Computing with Polynomials Given by Straight-Line Programs II
Sparse Factorization*

Erich Kaltofen

Rensselaer Polytechnic Institute
Department of Computer Science
Troy, New York 12181

Abstract

We develop an algorithm for the factorization of a multivariate polynomial represented by a straight-line program into its irreducible factors represented as sparse polynomials. Our algorithm is in random polynomial-time for the usual coefficient fields and outputs with controllably high probability the correct factorization. It only requires an a priori bound for the total degree of the input and over rational numbers a bound on the size of the polynomial coefficients.

Keywords. Polynomial factorization, sparse polynomial, straight-line program, random polynomial-time.

1. Introduction

In [1] we have started developing a theory for manipulating polynomials given by straight-line programs. There we have shown that the GCD of multivariate polynomials given by straight-line programs can be found again in terms of a straight-line program in probabilistic polynomial-time as a function of the input size and the total degree of the inputs. We have also presented a probabilistic polynomial-time solution for converting a polynomial given by a straight-line program into its sparse representation. Here we continue this theory by presenting a probabilistic polynomial-time procedure for factoring a polynomial given by a straight-line program into its sparse irreducible factors.

Our model of computation is that of a sequential algebraic RAM over a field as introduced in [2]. Our algorithm calls a bivariate polynomial factorization procedure and is

* This material is based upon work supported by the National Science Foundation under Grant No. DCR-85-04391 and by an IBM Faculty Development Award. Part of this work was done while the author was visiting the Tektronix Computer Research Laboratory in Beaverton, Oregon. An extended abstract appeared in Proc. 26th IEEE Symp. Foundations Comp. Sci., 451-458 (1985).

[1] E. Kaltofen, ‘‘Computing with polynomials given by straight-line programs I; Greatest common divisors,’’ *Proc. 17th ACM Symp. Theory Comp.*, pp. 131-142, 1985.

[2] E. Kaltofen, ‘‘Computing with polynomials given by straight-line programs I; Greatest common divisors,’’ *Proc. 17th ACM Symp. Theory Comp.*, pp. 131-142, 1985.

therefore only effective and of polynomial running time for the usual coefficient fields. Over the rationals, for instance, we get an algorithm of expected binary complexity that is a polynomial function in the binary size of the straight-line program computing the input polynomial, in a priori bounds for its total degree and for the size of the numerators and the common denominator of its rational coefficients, and in the number of monomials occurring in its sparse irreducible factors. The algorithm with controllably high probability outputs the correct sparse factorization. Polynomial-time factorization of rational bivariate polynomials is guaranteed by the results in [3] and [4]. Let us illustrate the power of our algorithm on a classical example, that of the van der Monde determinant

$$\det \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{bmatrix} = \prod_{i>j} (x_i - x_j).$$

Our algorithm can with high probability find the right-hand side factorization in $n^{O(1)}$ expected steps even though the product consists of $n!$ monomials. This remains true, of course, if we randomly perturb the input by multiplying the matrix with random unimodular matrices from the left and right and thus destroy any of its symmetries.

Multivariate polynomial factorization has been studied extensively in the past twenty years. Musser [5] not only for the first time applied Zassenhaus's [6] approach of Hensel lifting to the multivariate problem but he also realized that the algorithm could still take exponential running time. In [7] we showed that for multivariate polynomials in dense representation the problem is polynomial-time reducible to univariate factorization and therefore by [8] can be solved in polynomial-time. However, the dense representation seems to be useful only if the input polynomial has few variables. Already Zippel [9] investigated how the sparsity of input and output could be probabilistically preserved during the multivariate Hensel lifting algorithm. However, the problem of factoring a sparse multivariate polynomial

---

[3] A. K. Lenstra, H. W. Lenstra Jr., and L. Lovász, ''Factoring polynomials with rational coefficients,'' *Math. Ann.*, vol. 261, pp. 515-534, 1982.

[4] E. Kaltofen, ''Polynomial-time reductions from multivariate to bi- and univariate integral polynomial factorization,'' *SIAM J. Comp.*, vol. 14, pp. 469-489, 1985.

[5] D. R. Musser, ''Multivariate polynomial factorization,'' *J. ACM*, vol. 22, pp. 291-308, 1975.

[6] H. Zassenhaus, ''On Hensel factorization I,'' *J. Number Theory*, vol. 1, pp. 291-311, 1969.

[7] E. Kaltofen, ''Polynomial-time reductions from multivariate to bi- and univariate integral polynomial factorization,'' *SIAM J. Comp.*, vol. 14, pp. 469-489, 1985.

[8] A. K. Lenstra, H. W. Lenstra Jr., and L. Lovász, ''Factoring polynomials with rational coefficients,'' *Math. Ann.*, vol. 261, pp. 515-534, 1982.

[9] R. E. Zippel, ''Newton's iteration and the sparse Hensel algorithm,'' *Proc. '81 ACM Symp. Symbolic Algebraic Comp.*, pp. 68-72, 1981.

into its sparse irreducible factors in expected polynomial-time could not be solved before fairly deep effective versions of the Hilbert irreducibility theorem were established, cf. [10] and [11]. The theorem in the latter reference has a better probability of success and and simpler substitutions therefore will be used here. We refer to [12] for a sparse factoring algorithm that is polynomial-time provided the number of irreducible factors is small. In this paper we remove this provision and the necessity that the polynomial to be factored be also sparse.

Although our algorithm borrows the application of the effective Hilbert irreducibility theorem and the idea that resolves the so-called leading coefficient problem from [13], we must employ a very different sparse lifting method. The key idea, already described in [14], is to lift bivariate polynomials only and to make the coefficients rational functions over the remaining variables. It has been observed [15] that if one would carry this approach through by representing those rational functions as sparse polynomials then the intermediate results become very dense prohibiting efficient or polynomial-time solutions. However, our theory allows us to represent those coefficients as straight-line programs of small size. Two algorithms from [16] become crucial for successfully carrying out the lifting, namely the algorithm for determining a straight-line program for the coefficients of a single variable and that for converting a polynomial given by a straight-line program to its sparse representation. We note that our algorithm also provides an alternate solution to the problem of factoring a sparse input because the sparse representation of a polynomial always can be converted into a straight-line program of polynomial size.

