N+1$	Secs./60	$9\,N \log N+19\,N$	Secs./60
2		0.01		0.12
4		0.03		0.16
8		0.08		0.3
16	481	0.29	880	0.55
32	1,985	1.10	2,048	1.05
64	8,055	4.25	4,672	2.3
128		17		5.0
256		66		11

or multiply (we shall assume this throughout the paper) then the product of two polynomials of degree $N-1$ can be formed in $2N^2$ operations using the classical algorithm. This product can also be formed using the FFT method in $9\,N \log N+19N$ steps (see [6]).

These two operation count functions can be compared by evaluating them at some sample values. As Table 1 shows the FFT method theoretically is more efficient than the classical method for polynomials of degree greater than 31. An implementation of the two algorithms in Algol-W bears out this cross-over point of 31. However, whereas the classical method increases quadratically with the degree, the FFT method is only slightly faster than linear. Therefore, once the cross-over point has been reached, in the time the classical algorithm could form the product of twice the degree, the FFT method could multiply two polynomials of almost *four* times the degree.

We can draw an analogy with transportation systems in that the classical method is rather like a bus, being a good method to travel a distance of several miles. On the other hand, the FFT method is like an airplane, a good method to travel a distance of several hundred miles but not a few city blocks.

Part of the motivation for developing these methods was to derive algorithms which could be used in the algebraic manipulation setting. Although products of polynomials of degree greater than 31 do arise in algebraic manipulation they do not come up very frequently. Generally problems are much smaller.

We can conclude that the kinds of polynomials which are suitable for these fast algorithms are those with a large number of coefficients. However, problems in algebraic manipulation tend to be characterised by polynomials of low degree. This leads us to look at multivariate problems which lie in the intersection of these two constraints. In the past, it has been proposed (cf. [2, 4] or [7]) that to derive a multivariate algorithm one should apply the univariate algorithms recursively. However, here we will look at an alternative approach.

2. Polynomial Multiplication

Consider the problem of multiplying together two dense bivariate polynomials $p(x, y)$, $q(x, y)$ of degree n in each variable. There are several approaches we might take:

I. The Recursive Classical Method

The classical high school multiplication algorithm for ***univariate*** polynomials of degree n uses $(n+1)^2$ coefficient multiplications and n^2 coefficient additions. We can apply this method recursively to the bivariate case using:
$(n+1)^2$ univariate polynomial multiplications

$$O\ 2n^2+2n+1=(n+1)^2(2n^2+2n+1)$$

n^2 univariate polynomial additions

$$O(2n+1)=(2n+1)\ n^2$$

$$\text{Total}=2n^4+8n^3+8n^2+4n+1 \text{ coefficient operations}$$

which is asymptotically $O(n^4)$.

II. The Homomorphism Method

Over the past few years there has been developed a class of algorithms which can be described as homomorphism methods (see Lipson [8] or Horowitz [9] for surveys). These methods are based on the idea that evaluation of a polynomial at a point, is a homomorphism of a polynomial ring $\mathscr{R}[x]$ to its base ring $\mathscr{R}$. Since ring operations are preserved under the homomorphism, the original polynomial can be recovered by interpolation at sufficiently many points. The origin of this idea has been ascribed to Kronecker. For our bivariate case this would use:
Evaluation of $2\cdot(n+1)$ polynomial coefficients at $2n+1$ points

$$=2(n+1)\cdot(2n+1)\cdot 2n$$

Multiply the $2n+1$ univariate polynomials produced together

$$=(2n+1)\ (2n^2+2n+1)$$

Interpolate the $2n+1$ coefficients to give the polynomial coefficients

$$=2(n+1)^3$$

$$\text{Total}=(2n+1)\ (14n^2+14n+3) \text{ coefficient operations}$$

which is $O(n^3)$ asymptotically.

III. Recursive Fourier Transform Method

The idea used in ii) can be extended by use of the finite Fourier transform (cf. Pollard [1] or Bonneau [7]) which essentially provides a slick way to evaluate a polynomial at the m roots of unity (where $m=2^r$) in $m+3/2\,m\log m+$lower degree terms coefficient operations. The interpolation phase is accomplished by an inverse transform which takes $2m+3/2\,m\log m+ldt.$ coefficient operations. Since the FFT is an $O(m\log m)$ algorithm it can be effectively applied all the way down to the base ring to give an $O(m \log m)$ *univariate* polynomial multiplication algorithm. In our bivariate case we use:

$2 \cdot (n+1)$ applications of the FFT to the coefficient polynomials

$$= 2(n+1) \cdot (m + 3/2\, m \log m)$$

$2m$ applications of the FFT to give base ring elements

$$= 2m(m + 3/2\, m \log m)$$

multiplication of the base ring elements

$$= m^2$$

m applications of the inverse FFT to give univariate polynomials

$$= m(2m + 3/2\, m \log m)$$

m applications of the inverse FFT to give the coefficient polynomials

$$= m(2m + 3/2\, m \log m)$$

$$\overline{\text{Total} = (7m^2 + 6m^2 \log m) + (n+1) \cdot (2m + 3\, m \log m)}$$

However in order to get this $O(m^2 \log m)$ asymptotic behavior we must choose

$$m = 2^r \text{ where } r = [\log_2(2n+1)].$$

This means essentially that we must append zero leading coefficients to a polynomial. For large n the penalty is worthwhile since we get a faster algorithm despite this handicap. However for small values of n there is a marked discontinuity in the operation counts for n close to a power of 2. (cf. Table 2).