Notation: We use the same notation as in [17] but for the convenience of the reader we shall repeat it here. By **Q** we denote the the rational numbers and by GF(q) the finite field with q elements. F usually denotes a field and char(F) its characteristic. The coefficient of the highest power of x_1 in $f \in (F[x_2, \ldots, x_n])[x_1]$ is referred to as the leading coefficient of

---

[10] J. von zur Gathen, ''Irreducibility of multivariate polynomials,'' *J. Comp. System Sci.*, vol. 31, pp. 225-264, 1985.

[11] E. Kaltofen, ''Effective Hilbert irreducibility,'' *Information and Control*, vol. 66, pp. 123-137, 1985.

[12] J. von zur Gathen and E. Kaltofen, ''Factoring sparse multivariate polynomials,'' *J. Comp. System Sci.*, vol. 31, pp. 265-287, 1985.

[13] J. von zur Gathen and E. Kaltofen, ''Factoring sparse multivariate polynomials,'' *J. Comp. System Sci.*, vol. 31, pp. 265-287, 1985.

[14] D. R. Musser, ''Multivariate polynomial factorization,'' *J. ACM*, vol. 22, pp. 291-308, 1975.

[15] P. S. Wang and L. Rothschild, ''Factoring multivariate polynomials over the integers,'' *Math. Comp.*, vol. 29, pp. 935-950, 1975.

[16] E. Kaltofen, ''Computing with polynomials given by straight-line programs I; Greatest common divisors,'' *Proc. 17th ACM Symp. Theory Comp.*, pp. 131-142, 1985.

[17] E. Kaltofen, ''Computing with polynomials given by straight-line programs I; Greatest common divisors,'' *Proc. 17th ACM Symp. Theory Comp.*, pp. 131-142, 1985.

f in x_1, $\mathrm{ldcf}_{x_1}(f)$, the content of f with respect to x_1 is denoted by $\mathrm{cont}_{x_1}(f)$. Two polynomials f_1 and f_2 are associates, $f_1 \sim f_2$, if $f_1 = cf_2$ with $c \in F \setminus \{0\}$. The number of monomials of $f \in F[x_1, \ldots, x_n]$ in its sparse representation is denoted by mon(f). For $F = \mathbf{Q}$ the binary size of the monomial coefficients as fractions of integers with a common denominator, the combined coefficient size, is denoted by cc-size(f).

A straight-line program over a domain D is formally a quadruple $P = (X, V, C, S)$ where X is the set of inputs, V the set of program variables, C the computation sequence, and S the set of scalars occurring in the computation sequence. The length of C is the length of P, len(P). Each program variable v computes an element $f(v) \in D$, and $f(P) = \bigcup_{v \in V} f(v)$. A polynomial $f \in F[x_1, \ldots, x_n]$ is given by a straight-line program P if $P = (\{x_1, \ldots, x_n\}, V, C, S)$ over $F(x_1, \ldots, x_n)$ and $S \subset F$. The program P is defined at $\phi:\{x_1, \ldots, x_n\} \rightarrow F$ if no zero-division occurs when evaluating P at $\phi(x_m)$ in place of x_m. The element size of P, el-size(P), denotes the number of bits it takes to represent $X \cup S$.

By $M(d)$ we denote a function dominating the time for multiplying polynomials in $F[x]$ of maximum degree d. The cardinality of a set S is denoted by card(S). The vertical stroke $|$ denotes the divisibility relation.

2. Evaluation and Factorization Pattern

For several reasons it is crucial for our Sparse Factoring algorithm that the Hensel lifting is started with true factor images. Fortunately, the effective versions of the Hilbert irreducibility theorem [18] and [19] make it possible to probabilistically enforce this assumption. In this section we present a theorem on the probabilities that certain evaluations preserve the *factorization pattern* that determines the number of irreducible factors, their multiplicities, and their total degrees. The argument is essentially the same as that in [20], Theorem 3.6, but with our effective version of the Hilbert irreducibility theorem. The main advantage of this change is that the evaluations are simpler and the probability of success is higher.

Theorem 2.1: Let $f \in F[x_1, \ldots, x_n]$, F a perfect field, $d = \deg(f)$, $R \subset F$. The factorization pattern of f is a lexicographically ordered vector $((d_i, e_i))_{i=1,\ldots,r}$ such that for $f = \prod_{i=1}^{r} h_i^{e_i}$, $h_i \in F[x_1, \ldots, x_n]$,

$$h_i \text{ irreducible}, \quad d_i = \deg(h_i) \geq 1,\ 1 \leq i \leq r,\quad h_i \nmid h_j \text{ for } i \neq j.$$

Let $a_1, a_3, \ldots, a_n, b_3, \ldots, b_n \in R$ be randomly selected elements,

---

[18] J. von zur Gathen, ''Irreducibility of multivariate polynomials,'' *J. Comp. System Sci.*, vol. 31, pp. 225-264, 1985.

[19] E. Kaltofen, ''Effective Hilbert irreducibility,'' *Information and Control*, vol. 66, pp. 123-137, 1985.

[20] J. von zur Gathen, ''Irreducibility of multivariate polynomials,'' *J. Comp. System Sci.*, vol. 31, pp. 225-264, 1985.

$$f_1 = f(x_1 + a_1,\ x_2,\ b_3 x_1 + a_3, \ldots,\ b_n x_1 + a_n).$$

Then

$$\mathrm{Prob}(f \text{ and } f_1 \text{ have the same factorization pattern}) \ge 1 - \frac{4d\,2^d + d^3}{\mathrm{card}(R)}.$$

Proof: By [21], theorem 3,

$$h_{i1} = h_i(x_1 + a_1,\ x_2,\ b_3 x_1 + a_3, \ldots,\ b_n x_1 + a_n)$$

remains irreducible in $F[x_1, x_2]$ with probability $\ge 1 - 4d_i 2^{d_i}/\mathrm{card}(R)$. It remains to estimate the probability that $\deg(h_{i1}) = d_i$ and that $h_{i1} \nmid h_{j1}$ for all $i \ne j$. Let

$$\hat{h}_i = h_i(x_1 + \alpha_1,\ x_2,\ \beta_3 x_1 + \alpha_3, \ldots,\ \beta_n x_1 + \alpha_n)$$

in $F(\alpha_1, \alpha_3, \ldots, \alpha_n, \beta_3, \ldots, \beta_n)[x_1, x_2]$. Clearly, $\deg_{x_1, x_2}(\hat{h}_i) = d_i$. Let

$$\pi_i(\beta_3, \ldots, \beta_n) \in F[\beta_3, \ldots, \beta_n] \setminus \{0\}$$

be the coefficient of a monomial $x_1^{j_1} x_2^{j_2}$, $j_1 + j_2 = d_i$, in $\hat{h}_i$. Now $\deg(\pi_i) \le d_i$ and

$$\pi_i(b_3, \ldots, b_n) \ne 0 \quad \text{implies} \quad \deg(h_{i1}) = d_i.$$

By [22], Lemma 1, this happens with probability $\ge$

$$1 - \frac{\deg(\pi_i)}{\mathrm{card}(R)} \ge 1 - \frac{d_i}{\mathrm{card}(R)}.$$

We finally estimate the chance that $h_{i1} \nmid h_{j1}$ by completing an argument made in [23], Theorem 4.5. First we note that $\hat{h}_i \nmid \hat{h}_j$ in $F(\alpha_1, \alpha_3, \ldots, \alpha_n, \beta_3, \ldots, \beta_n)[x_1, x_2]$, $i \ne j$, because both polynomials are irreducible and not associated in $F[\alpha_1, \alpha_3, \ldots, \alpha_n, \beta_3, \ldots, \beta_n, x_1, x_2]$. Irreducibility of $\hat{h}_i$, say, follows because any factorization of it could be back-translated by $\alpha_k \leftarrow x_k - \beta_k x_1$, $3 \le k \le n$ and then $x_1 \leftarrow x_1 - \alpha_1$ into a factorization of h_i. We also note that each of $\hat{h}_i$ and $\hat{h}_j$ has at least two non-zero monomials. Therefore there exist two monomials $\sigma_{i1}, \sigma_{i2} \in F[\alpha_1, \alpha_3, \ldots, \alpha_n, \beta_3, \ldots, \beta_n]$ of $\hat{h}_i$ and two monomials σ_{j1}, σ_{j2} of $\hat{h}_j$ such that

$$\tau_{ij} = \sigma_{i1}\sigma_{j1} - \sigma_{i2}\sigma_{j2} \ne 0.$$

Now

$$\tau_{ij}(a_1, a_3, \ldots, a_n, b_3, \ldots, b_n) \ne 0 \quad \text{implies} \quad h_{i1} \nmid h_{j1}.$$

---

[21] E. Kaltofen, ‘‘Effective Hilbert irreducibility,’’ *Information and Control*, vol. 66, pp. 123-137, 1985.

[22] J. T. Schwartz, ‘‘Fast probabilistic algorithms for verification of polynomial identities,’’ *J. ACM*, vol. 27, pp. 701-717, 1980.

[23] J. von zur Gathen, ‘‘Irreducibility of multivariate polynomials,’’ *J. Comp. System Sci.*, vol. 31, pp. 225-264, 1985.

Since $\deg(\tau_{ij}) \le d_i + d_j$ the probability of this event is $\ge 1 - (d_i + d_j)/\mathrm{card}(R)$. Overall, the factorization pattern is preserved with probability not less than

$$1 - \left[\sum_{i=1}^{r} \frac{4d_i 2^{d_i}}{\mathrm{card}(R)} + \sum_{i=1}^{r} \frac{d_i}{\mathrm{card}(R)} + \sum_{1 \le i < j \le r} \frac{d_i + d_j}{\mathrm{card}(R)} \right]$$

$$\ge 1 - \left[\frac{4d\ 2^d}{\mathrm{card}(R)} + \frac{d}{\mathrm{card}(R)} + \frac{d(d-1)}{2}\ \frac{d}{\mathrm{card}(R)} \right] \ge 1 - \frac{4d\ 2^d + d^3}{\mathrm{card}(R)}. \quad \square$$

It is clear from the above theorem that we can probabilistically obtain the factorization pattern of a polynomial given by a straight-line program by evaluation. The assumption that the field is perfect can be dropped at the cost of increasing the failure probability somewhat (cf. [24], Lemma 3.2), but since the usual coefficient fields are perfect we do not incorporate this generalization. A more important observation is that with high probability the factor degrees in x_2 are preserved as well. We define as the *extended factorization pattern* of f the lexicographically ordered vector

$$((d_i, e_i, d_{i,1}, \ldots, d_{i,n}))_{i=1,\ldots,r}$$

where d_i and e_i are as in the factorization pattern and $d_{i,m} = \deg_{x_m}(h_i)$. By letting each variable in turn take the role of x_2 in the theorem and mapping the obtained bivariate factors down to

$$f(b_1 x_1 + a_1, b_2 x_1 + a_2, \ldots, b_n x_1 + a_n)$$

we can deduce the extended factorization pattern of f given by a straight-line program in probabilistic polynomial-time. The relevant theorem follows now.

Theorem 2.2: Let $f = \prod_{i=1}^{r} h_i^{e_i} \in F[x_1, \ldots, x_n]$, F a perfect field, $d = \deg(f)$, $R \subset F \setminus \{0\}$, and let $a_m, b_m \in R$, $1 \le m \le n$, be randomly selected elements. We define

$$f^{(m)} = f(b_1 x_0 + a_1, \ldots, b_{m-1} x_0 + a_{m-1}, x_m, b_{m+1} x_0 + a_{m+1}, \ldots, b_n x_0 + a_n),\ 1 \le m \le n,$$

and

$$f^{(m)}(x_0, x_m) = \prod_{i=1}^{r_m} \left[h_i^{(m)}(x_0, x_m) \right]^{e_{i,m}}$$

its factorization. Furthermore, we set

$$h_i^{(0)} = h_i(b_1 x_0 + a_1, \ldots, b_n x_0 + a_n),\ 1 \le i \le r.$$

Then the probability that

[24] J. von zur Gathen, ‘‘Irreducibility of multivariate polynomials,’’ *J. Comp. System Sci.*, vol. 31, pp. 225-264, 1985.

i) the polynomials f and $f^{(m)}$, $1 \le m \le n$, have the same factorization pattern,

ii) $\deg_{x_m}(h_i) = \deg_{x_m}(h_i^{(m)})$, $1 \le m \le n$, $1 \le i \le r$,

iii) $h_i^{(0)} \nmid h_j^{(0)}$ for $1 \le i < j \le r$,

is not less than

$$1 - \frac{4nd\ 2^d + (n+1)d^3 + nd}{\mathrm{card}(R)}.$$

Proof: Condition i) is probabilistically enforced essentially by theorem 2.1. We apply it to $f^{(m)}(b_1 x_0, x_2, \ldots, x_n)$ with x_0, x_m, a_1/b_1 taking the place of x_1, x_2, and a_1, respectively. For $m = 1$ we replace x_2 by $b_2 x_0$. This means that a_1 of theorem 2.1 is chosen from a different set, but since Lemma 1 of [25] can be trivially generalized to choosing the values for each variable from different sets, all probability estimates for theorem 3 of [26] remain the same. Therefore, the probability that i) is violated is no more than $n(4d\ 2^d + d^3)/\mathrm{card}(R)$.