IV. Degree Homomorphism Method

In view of the fact that algorithms based on the FFT tend to have high crossover points we would expect that the recursive FFT algorithm would not perform as well as methods i) and ii) for polynomials of small degree. The question is how we can adapt the problem of small bivariate polynomial multiplication to a situation where the FFT would be used to its best effect.

One possibility proposed here is that we map the bivariate problem onto a univariate problem. Consider the multiplication of $p(x, y) \cdot q(x, y)$ where:

$$p(x, y) = x(2y+1) + (-y+2)$$
$$q(x, y) = x(y+3) + (4y-3)$$

since the product will be of degree two in each variable we might try substituting $x = y^3$ in $p(x, y)$ and $q(x, y)$ to get $p'(y)$ and $q'(y)$ respectively.

Now:

$$p'(y) = 2y^4 + y^3 - y + 2$$
$$q'(y) = y^4 + 3y^3 + 4y - 3$$

and

$$p'(y)\cdot q'(y)=2y^8+7y^7+3y^6+7y^5-3y^4+3y^3-4y^2+11y-6.$$

Taking the remainder of successive quotients with respect to $\{y^{3i}\}$, $2\geqq i\geqq 0$. We invert the substitution to get

$$(2y^2+7y+3)x^2+(7y^2-3y+3)x+(-4y^2+11y-6).$$

This is the product of $p(x,y)\cdot q(x,y)$ as can be verified by carrying out the multiplication in some more conventional way. This method is usually known as "Kronecker's Trick".

In general for bivariate polynomials the validity of this method is established by:

Lemma 1. Over the polynomial ring $\mathscr{R}[x,y]$ the mapping:

$$\phi:\mathscr{R}[x,y]\to\mathscr{R}[y]$$
$$\phi: x\mapsto y^m,\qquad m\geqq 0$$

is a homomorphism of rings.

Proof. We must show that for polynomials $p(x,y)$, $q(x,y)$:

$$\phi(p(x,y))\underset{*}{+}\phi(q(x,y))=\phi(p(x,y)\underset{*}{+}q(x,y)).$$

This can be done by a suitable grouping of terms ot the homomorphic images of p and q and some elementary manipulation of sum and product formulae for polynomials. □

An alternative way of viewing this method is as a homomorphism on the multiplicative monoid of monomials in x and y. Lemma 1 can then be had as a corollary of a much more general theorem given in [10, Ch. V §2]

Theorem 1. Universal mapping theorem for polynomial rings. Let A, B be commutative rings, and let f_0: A→B be an A-algebra. Let S be a subset of B, which generates B, and assume that the elements of S are algebraically independent over A. Let A^1 be a commutative ring. Let f: $A\to A^1$ be ring-homomorphism, and λ: $S\to A^1$ a map. Then there exists a unique ring homomorphism h: $B\to A^1$ such that the diagram is commutative:

$$\begin{tikzcd}
B \arrow[r, "h"] & A^1 \\
A \arrow[u, "f_0"] \arrow[ur, "f"'] &
\end{tikzcd}$$

and such that the restriction of h to S is equal to λ.

For *Lemma 1.*

$$A=A^1=\mathscr{R}[y],\qquad B=\mathscr{R}[x,y],$$
$$S=\{x\}\quad\text{and}\quad h=\phi.$$

We can obtain an "inverse" for the homomorphism ϕ as follows:

Lemma 2. Let ψ be the homomorphism of free $\mathscr{R}$-modules:

$$\psi\colon \mathscr{R}[x] \to \mathscr{R}[x, y]$$
$$\psi\colon y^k \to x^q y^r$$

where $k = q \cdot m + r$, $0 \leqq r < m$. Then for a polynomial $p(x, y)$ in $\mathscr{R}[x, y]$ $\psi\;(\phi(p(x, y))) = p(x, y)$ iff $m > \partial_y(p)$ (where $\partial_y(p)$ is the maximum degree p in the variable y).

Proof. If $m > \partial_y(p)$ then the process of taking remainders wrt. $y^{m(i+1)}$, $i = 0, \ldots, n$ will generate the coefficient polynomials of $p_i(y)$ of the original polynomial uniquely. If $m \not> \partial_y(p)$ then when ϕ is applied to p some of the coefficients in $\mathscr{R}$ will be added together and so $p^1(y) = \phi(p(x, y))$ will correspond to an infinite set of bivariate polynomials. □

Further we note:

Lemma 3. The degree of the image polynomial under the mapping

$$\phi\colon x \to y^m \quad \text{is} \quad \partial_y(p) \cdot m + \partial_x(p).$$

Returning to our problem of bivariate polynomial multiplication we can apply degree-homomorphism $x = y^{2n+1}$ to the bivariate polynomials $p(x, y)$ and $q(x, y)$ to give polynomials $p^1(y)$, $q^1(y)$ of degree $(2n+1)\,n + n = 2(n+1)\,n$ we can multiply these together using the univariate FFT polynomial multiplication method:

a) 2 forward FFTs	$= 2(m + 3/2\, m \log m)$
b) multiplication of the sequences	$= m$
c) one inverse transform	$= 2m + 3/2\, m \log m$
	Total $= 9/2\, m \log m + 5m$

Again because we are dealing with the Fourier transform we must choose:

$$m = 2^r \quad \text{but here } r = [\log_2 (2n+1)^2]$$

Also we have a discontinuous operation count function. However since we are dealing with the larger values of m than before, the steps are not so high as with the recursive FFT method.