Conditions ii) and iii) are proven as for theorem 2.1. In other words, there exist polynomials $\pi_i^{(m)} \in F[\beta_1, \ldots, \beta_{m-1}, \beta_{m+1}, \ldots, \beta_n]$ and $\tau_{ij} \in F[\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n]$ with

$$\deg(\pi_i^{(m)}) \le \deg(h_i^{(m)}) \text{ and } \pi_i^{(m)}(b_1, \ldots, b_n) \ne 0 \text{ implies } \deg_{x_m}(h_i) = \deg_{x_m}(h_i^{(m)})$$

and

$$\deg(\tau_{ij}) \le \deg(h_i) + \deg(h_j) \text{ and } \tau_{ij}(a_1, \ldots, b_n) \ne 0 \text{ implies } h_i^{(0)} \nmid h_j^{(0)}.$$

Thus, condition ii) is not met with probability $\le nd/\mathrm{card}(R)$ and condition iii) with probability $\le d^3/(2\mathrm{card}(R))$. Adding up the failure probabilities now gives the theorem. □

It should be clear now how to determine the extended factorization pattern. First we factor all $f^{(m)}$. Then we compute

$$h_i^{(m)}(x_0, b_m x_0 + a_m).$$

By iii) with high probability those polynomials that are associates are images of the same irreducible factor of f, and by ii) we now know its degrees in x_m. One can accomplish the task of factoring all $f^{(m)}$, $m \ne 2$, efficiently by separately lifting the factorization

$$\prod_{i=1}^{r} h_i^{(2)}(x_0, b_2 x_0 + a_2) = f^{(2)}(x_0, b_2 x_0 + a_2)$$

with respect to the variables $x_1, x_3, \ldots, x_n$, see §3, algorithm One-Variable Lifting. We mention that von zur Gathen [27], Remark 5.6, has later shown how to obtain the extended

[25] J. T. Schwartz, ‘‘Fast probabilistic algorithms for verification of polynomial identities,’’ *J. ACM*, vol. 27, pp. 701-717, 1980.

[26] E. Kaltofen, ‘‘Effective Hilbert irreducibility,’’ *Information and Control*, vol. 66, pp. 123-137, 1985.

[27] J. von zur Gathen, ‘‘Irreducibility of multivariate polynomials,’’ *J. Comp. System Sci.*, vol. 31, pp. 225-264,

factorization pattern from his version of the Hilbert irreducibility theorem. However, that solution requires to factor trivariate polynomials and is therefore computationally less efficient than ours.

3. Sparse Lifting of Polynomials in Straight-Line Representation

We now present the algorithm for lifting the last variable in the sparse factorization of a polynomial given by a straight-line program. The algorithm is derived from a bivariate version of the Hensel lemma (cf. [28], Lemma 2.1) together with the ability to compute the coefficients of individual variables as well as the solution of linear systems in terms of straight-line programs, and the ability to convert the resulting straight-line programs to sparse representation [29]. In order for the process to work we must make several assumptions about the leading multivariate coefficients and the factor images. The sparse factorization algorithm in §4 probabilistically enforces all these conditions before calling the lifting algorithm repeatedly with new variables.

Algorithm *One-Variable Lifting*

Input: $f \in F[x_1, \ldots, x_m]$ given by a straight-line program P that is defined at $\phi(x_1) = \phi(x_m) = 0$, a bound $d \geq \deg_{x_m}(f)$, a failure probability $\varepsilon \ll 1$, and $g_i \in F[x_1, \ldots, x_{m-1}] \setminus F[x_2, \ldots, x_{m-1}]$ in sparse representation, $e_i \geq 1$ with $\mathrm{char}(F) \nmid e_i$, $1 \leq i \leq r$, such that

i) $\mathrm{ldcf}_{x_1}(f) \in F[x_2, \ldots, x_{m-1}]$, $f(x_1, \ldots, x_{m-1}, 0) = \prod_{i=1}^{r} g_i^{e_i}$,

ii) $\mathrm{GCD}(g_i, g_j) = 1$ for $1 \leq i < j \leq r$,

iii) there exist $h_i \in F[x_1, \ldots, x_m]$ with $h_i(x_1, \ldots, x_{m-1}, 0) = g_i$, $1 \leq i \leq r$, and $f = \prod_{i=1}^{r} h_i^{e_i}$.

Notice that by ii) and iii) the h_i are uniquely determined. Also by i) $\mathrm{ldcf}_{x_1}(h_i) = \mathrm{ldcf}_{x_1}(g_i)$, $1 \leq i \leq r$.

Output: Either ''failure'' (that with probability $< \varepsilon$) or r sparse polynomials in $F[x_1, \ldots, x_m]$ that with probability $1 - \varepsilon$ are equal to $h_1, \ldots, h_r$.

FOR $k \leftarrow 1, 2, \cdots$ WHILE($d > \sum_{i=1}^{r} e_i \deg_{x_m}(h_i)$) DO step L.
Then return $h_i \leftarrow g_i^{(k)}$.

1985.

[28] E. Kaltofen, ''Sparse Hensel lifting,'' *Proc. EUROCAL '85, Vol. 2, Springer Lec. Notes Comp. Sci.*, vol. 204, pp. 4-17, 1985.

[29] E. Kaltofen, ''Computing with polynomials given by straight-line programs I; Greatest common divisors,'' *Proc. 17th ACM Symp. Theory Comp.*, pp. 131-142, 1985.

Step L (Lift): At this point we have $g_i^{(k)} \in F[x_1, \ldots, x_m]$, $1 \le i \le r$, in sparse representation, such that with high probability

$$g_i^{(k)} \equiv h_i \bmod x_m^k, \quad \deg_{x_m}(g_i^{(k)}) < k, \quad 1 \le i \le r. \tag{†}$$

In this step we determine $\hat{g}_i \in F[x_1, \ldots, x_{m-1}]$ in sparse representation, $\deg_{x_1}(\hat{g}_i) < \deg_{x_1}(g_i)$, $1 \le i \le r$, such that

$$g_i^{(k+1)} = g_i^{(k)} + \hat{g}_i x_m^k \equiv h_i \bmod x_m^{k+1}, \; 1 \le i \le r.$$

From iii) and (†) we get

$$g_1^{e_1-1} \cdots g_r^{e_r-1} \sum_{i=1}^{r} (e_i \hat{g}_i x_m^k \prod_{\substack{j=1 \\ j \ne i}}^{r} g_j) \equiv f - \prod_{i=1}^{r} (g_i^{(k)})^{e_i} \bmod x_m^{k+1} \tag{‡}$$

and

$$f - \prod_{i=1}^{r} (g_i^{(k)})^{e_i} \equiv \tau(x_1, \ldots, x_{m-1}) \; x_m^k \bmod x_m^{k+1}, \; \tau \in F[x_1, \ldots, x_{m-1}].$$

We can construct a straight-line program $P_\tau = (\{x_1, \ldots, x_{m-1}\}, V_\tau, C_\tau, S_\tau)$ with $\tau \in f(P_\tau)$ by applying the Polynomial Coefficients algorithm to a program P_k with $f - \prod_{i=1}^{r} (g_i^{(k)})^{e_i} \in f(P_k)$. Since P_k is defined at $\phi(x_m) = 0$, we can even use k as the degree bound needed in the Polynomial Coefficients algorithm. Notice that

$$\operatorname{len}(P_k) = O(\operatorname{len}(P) + \sum_{i=1}^{r} \log(e_i) \operatorname{mon}(g_i^{(k)}) \, m \log(d))$$

and that $\operatorname{len}(P_\tau) = O(M(k) \operatorname{len}(P_k))$. By iii), (†), (‡), and the bounds on the degree in x_1 we conclude that

$$\rho = \frac{\tau(x_1, \ldots, x_{m-1})}{g_1^{e_1-1} \cdots g_r^{e_r-1}} \in F[x_1, \ldots, x_{m-1}], \; \deg_{x_1}(\rho) < \deg_{x_1}(g_1 \cdots g_r).$$

Furthermore,

$$\frac{\rho}{g_1 \cdots g_r} = \frac{e_1 \hat{g}_1}{g_1} + \cdots + \frac{e_r \hat{g}_r}{g_r}, \tag{§}$$

and hence by ii) $\hat{g}_i / e_i$ form the partial fraction decomposition of $\rho/(g_1 \cdots g_r)$ in $F(x_2, \ldots, x_{m-1})[x_1]$.

The sparse representation for $\hat{g}_i$ now can be computed in several ways. One idea is to first compute the coefficients of

$$\hat{g}_i = \sum_{j=1}^{d_i} c_{i,j}(x_2, \ldots, x_{m-1}) \, x_1^{j-1}, \; d_i = \deg_{x_1}(g_i), \; 1 \le i \le r,$$

as straight-line programs and then convert those into sparse representation. A straight-line program for $c_{i,j}$ can be obtained by starting out with a straight-line program for the coefficients of

$$\rho = \sum_{j=1}^{d_1+\cdots+d_r} c_j(x_2,\ldots,x_{m-1})\, x_1^{j-1},$$

which can be obtained from P_τ, $g_1, \ldots, g_r$, and again by the Polynomial Coefficients algorithm. Notice that P_τ is defined at $\psi(x_1) = 0$ because P is assumed to be defined at ϕ. Now the $c_{i,j}$ form the unique solution of a $\sum d_i$ by $\sum d_i$ linear system, all of whose entries are given by straight-line programs. We can obtain straight-line programs for $c_{i,j}$ from Strassen's [30] division-free determinant program and by Cramer's rule. Finally, we perform the Sparse Conversion algorithm on the straight-line programs determining $c_{i,j}$ with failure probability ε/d^2. For $F = \mathbf{Q}$ we must also have a bound for cc-size(f) to carry out this last step. □

''Failure'' or an incorrect result can be produced in any conversion of $c_{i,j}$ to sparse representation. Clearly, for each k there are at most d such conversions and hence such events do not occur at all with probability $1 - \varepsilon$. We conclude this section by mentioning an alternate way of computing the sparse representation of $\hat{g}$. The idea is to apply the sparse interpolation algorithm (cf. [31], §6) directly to (§). This is possible because if the evaluations in the minor variables preserve pairwise relative primeness of the g_i then the partial fraction decomposition is unique. Note that the partial fraction decomposition can be computed quickly by the algorithm of [32].

4. Sparse Factorization of Polynomials in Straight-Line Representation

We now describe the factorization algorithm itself and analyze its total complexity. Since this algorithm calls Sparse Lifting, the input conditions to Sparse Lifting must be enforced. Conditions ii) and iii) can be probabilistically guaranteed by theorem 2.1. It is quite coincidental that the kind of evaluations used in that theorem also allow to guarantee condition iii), which is important for overcoming the so-called leading coefficients problem.

Algorithm *Sparse Factorization*

Input: $f \in F[x_1, \ldots, x_n]$ given by a straight-line program P of length l, a bound $d \geq \deg(f)$, and the allowed failure probability $\varepsilon \ll 1$.

Output: Either ''failure'' (that with probability $< 2\varepsilon$) or the sparse representation of polynomials $h_i \in F[x_1, \ldots, x_n]$, $e_i \geq 1$, $1 \leq i \leq r$, such that with probability $> 1 - \varepsilon$

---

[30] V. Strassen, ''Vermeidung von Divisionen,'' *J. reine u. angew. Math.*, vol. 264, pp. 182-202, 1973. (In German).

[31] E. Kaltofen, ''Computing with polynomials given by straight-line programs I; Greatest common divisors,'' *Proc. 17th ACM Symp. Theory Comp.*, pp. 131-142, 1985.

[32] H. T. Kung and D. M. Tong, ''Fast algorithms for partial fraction decomposition,'' *SIAM J. Comp.*, vol. 6, pp. 582-593, 1977.

$$f = \prod_{i=1}^{r} h_i^{e_i}$$

is the factorization of f into irreducible polynomials.

Step R (Random Points Selection): From a set $R \subset F$ with

$$\mathrm{card}(R) > \frac{\max(2^{l+4},\ 8d\, 2^d + 2d^3 + 2(n-2)d)}{\varepsilon}$$

select random elements $a_1, \ldots, a_n, b_3, \ldots, b_n$. If $F = \mathrm{GF}(q)$ with q small we can instead work over $\mathrm{GF}(q^p)$, p a prime integer $> d$. By Theorem 6.1 in [33] no additional factors occur.