Small initial values of the four operation count functions are displayed in Table 2. The table illustrates that the degree homomorphism method is more efficient than the classical polynomial multiplication method for all polynomials of degree greater than ten in each variable. For the recursive FFT method the cross-over point is at degree 20. Surprisingly, the evaluation homomorphism method which was once considered a reasonable alternative to the classical method has a higher cross-over point than the degree homomorphism method.

Table 2. Operation counts of the four methods for multiplying 2 bivariate dense polynomials of degree N

Degree N	Recursive classical	Evaluation	Recursive FFT	Degree homomorphism
2	137	435	1,192	880
4	1,169	2,547	7,264	4,670
6	4,633	7,683	7,712	10,496
8	12,833	17,187	47,392	23,296
10	28,841	32,403	48,480	23,296
12	56,497	54,675	49,568	51,200
14	100,409	85,347	50,656	51,200
16	165,953	125,763	328,960	111,616
18	259,273	177,267	331,520	111,616
20	387,281	241,203	334,080	111,616
22	557,657	318,915	336,640	111,616
24	778,849	411,747	339,200	241,664
26	1,060,073	521,043	341,760	241,664
28	1,411,313	648,147	344,320	241,664
30	1,843,321	794,403	346,880	241,664

3. Larger Numbers of Variables

We can apply the "Kronecker trick" to polynomials with larger numbers of variables. This is shown by:

Theorem 2. In the polynomial ring $\mathscr{R}[x_1, \ldots, x_v]$, $v \geqq 2$. The mapping

$$\phi: \mathscr{R}[\bar{x}] \to \mathscr{R}[x_1]$$

$$\phi: x_i \mapsto x_1^{n_i}, \quad 1 \leqq i \leqq v$$

where $n_v > \cdots > n_1 = 1$ is a homomorphism of rings.

Proof. By induction on v. If $v=2$ then we can apply Lemma 1 to obtain our result. If the result is true up to $v-1$ then we can apply our hypothesized homomorphism to the coefficient polynomials to obtain a mapping:

$$\phi_{v-1}: \mathscr{R}[\bar{x}] \to \mathscr{R}[x_2, x_1]$$

and then apply Lemma 1 for a mapping

$$\phi_1: \mathscr{R}[x_2, x_1] \to \mathscr{R}[x_1].$$

Since homomorphisms are closed under composition the result follows. □

Again we could have obtained this result as a corollary of the Universal Mapping Theorem for Rings. We note that the kernel of the homomorphism is:

$$(x_2 - x_1{}^{n_1}, \ldots, x_v - x_1{}^{n}v)$$

We can use the following notation:

Definition. For a polynomial $p(\bar{x}) \in \mathscr{R}[x]$, $\partial_i(p)$ is the maximum degree of the polynomial in the variable x_i.

Now we can state the multivariate analogues of Lemmas 2 and 3.

Theorem 3. Let ψ be the homomorphism of free $\mathscr{R}$-modules defined by:

$$\psi\colon \mathscr{R}[x_1] \to \mathscr{R}[x_1, \ldots, x_v]$$

$$\psi\colon x_1^k \mapsto \begin{cases} 1 \text{ if } k=0 \\ \psi(x_1^r)\cdot x_i^q \end{cases}$$

where

$$n_{i+1} > k \geqq n_i, \quad k = q\cdot n_i + r$$

$$0 \leqq r < n_i \quad \text{and} \quad n_v > \cdots > n_1 = 1.$$

Then

$$\forall p(\bar{x}) \in \mathscr{R}[\bar{x}], \quad \psi(\phi(p)) = p$$

iff

$$\forall i, \quad 1 \leqq i < v, \ \sum_{j=1}^{i} \partial_j(p)\, n_j < n_{i+1}.$$

Proof. It is sufficent to consider $p = x_1^{a_1} \ldots x_v^{a_v}$ and an induction on $i = \max_{a_j \neq 0} j$.

If $i=2$ then there are only two relevant variables and so Lemma 2 applies. To show the result for i from the hypothesis for $i-1$ it is sufficient to observe that the division $k = q\cdot n_i + r$, $n_{i+1} > k \geqq n_i$, $0 \leqq r < n_i$ uniquely defines the exponent q of x_i if : $\sum_{j=1}^{i} \partial_i(p)\cdot n_j < n_{i+1}$.