Step I (Initialize for Lifting): Test whether P is defined at $\phi(x_m) = a_m$, $1 \le m \le n$. For $F = \mathbf{Q}$ we call algorithm Zero-Division Test in [34] so often that the probability of ''failure'' occurring all the times even if P were defined at ϕ is less than $\varepsilon/8$.

If P turns out to be (probably) undefined at ϕ we return ''failure''. Otherwise, P is definitely defined at ϕ and we compute the dense representation of

$$f_1 = f(x_1 + a_1,\ x_2,\ b_3 x_1 + a_3, \ldots, b_n x_1 + a_n).$$

This can be done similarly to the Sparse Conversion algorithm and we must again make the probability that ''failure'' occurs less than $\varepsilon/4$. If $F = \mathbf{Q}$, a bound for the cc-size(f) must be added to the input parameters.

Factor $f_1 = \prod_{i=1}^{r} g_{i,2}^{e_i}$, $g_{i,2} \in F[x_1, x_2]$ irreducible and pairwise not associated. Notice that with high probability f and f_1 have the same factorization pattern.

Step V (Variable by variable lifting): FOR $m \leftarrow 3, \ldots, n$ DO step L.

Step L (Lift one variable): At this point we have $g_{i,m-1} \in F[x_1, \ldots, x_{m-1}]$ such that with high probability

$$\prod_{i=1}^{r} g_{i,m-1}^{e_i} = f(x_1 + a_1,\ x_2, \ldots, x_{m-1},\ b_m x_1 + a_m, \ldots, b_n x_1 + a_n)$$

and

$$g_{i,m-1}(x_1,\ x_2,\ b_3 x_1 + a_3, \ldots, b_{m-1} x_1 + a_{m-1}) = g_{i,2},\ 1 \le i \le r.$$

Let $J \subset \{1, \ldots, r\}$ be such that $j \in J$ if and only if $g_{j,m-1} \in F[x_2, \ldots, x_{m-1}]$. For

[33] J. von zur Gathen, ''Irreducibility of multivariate polynomials,'' *J. Comp. System Sci.*, vol. 31, pp. 225-264, 1985.

[34] E. Kaltofen, ''Computing with polynomials given by straight-line programs I; Greatest common divisors,'' *Proc. 17th ACM Symp. Theory Comp.*, pp. 131-142, 1985.

$$\begin{aligned}&\bar{f_m}(x_1,\ldots,x_{m-1},y)\\&= f(x_1+a_1, x_2,\ldots,x_{m-1}, b_m x_1+a_m+y, b_{m+1}x_1+a_{m+1},\ldots,b_n x_1+a_n)\end{aligned}$$

construct a straight-line program $P_m = (\{x_1, \ldots, x_{m-1}, y\}, V_m, C_m, S_m)$ over $F(x_1, \ldots, x_{m-1}, y)$ that computes

$$f_m = \frac{\bar{f_m}}{\prod_{j\in J} g_{j,m-1}^{e_j}}.$$

We will show below that with high probability

$$f_m \in F[x_1,\ldots,x_{m-1},y] \ \text{ and } \ \mathrm{ldcf}_{x_1}(f_m) \in F[x_2,\ldots,x_{m-1}]. \qquad (*)$$

Call the One-Variable Lifting algorithm with $f_m \in f(P_m)$, $d - \sum_{j\in J} e_j \deg(g_{j,m-1})$, failure probability $\varepsilon/(2n-4)$, and $g_{i,m-1}$, e_i for $i \in \{1, \ldots, r\} \setminus J$.

If $p = \mathrm{char}(F)$ divides one of the e_i, the algorithm has to be slightly modified. We then lift $g_{i,2}^{p^{\nu_i}}$ instead of $g_{i,2}$, where $e_i = p^{\nu_i}\mu_i$ and $p \nmid \mu_i$. We ultimately obtain $h_i^{p^{\nu_i}}$ from which we recover h_i by taking p th roots of the coefficients.

The polynomials $h_{i,m} \in F[x_1, \ldots, x_{m-1}, y]$ are returned such that with high probability

$$\prod_{i\in\{1,\ldots,r\}\setminus J} h_{i,m}^{e_i} = f_m.$$

Set $g_{i,m} \leftarrow h_{i,m}(x_1, \ldots, x_{m-1}, x_m - b_m x_1 - a_m)$ for all $i \in \{1, \ldots, r\} \setminus J$, $g_{j,m} \leftarrow g_{j,m-1}$ for all $j \in J$.

Step T (Final Translation): Return

$$h_i \leftarrow g_{i,n}(x_1 - a_1, x_2, \ldots, x_n),\ e_i,\ 1 \le i \le r. \quad \square$$

We wish to remark that the lifting with the multiplicities e_i could have been avoided over fields of characteristic 0. Once we know the factorization of f_1 we can lift $\prod g_{i,2}$ with respect to the squarefree part of f. A straight-line program for the squarefree part of f can be obtained by using the Polynomial GCD algorithm [35] on a translated image of f and its partial derivative. We mention this fact because it distinguishes our straight-line approach further from sparse factorization algorithm in [36]. That algorithm cannot afford to compute the squarefree part of the sparse input since that part can become dense. However, with our theory we can always find a straight-line program computing this dense squarefree part.

---

[35] E. Kaltofen, ‘‘Computing with polynomials given by straight-line programs I; Greatest common divisors,’’ *Proc. 17th ACM Symp. Theory Comp.*, pp. 131-142, 1985.

[36] J. von zur Gathen and E. Kaltofen, ‘‘Factoring sparse multivariate polynomials,’’ *J. Comp. System Sci.*, vol. 31, pp. 265-287, 1985.