If $\sum_{j=1}^{i} \partial_i(p)\cdot n_j \geqq n_i + 1$ then clearly several monomials could be mapped onto the same power of x_1 and so the mapping is no longer unique. □

Note that this result in effect shows that for the set of polynomials for which the relation:

$$n_{i+1} > \sum_{j=1}^{i} \partial_j(p)\cdot n_j, \quad 1 \leqq i < v, \quad \text{where } n_1 = 1.$$

holds are isomorphic to their univariate images. Thus any polynomial ring operation on elements of the set giving results in the set will be preserved by the isomorphism. In this sense ϕ behaves like a ring isomorphism on this set of polynomials.

Another viewpoint is to choose a set of multipliers $\{n_j\}$ of the exponents of a polynomial in such a fashion that we get the desired isomorphism. If we have $m_i > \partial_i(p)$, $1 < i < v$ and $m_1 = 1$ then it is evident that:

$$n_{i+1} \geqq \sum_{j=1}^{i} \partial_j(p) \prod_{k=1}^{j} m_k$$

i.e., $n_j = \prod_{k=1}^{j} m_j$ is such a set of multipliers.

This in turn means that we can consider the map ϕ as being recursively defined as:

$$\phi\colon x_i \to x_{i-1}^{m_i}, \quad 1 < i \leqq v.$$

Theorem 4. The degree of the image polynomial under the map:

$$\phi\colon x_i \to x_{i-1}^{m_i}, \quad 1 < i \leqq v$$

is:

$$\deg(p') = ((\dots(\partial_v(p)\cdot m_{v-1} + \partial_{v-1}(p))\cdot m_{v-2} + \cdots)\, m_1 + \partial_1(p)$$

i.e., if $m_i = m$, $1 \leqq i, < v$ then

$$\deg(p') = \sum_{i=1}^{v} \partial_i(p)\, m^{v-i}$$

$$\leqq (m+1)^v, \quad \text{if } \partial_i(p) \leqq m, \quad 1 \leqq i < v. \quad \Box$$

This means that if we are going to apply the degree,homomorphism method in an algorithm for some polynomial operation, we must have reasonably tight bounds on the degree of result.

We can now proceed to establish bounds for some of the simpler polynomial operations.

Theorem 5. If a$(\bar{x})$, $b(\bar{x}) \in \mathscr{R}[\bar{x}]$ where $\mathscr{R}$ contains no divisors of zero. Then for the product:

$$c(\bar{x}) = a(\bar{x})\cdot b(\bar{x})$$

$$\partial_i(c) = \partial_i(a) + \partial_i(b), \quad 1 \leqq i \leqq v.$$

Proof. By induction on v:

If $v=1$ then $\deg(c) = \deg(a) + \deg(b)$.

If we assume the result is for $v-1$ and consider the polynomials a, b, c in recursive canonical form with main variable x_v. Then

$$\partial_v(c) = \partial_v(a) + \partial_v(b).$$

The coefficients of x_v in c will be sums of products of polynomials in the subsidiary variables $\{x_i\}$, $1 \leqq i < v$. For any product of two such coefficients a_j and b_k:

$$\partial_i(a_j\cdot b_k) = \partial_i(a_j) + \partial_i(b_k), \quad 1 \leqq i < v \text{ by hyp.}$$

For any sum of such products the relation will hold since the degree of the sum of two polynomials cannot be greater than their maximum degree. Therefore:

$$\partial_i(c) = \partial_i(a) + \partial_i(b), \quad 1 \leqq i \leqq v. \quad \Box$$

A well known consequence of this result (cf. [10], Ch V §3) is that $\partial_i(a)$ is a valuation of $\mathscr{R}[\bar{x}]$.

A number of corollaries follow immediately from this theorem.

Corollary 1. If $b(\bar{x}) = a(\bar{x})^d$ where a, $b \in \mathscr{R}[\bar{x}]$ then $\partial_i(b) = d\cdot \partial_i(a)$ for $1 \leqq i \leqq v$.

Corollary 2. If $b(\bar{x})$ divides some polynomial $c(\bar{x})$ exactly to give a quotient $a(\bar{x}) = c(\bar{x})/b(\bar{x})$ where a, b, $c \in \mathscr{R}[\bar{x}]$ then

$$\partial_i(a) = \partial_i(c) - \partial_i(b), \quad 1 \leqq i \leqq v.$$

and also

Corollary 3. If $a(\bar{x})$ is a common divisor of two polynomials $b(\bar{x})$ and $c(\bar{x})$ where a, b, $c \in \mathscr{R}[\bar{x}]$ then

$$\partial_i(a) \leqq \min(\partial_i(b), \partial_i(c)), \quad 1 \leqq i \leqq v.$$

General Multivariate Polynomial Multiplication

We can perform a similar analysis of the four methods of polynomial multiplication considered in Section 2 in terms of the degree n and v the number of variables of the polynomials involved. The operation counts are best described in terms of recurrence relations:

I. The Recursive Classical Method

Let MCPM (n, v) be the time to multiply dense polynomials of degree n in each of v variables using this method.

$$\begin{aligned} \mathrm{MCPM}(n, v) &= (n+1)^2 \text{ coefficient multiplications} \\ &\quad + n^2 \text{ summations of these products} \\ &= (n+1)^2\, \mathrm{MCPM}(n, v-1) + n^2(2n+1)^{v-1} \end{aligned}$$

with $\mathrm{MCPM}(n, 0) = 1$. Its's asymptotic form is $\mathrm{MCPM}(n, v) = O(n^{2v})$.

II. The Evaluation Homomorphism Method

Let $\mathrm{MEPM}(n, v)$ be the operation count function for this method

$$\begin{aligned} \mathrm{MEPM}(n, v) = {} & \text{steps to evaluate } 2\cdot(n+1)^{v-1} \text{ coefficient polynomials of degree } n \text{ at } (2n+1) \text{ points} \\ & + \mathrm{MEPM}(n, v-1)\cdot(2n+1) \\ & + \text{steps to interpolate } (2n+1)^{v-1} \text{ coefficients at } (2n+1) \text{ points} \end{aligned}$$

with

$$\mathrm{MEPM}(n, 1) = (2n^2 + 2n + 1)$$

thus

$$\begin{aligned} \mathrm{MEPM}(n, v) = {} & 2(n+1)^{v-1}\cdot 2n(2n+1) \\ & + (2n+1)\cdot \mathrm{MEPM}(n, v-1) \\ & + (2n+1)^{v-1}\cdot 2(2n+1)^2. \end{aligned}$$

The asymptotic form is $\mathrm{MEPM}(n, v) = O(v\, n^{v+1})$.

III. The Recursive Fourier Transform Method

Let $\mathrm{MFPM}(n, v)$ be the operation count function for this method. It will be similar to that for MEPM.

$$\begin{aligned} \mathrm{MFPM}(n, v) = {} & 2\cdot(n+1)^{v-1}(m + 3/2\, m \log m) \\ & + m\cdot \mathrm{MFPM}(n, v-1) \\ & + m^{v-1}\cdot(2m + 3/2\, m \log m) \end{aligned}$$

with MFPM $(n, 0) = 1$ and $m = 2^r$, $r = [\log_2(2n+1)]$. The function is asymptotically: $\mathrm{MFPM}(n, v) = O(v\, n^v \log n)$.

Table 3. Operation counts of the four methods for multiplying to dense polynomials in 3, 4, 5 and 6 variables of degree N in each variable.

Vars.	Degree N	Recursive classical	Evaluation	Recursive FFT	Degree homomorphism
3	2	1,333	3,785	10,328	4,672
	3	7,849	14,525	11,648	23,296
	4	30,521	39,645	121,824	51,200
	5	91,981	88,385	127,872	111,616
	6	233,101	172,289	134,368	241,664
	7	520,913	305,205	141,312	241,664
	8	1,057,969	503,285	1,560,608	520,192
	9	1,993,141	784,985	1,588,352	520,192
	10	3,533,861	1,151,065	1,617,184	1,114,112
	11	5,959,801	1,684,589	1,647,104	1,114,112
	12	9,637,993	2,350,925	1,678,112	1,114,112
	13	15,039,389	3,197,745	1,710,208	2,375,680
	14	22,756,861	4,255,025	1,743,392	2,375,680
	15	33,524,641	5,555,024	1,777,664	2,375,680
	16	48,239,201	7,132,389	21,423,360	5,046,272
4	2	12,497	26,255	, 85,000	51,200
	3	128,671	140,665	98,816	241,664
	4	774,689	492,903	1,977,184	520,192
	5	3,344,591	1,341,857	2,094,336	1,114,112
	6	11,501,041	3,089,359	2,226,720	2,375,680
	7	33,503,807	6,311,865	2,375,680	5,046,272
	8	86,009,921	11,792,135	50,336,032	10,682,368
	9	199,869,679	20,550,913	51,371,264	10,682,368
	10	428,523,281	33,878,607	52,473,952	22,544,384
5	2	114,937	165,765	687,128	241,664
	3	2,080,345	1,241,457	813,056	2,375,680
	4	19,472,201	5,589,009	31,774,944	5,046,272
	5	120,771,301	18,588,669	33,799,680	22,544,384
6	2	1,047,257	994,795	5,518,408	1,114,112
	3	33,436,788	10,423,301	6,594,560	10,682,368
	4	487,749,809	60,317,019	509,099,104	99,614,720
	5	4,351,793,111	245,160,421	542,536,704	208,666,624

IV. The Degree Hommorphism Method

Let MDPM (n, v) be the operation count function for this method. It will have the form:

$$\text{MDPM}\ (n, v) = (5m + 9/2\, m \log m)$$

where $m = 2^r$ and $r = [\log_2 (2n+1)^v]$.

Again the asymptotic form is MDPM $(n, v) = O(v\, n^v \log n)$. But in fact it does grow slightly faster than MFPM (n, v), because of its lower order terms.

The values of these functions for $v = 3$, 4, 5 and 6 and small n are displayed in Table 3.

The values in these tables are essentially all the values of n for which one might try to perform a polynomial multiplication in the setting of algebraic manipulation and still expect to receive an answer.