We now analyze the overall failure probability of the Sparse Factorization algorithm. We first estimate the probability with which the correct answer is computed, under the assumption that ''failure'' does not occur. First of all, f_1 must have the same factorization pattern as f. The probability of this event is estimated in theorem 2.1. We shall assume now that the event has happened. The second event necessary is (*) for all $2 \le m \le n$, which guarantees the input conditions for the calls to One-Variable Lifting. We will show in the proof of theorem 4.1 that this event occurs with probability $> 1 - (n-2)d/\mathrm{card}(R)$. Now also assume that the second event has occurred. Then all $n-2$ calls to One-Variable Lifting produce correct answers with probability $> 1 - \varepsilon/2$. Therefore all three events occur together with with probability greater than

$$\left(1 - \frac{4d\,2^d + d^3}{\mathrm{card}(R)}\right)\left(1 - \frac{(n-2)d}{\mathrm{card}(R)}\right)\left(1 - \frac{\varepsilon}{2}\right) > 1 - \varepsilon.$$

''Failure'' is returned under three circumstances in step I and during the calls to One-Variable Lifting. First, P is not defined at ϕ with probability $< 2^{l+1}/\mathrm{card}(R) < \varepsilon/8$ by an argument similar to that used in Lemma 4.3 of [37]. The remaining two possibilities of step I to fail are that P is not recognized to be defined at ϕ in case $F = \mathbf{Q}$ or that the computation of f_1 fails. In summary, ''failure'' is returned in step I with probability $< \varepsilon/2$. Any of the calls to One-Variable Lifting can fail in two different ways, namely that either it is called with incorrect inputs or that the Sparse Conversion inside the Lifting procedure fails. By the above analysis the first event does not happen with probability $> 1 - \varepsilon$ and then the second does not happen in all $n-2$ calls to Lifting with probability $> 1 - \varepsilon/2$. Therefore ''failure'' is not returned by Sparse Factorization with probability $> 1 - 2\varepsilon$. We have the following theorem.

Theorem 4.1: Algorithm Sparse Factorization does not fail and outputs the sparse factorization of f with probability $1 - 3\varepsilon$. In that case it reduces the problem in polynomially many arithmetic steps on an algebraic RAM over F as a function of

$$\mathrm{len}(P),\ \ d,\ \ \text{and}\ \ \mathrm{mon}(h_i),\ 1 \le i \le r,$$

to factoring bivariate polynomials. It requires polynomially many randomly selected field elements.

Proof: If the algorithm does not fail or does not compute an intermediate incorrect result, it clearly takes polynomial-time. It remains to show that condition (*) is satisfied with the said probability. We have assumed that the evaluations preserve the factorization pattern. Now if $h_i \in F[x_2, \ldots, x_{m-1}]$ then h_i is equal to some $g_{j,m-1}$, $j \in J$. Clearly such a $g_{j,m-1}$ must divide $\overline{f_m}$. We need to insure that for

---

[37] E. Kaltofen, ''Computing with polynomials given by straight-line programs I; Greatest common divisors,'' *Proc. 17th ACM Symp. Theory Comp.*, pp. 131-142, 1985.

$$h_{i,m}(x_1 + a_1,\, x_2, \ldots, x_{m-1},\, b_m x_1 + y + a_m,\, b_{m+1} x_1 + a_{m+1}, \ldots, b_n x_1 + a_n)$$

we have $h_{i,m}(x_1, \ldots, x_{m-1}, 0) \notin F[x_2, \ldots, x_{m-1}]$ if $h_i \notin F[x_2, \ldots, x_{m-1}]$. Assuming the latter let

$$\hat{h}_{i,m} = h_i(x_1 + \alpha_1,\, x_2, \ldots, x_{m-1},\, \beta_m x_1 + y + \alpha_m,\, \beta_{m+1} x_1 + \alpha_{m+1}, \ldots, \beta_n x_1 + \alpha_n)$$

$\in F[\alpha_1, \alpha_m, \ldots, \alpha_n, \beta_m, \ldots, \beta_n, x_1, \ldots, x_{m-1}, y]$. Now

$$\deg_{x_1}(\hat{h}_{i,m}) > 0 \quad\text{and}\quad \mathrm{ldcf}_{x_1}(\hat{h}_{i,m}) \in F[\beta_m, \ldots, \beta_n, x_2, \ldots, x_{m-1}].$$

Let $\sigma_i(\beta_m, \ldots, \beta_n) x_2^{j_2} \cdots x_{m-1}^{j_{m-1}}$ be a non-zero monomial in the $\mathrm{ldcf}_{x_1}(\hat{h}_{i,m})$. Then $\prod_{i \notin J} \sigma_i(b_m, \ldots, b_n) \neq 0$ guarantees that

$$h_{i,m}(x_1, \ldots, x_{m-1}, 0) = g_{i,m-1} \notin F[x_2, \ldots, x_{m-1}] \text{ for all } i \notin J,$$

which already implies (*). Since $\deg(\prod \sigma_i) \le \sum \deg(h_i) \le d$, this happens for all $3 \le m \le n$ with probability $\ge 1 - (n-2)d/\mathrm{card}(R)$. □

From the polynomial-time results on bivariate factorization [38] and [39] we get the following corollary.

Corollary: If $F = \mathbf{Q}$ or $F = \mathrm{GF}(q)$ the sparse factorization of $f \in F[x_1, \ldots, x_n]$ given by a straight-line program P can be computed correctly with probability $1 - \varepsilon$ on an algebraic RAM over F in expected polynomially many binary steps in

$$\mathrm{len}(P),\;\; d,\;\; \log(\frac{1}{\varepsilon}),\;\; \text{cc-size}(f),\;\; \text{el-size}(P),\;\; \text{and}\;\; \mathrm{mon}(h_i),\; 1 \le i \le r.$$

Proof: The difference between theorem 4.1 and this corollary is that we would always output a factorization of f. If ‘‘failure’’ occurs we just restart the algorithm. We now note that the algorithm may in unfortunate circumstances take exponential or even infinite time. However, such an event can be made quite unlikely so that the expected running time stays polynomial. □

As in the Sparse Conversion algorithm, it is an important feature of the Sparse Factorization algorithm that the number of monomials need not be known before-hand. Therefore, a symbolic computation system can begin the factorization and inform the user of lower bounds of the output size from time to time. Usually, it suffices to know that the final factorization is large without completely computing it. The question most users have about their computations is whether they lead to simple answers and if so which.

---

[38] E. Kaltofen, ‘‘Polynomial-time reductions from multivariate to bi- and univariate integral polynomial factorization,’’ *SIAM J. Comp.*, vol. 14, pp. 469-489, 1985.

[39] J. von zur Gathen and E. Kaltofen, ‘‘Factoring multivariate polynomials over finite fields,’’ *Math. Comp.*, vol. 45, pp. 251-261, 1985.