We can see that the degree homomorphism method is better than the classical for:

$$\begin{array}{lll} v=3 & \text{at} & n=7 \\ v=4 & \text{at} & n=4 \\ v=5 & \text{at} & n=4 \\ v=6 & \text{at} & n=3 \end{array}$$

The evaluation homomorphism method has a cross-over point wrt. classical multiplication about one degree lower than the degree homomorphism method. However, the evaluation method itself has a cross-over point wrt. the degree method about one degree above that of classical multiplication. So the evaluation method is the "best" algorithm only for a small range of degrees. The recursive FFT method is faster than the degree method at a few points, but because of the high steps in its computation time function it is not consistantly better.

4. Other Applications

The degree homomorphism can be applied to other problems. Since the transformation is a ring homomorphism, we can use this fact to apply it to other situations. Some of these are:

I. Powers of Polynomials

Fateman [2] discusses a variety of techniques to raise a polynomial to an integer power d. One method is to apply the FFT recursively as follows:

a) apply the transform until the polynomial $p(\bar{x})$ and all its coefficient polynomials are evaluated at the primitive roots of unity to give a sequence ot values $\{\hat{p}_i\}$.

b) raise each value $\{\hat{p}_i\}$ to the d-th power

c) apply the inverse transform to the powered values to obtain the powered polynomial in coefficient representation.

Again we have to append sufficient leading zero coefficients to the polynomial before the forward transform is applied in order to allow for the growth in degree.

Fateman shows that the recursive FFT method becomes better than other methods for larger degrees and numbers of variables. An alternative for the smaller degrees is:

1. Apply the degree homomorphism ϕ: $x_i \to x_{i-1}^{d\cdot n+1}$ to the multivariate polynomial $p(\bar{x})$ to obtain a univariate image $p'(x)$.

2. Raise $p'(x)$ to the power d using the standard FFT (or other) method to get the result $q'(x)$.

3. Apply the inverse degree homomorphism to $q'(x)$ to obtain its multivariate analogue $q(\bar{x})$.

The validity of this approach is guaranteed by Theorem 2 since in step 2 we are just performing a sequence of ring operations with a result in the isomorphic set. This time through, the leading zero coefficients would be appended by the degree homomorphism.

The degree of the result will be $d \cdot n_i$ where $n_i = \partial_i(p(\bar{x}))$ and therefore the degree homomorphism is:

$$x_i = x_{i-1}^{d \cdot n_i + 1}.$$

The method is asymptotically $O(v\, d\, n^{vd} \log n)$ as with the recursive FFT method. However the degree homomorphism method should take advantage of its larger degree to exploit the asymptotic properties of the FFT.

II. Exact Division

It we wanted to divide some polynomial $p(\bar{x})$ by another polynomial $p(\bar{x})$ exactly (i.e. with no remainder) again we could apply the Fourier transform recursively. Pollard [1] points out that the Fourier Transform method would be:

a) apply the FFT to $p(\bar{x})$ and $q(\bar{x})$ to give two sequences of values $\{\hat{p}_i\}$ and $\{\hat{q}_i\}$.

b) divide the sequences $\hat{r}_i = p_i/\hat{q}_i$.

c) apply the inverse FFT to the sequence $\{\hat{r}_i\}$ to obtain the quotient $r(\bar{x})$ in coefficient form.

Again the FFT method is faster than other more classical methods for degrees greater than the cross-over point. However we can also use the following degree homomorphism method:

1. Map the two polynomials $p(\bar{x})$ and $q(\bar{x})$ into their univariate images $p'(x)$ and $q'(x)$ using the map ϕ: $x_{i-1}^{n_i}$, where $n_i = \partial_i(p) + 1$, $1 < i \leqq v$.

2. Apply the FFT method to the univariate polynomials $p'(x)$ and $q'(x)$ to compute their quotient $r'(x)$.

3. Map $r'(x)$ back into its multivariate analogue $r(\bar{x})$.

Since $p'(x)$ and $q'(x)$ are polynomials of relatively high degree, the constant of proportionality in the computing time for the method will be amortised over a larger problem so that this approach should be quite competitive with the recursive FFT method and other classical methods. The validity of this approach is shown by:

Theorem 6. For this method $p(\bar{x}) = r(\bar{x})$ (i.e. $r(\bar{x})$ is the desired quotient).

Proof. By construction $p'(x) = r'(x) \cdot q'(x)$ and by Corollary 2: $\partial_i(r) = \partial_i(p) - \partial_i(q)$. Therefore by Theorem 3: $\psi(r'(x)) = \psi(\phi(r(\bar{x}))) = r(\bar{x})$ under the map ϕ: $x_i \to x_{i-1}^{n_i}$, where $n_i = \partial_i(p) + 1$.

We have assumed here that $q(\bar{x})$ is known to divide $p(\bar{x})$ but we could also use this method as a "divide if divisible" test by incorporating extra steps. We would check that $\partial_i(p) \geqq \partial_i(q)$ when applying the map and that $\partial_i(p) = \partial_i(q) + \partial_i(r)$ when inverting it. If any of these tests fail, then we would know exact divisibility is impossible; on the other hand, if they all succeed, then we would know we have a correct quotient.

III. Division with Remainder

Another potential application of the degree homomorphism, is to the problem of polynomial division where a remainder is produced. Strassen [3] shows how to use a fast polynomial multiplication algorithm to obtain an $O(N \log N)$ univariate complete polynomial division algorithm. However, there are several problems to be overcome:

a) Polynomial division requires exact divisibility in the coefficient domain and this may not hold in general.

b) Even if it is known that a division is possible, bounding the degrees of the quotient and remainder for the degree homomorphism is a difficult operation.

Brown and Traub [11] show that for the division of $S(\bar{x})$ by $T(\bar{x})$ where S, $T \in \mathscr{F}[x]$, and Q, R are the quotient and remainder respectively

$$S = QT + R,$$

$$\partial_v(S) < \partial_v(T) < \partial_v(R).$$

Then the degrees of the coefficients of the remainder in the subsidiary variables $\{x_i\}$, $1 \leqq i < v$ will be bounded by:

$$\partial_i(R) \geqq 2m \cdot \max(\partial_i(S), \quad \partial_i(T))$$

where:

$$m = 1/2(\partial_v(S) + \partial_v(T) - 2(\partial_v(R) - 1)).$$

Since it is difficult to bound $\partial_v(R)$ a-priori we could choose the very gross bound

$$\partial_v(R) = 0$$

and get

$$m \geqq 1/2(\partial_v(S) + \partial_v(T) + 2).$$

If

$$\partial_i(S),\ \partial_i(T) \leqq n, \quad 1 \leqq i \leqq v.$$

then

$$\partial_i(R) \geqq 2n(n+1).$$

We could use this as a bound in the degree homomorphism to convert a multivariate polynomial division problem to a univariate division problem and obtain an algorithm with an $O(2v(2n+1)^{2v} \log(2n+1))$ time bound. This might be advantageus when compared to a classical worst case $O((2n+1)^{4v})$, however it has been observed in practice that most division problems do not exhibit this catastrophic growth of coefficient degrees.

If it is not known that a multivariate division is possible then a process known as pseudo-division can be used. This involves dividing T into $S \cdot \mathscr{L}(T)^k$, where $\mathscr{L}(T)$ is the coefficient of the highest degree term in T and $k = \partial_v(S) - \partial_v(T) + 1$. This altered form of division guarantees exact divisibility of the coefficients and therefore a unique quotient and remainder.

IV. Polynomial Greatest Common Divisors (GCDs)

There has been considerable interest shown in the efficiency of algorithms for multivariate polynomial GCDs. In spite of the great strides made by Brown [12] with a recursive evaluation homomorphism algorithm, current algorithms can only compute the GCDs of relatively small polynomials.

In Moenck [4] it was shown that the GCD of two univariate polynomials of degree N could be computed in $O(N \log^2 N)$ operations (cf. $O(N^2)$ for the classical euclidean algorithm). There it was advocated that a variant of Brown's algorithm be used for the multivariate case to obtain an algorithm with a time of $O(v N^v \log^2 N)$. However, using this method, the cross-over point (already high in the univariate case) compared to Brown's method, would be prohibitively large.

An alternate approach is to apply a degree homomorphism method as follows:

1. Map the two polynomials $p(\bar{x})$ and $q(\bar{x})$ into their images $p'(x)$, $q'(x)$ by the map

$$\phi: x_i \to x_{i-1}^{n_i}, \qquad \text{where } n_i = \max(\partial_i(p), \partial_i(q)).$$

2. Compute the univariate GCD $g'(x)$ of $p'(x)$ and $q'(x)$ using the $O(N \log^2 N)$ algorithm.

3. Map the GCD $g'(x)$ back to its multivariate image $g(\bar{x})$.

This would give a method with an operation count of $O(v^2 N^v \log^2 N)$ where $\partial_i(p, q) \leqq n$, $1 \leqq i \leqq v$. However if v is small compared to the other parameters it might offer a considerable improvement over existing methods.

The validity of this method is shown by:

Theorem 7. $g(\bar{x})$ is the GCD of $p(\bar{x})$ and $q(\bar{x})$.

Proof. By construction $g'(x)$ is GCD of $p'(x)$ and $q'(x)$. Therefore $g'(x) \mid p'(x)$ and $g'(x) \mid q'(x)$ and by the isomorphism given by Theorem 3: $\psi(g'(\bar{x})) = \psi(\phi(g(\bar{x}))) = g(\bar{x})$ and therefore $g(\bar{x}) \mid p(\bar{x})$ and $g(\bar{x}) \mid q(\bar{x})$. This means that $g(\bar{x})$ is a common divisor of p and q. Suppose it is not the greatest one but instead that some polynomial $f(x)$ is the GCD of p and q. This would imply that:

$p(\bar{x}) = u(\bar{x}) f(\bar{x})$ and $q(\bar{x}) = v(\bar{x}) d(\bar{x})$ for some $u, v \in \mathscr{R}[\bar{x}]$ and
$g(\bar{x}) \mid f(\bar{x})$ (properly).