5. Conclusion

Our development of a theory for manipulating polynomials given by straight-line programs lead us to what we believe is one of the most sophisticated random polynomial-time algorithm among problems being known to belong to the random polynomial-time complexity class. To justify our claim we wish to again refer to the classical problem of factoring the van der Monde determinant as mentioned in the introduction. We are in the position of automatically producing its factorization in random polynomial-time by bringing together several very different and important ideas. First, the Hensel lemma applied to polynomial factorization in the uni-, bi-, and multivariate situation. Second, Berlekamp's polynomial factorization algorithm over finite fields and Lovász's short lattice-vector algorithm, which are the keystones for univariate rational polynomial factorization in polynomial-time. Third, the Hilbert irreducibility theorem and its effective versions. Then the technique of randomly evaluating multivariate polynomials at points and modulo large primes leading also to the Sparse Conversion algorithm. And finally, Strassen's technique of eliminating divisions from straight-line programs that has helped us to produce the Polynomial Coefficients algorithm.

We are, however, left with an important open problem. Notice that our Sparse Factorization algorithm is of random polynomial running time only if the input polynomial factors sparsely. This is the case for the van der Monde determinant and it is easy to construct different classes of determinants that factor sparsely. However, even sparse polynomials can have dense irreducible factors [40], and it would be better if we could compute the factorization in terms of straight-line programs themselves. Then one could always convert those factors that are sparse to their sparse representation but leave the dense factors in straight-line representation. At this moment we cannot prove that the irreducible factors of a polynomial given by a straight-line program can always be represented by straight-line programs of polynomial length. The possibility that this problem is inherently difficult cannot be completely ruled out. For instance, Valiant [41] has demonstrated that p-computable polynomials, which are those of polynomially bounded degree and straight-line program length, are very unlikely closed under operations such as multiple partial derivatives. However, from our experience with the GCD problem [42] we conjecture that p-computable polynomials are closed under factorization into irreducibles. The more important problem is, of course, to find the straight-line programs for the irreducible factors in polynomial-time.

---

[40] J. von zur Gathen and E. Kaltofen, ''Factoring sparse multivariate polynomials,'' *J. Comp. System Sci.*, vol. 31, pp. 265-287, 1985.

[41] L. Valiant, ''Reducibility by algebraic projections,'' *L'Enseignement mathématique*, vol. 28, pp. 253-268, 1982.

[42] E. Kaltofen, ''Computing with polynomials given by straight-line programs I; Greatest common divisors,'' *Proc. 17th ACM Symp. Theory Comp.*, pp. 131-142, 1985.

Note added in on November 9, 1985: The author has shown that the above mentioned problem has indeed a positive solution. The paper containing the proof, ‘‘Uniform closure properties of p-computable functions’’, is currently being written.

Corrections to [Ka85d]

p. 132, c. 2, l.+8:

The example should read

$$f = \prod_{i=2}^{n}(x_i - 1) \prod_{i=2}^{n}(x_1 - \sum_{j=0}^{d} x_i^j).$$

p. 137, c. 1, Step BT:

Replace step BT by

Step BT (Back-Transformation): At this point Q computes $w_{l+1,\delta}$ with

$$f(x_1,\ldots,x_n) = \sum_{\delta=0}^{d} f(w_{l+1,\delta})(x_1 - a_1)^{\delta}.$$

We compute $w_{l+2,\delta}$ such that

$$f(x_1,\ldots,x_n) = \sum_{\delta=0}^{d} f(w_{l+2,\delta})\, x_1^{\delta}.$$

by a fast ‘‘radix conversion’’ method [43], §4.4, Exercise 14, which we shall briefly present as an algebraic RAM algorithm.

Split $f = f_0 + (x_1 - a_1)^{\lceil d/2 \rceil} f_1$ with

$$f_0 = \sum_{0 \le \delta < \lceil d/2 \rceil} c_\delta (x_1 - a_1)^{\delta}, \quad f_1 = \sum_{0 \le \delta \le \lceil d/2 \rceil} c_{\delta + \lceil d/2 \rceil} (x_1 - a_1)^{\delta}.$$

Convert f_0 and f_1 by recursive application of the algorithm.
Compute $f_2 = (x_1 - a_1)^{\lceil d/2 \rceil}$ by repeated squaring.
Compute $f_1 f_2 + f_0$ by multiplication and addition.

The complexity $T(d)$ of this algorithm satisfies $T(d) \le 2\,T(d/2) + c\,M(d)$, c a constant, and hence is $T(d) = O(\log(d)\,M(d))$.

Finally, set $W = \{w_{\lambda\delta}\} \cup \{u \mid u$ is any of the intermediate variables$\}$. Return $Q = (\{x_2, \ldots, x_n\}, W, C_Q, S \cup \{a_1\})$. □

p. 138, c. 2, l.+3:

... stating that products of primitive polynomials ...

p. 138, c. 2, Step R:

$\mathrm{card}(R) > \max(2^{l+4}, 8d^3)/\varepsilon$, because of changes in step C. Note that Theorem 5.2 remains valid as stated.

p. 138, c. 2, Step C:

Replace step C by

Step C (Coefficient Determination): For $\rho = 1, 2$ test whether $\tilde{P}_\rho$ is defined at $\phi(y_\nu) = a_\nu$, $1 \le \nu \le n$. If not return ‘‘failure’’. For $F = \mathbf{Q}$ we proceed as follows. We call algorithm Zero-Division

[43] D. E. Knuth, *The Art of Programming, vol. 2, Semi-Numerical Algorithms, ed. 2,* Addison Wesley, Reading, MA, 1981.

Test on each P_ρ five times. If ''failure'' occurs all five times for any of the two programs return ''failure''. Otherwise both P_ρ are defined at ϕ.

Now call algorithm Polynomial Coefficients with input $\tilde{P}_\rho$, a_1, and d. We obtain $Q_\rho = (\{y_2, \ldots, y_n\}, W_\rho, C_\rho, T_\rho)$ such that $c_{\rho\delta} \in f(Q_\rho)$. Notice that Q_ρ are still defined at $\phi(y_\nu) = a_\nu$, $2 \le \nu \le n$.

p. 140, c. 1, l.-1:

The very faint equation is

$$\sum_{(e_1,\ldots,e_{i-1})\in J_i} \sum_{\delta=0}^{d-e_1-\cdots-e_{i-1}} \gamma_{e_1,\ldots,e_{i-1},\delta}\, b_{k1}^{e_1} \cdots b_{k,i-1}^{e_{i-1}}\, b_{ki}^{\delta} = g_{ki}, \qquad (\dagger)$$

p. 140, c. 2, l.-14:

Since the events are not independent, the probability estimate should be

$$1 - \sum_{i=1}^{n} \frac{\deg(\sigma_i)}{r} \ge 1 - \frac{ndm}{r} > 1 - \frac{nd(d+1)^n}{r} > 1 - \varepsilon.$$