Then under the degree mapping:

$$p'(x) = u'(x)\, d'(x), \qquad q'(x) = v'(x)\, f'(x)$$
$$\text{and } g'(x) \mid f'(x) \text{ (properly).}$$

Therefore $f'(x)$ would be a common divisor of $p'(x)$ and $q'(x)$ of greater degree than $g'(x)$ contradicting the construction. □

Again the advantage of a degree homomorphism method is that it maps a small dense multivariate problem into a large dense univariate problem which may be more efficiently computed by a "fast" algorithm. Also in this case, the concern about intermediate coefficient growth and unlucky homomorphisms (cf. Brown [12]) can be put aside because of the theorem above.

V. Factoring Polynomials

In Moenck [6] it is shown that a univariate polynomial over a finite field can be factored into its irreducible components in $O(N^2 \log^4 N)$ steps. Corollary 2 gives a bound on the degrees of the factors of a polynomial $p(\bar{x})$ of $\partial_i(p)$.

Thus we can apply the degree homomorphism:

$$\phi: x_i \to x_{i+1}^{\partial_i(p)+1}$$

to obtain a univariate polynomial $p'(x_v)$. We can then use the efficient factoring method to obtain its irreducible factors. These factors will then be isomorphic to the original multivariate factors of $p(\bar{x})$ (by the isomorphism shown in Theorem 3).

This method is more of a curiousity than a practical method since it yields an algorithm with a bound of $O(v^4 n^{2v} \log^4 n)$ which is inferior to the techniques based on Hensel's lemma (cf. Yun [13]) which perform as $O(n^{2v})$, if all goes well.

5. Summary and Conclusions

We have seen that a viable alternative to the conventional approach to multivariate polynomial problems, of applying univariate algorithms recursively, is to apply a degree homomorphism:

$$\phi: x_i \to x_{i+1}^{m_i}, \quad \text{in } \mathscr{R}[x].$$

This approach leads to algorithms for a variety of multivariate polynomial problems which have lower cross-over points than, say, methods based on recursive application of the FFT. It would seem that this approach would be a reasonable one for intermediate sized problems in the field of algebraic manipulation. The areas which should be explored further are:

I. Other problems to which this method could be applied. Those that spring to mind immediately are linear algebra and truncated power series operations.

II. Deriving tight bounds on the degrees of polynomial results and intermediate values.

Finally, it should be noted that the notion of a degree homomorphism is not very new. As a mathematical devise it dates back at least to Kronecker. In the setting of algebraic manipulation it is just an alternative way to view the packing of exponents which occurs when a polynomial is represented in expanded canonical form. Several algebraic manipulation systems use this representation, the most notable of which is Altran (cf. Hall [14]). Williams [15] describes an early use of this representation.

However, the exponent packing used in such algebraic manipulation systems is different in that the same packing for a polynomial is used throughout a program. Also two polynomials must be "packed" in the same way before an operation is performed on them. What the degree homomorphism implies, is that an optimum packing be chosen for each operation and a polynomial be suitably "repacked" if necessary.

Achnowledgement. The author would like to thank the referee for his helpful comments.

References

1. Pollard, J. M.: The fast Fourier transform in a finite field. Mathematics of Computation **25**, 365–374 (1971)
2. Fateman, R. J.: Polynomial multiplication, powers and asymptotic analysis: Some comments. SIAM J. of Computing **3**, 196–213 (1974)
3. Strassen, V.: Die Berechnungskomplexität von elementar symetrischen Funktionen und von Interpolationskoeffizienten. Numer. Math. **20**, 238–251 (1973)
4. Moenck, R.: Studies in fast algebraic algorithms. University of Toronto, Ph. D. Thesis, Sept 1973
5. Borodin, A., Moenck, R.: Fast modular transforms. J. Computer and System Sciences **18**, 366–387 (1974)
6. Moenck, R.: On the efficiency of algorithms for polynomial factoring. To appear in Mathematics of Computation
7. Bonneau, R. J.: Fast polynomial operations using the fast Fourier transform. Pre-print MIT 1973
8. Lipson, J. D.: Chinese remainder and interpolation algorithms. In: Petrick, S.R. (ed.): Proc. 2nd Symp. on Symbolic and Algebraic Manipulation. New York: ACM 1971, p. 188–194
9. Horowitz, E.: Modular arithmetic and finite field theory: A tutorial. In: Petrick, S. R. (ed.): Proc. 2nd Symp. on Symbolic and Algebraic Manipulation. New York: ACM 1971, p. 188–194
10. Lang, S.: Algebra. Reading (Mass.): Addison-Wesley 1971
11. Brown, W. S., Traub, J. F.: On Euclid's algorithm and the theory of subresultants. J. ACM **18**, 505–514 (1971)
12. Brown, W. S.: On Euclid's algorithm and the computation of polynomial greatest common divisors. J. ACM **18**, 478–504 (1971)
13. Yun, D. Y. Y.: The Hensel lemma in algebraic manipulation. MIT, Cambridge (Mass.), Ph. D. Thesis, Nov 1974
14. Hall, A. D.: The Altran system for rational function manipulation—A survey. Comm. ACM **14**, 517–521 (1971)
15. Williams, L. H.: Algebra of polynomials in several variables for a digital computer. J. ACM **9**, 29–40 (1962)

Robert T. Moenck
Department of Computer Science
University of Waterloo
Waterloo, Ontario
Canada