SIAM J. COMPUT.
Vol. 12, No. 4, November 1983

0097-5397/83/1204-0003 $01.25/0

FAST PARALLEL COMPUTATION OF POLYNOMIALS USING FEW PROCESSORS*

L. G. VALIANT†, S. SKYUM‡, S. BERKOWITZ§ AND C. RACKOFF§

Abstract. It is shown that any multivariate polynomial of degree d that can be computed sequentially in C steps can be computed in parallel in $O((\log d)(\log C + \log d))$ steps using only $(Cd)^{O(1)}$ processors.

Key words. parallel computation, polynomials, complexity theory

Introduction. Hyafil [6] showed that any polynomial q of degree d that can be computed sequentially in C $\{+, -, \times\}$-steps can be computed in parallel in time proportional to $(\log d)(\log C + \log d)$. Unfortunately his method requires $C^{\log d}$ processors in general. Thus even if C and d are both bounded polynomially in terms of the number of indeterminates the number of processors required would not be. In this paper we give an improved construction that achieves the same time bound but with only $(Cd)^{\beta}$ processors, for an appropriate constant β. The achievability of such fast time bounds with only polynomially many processors was known previously only for certain specific polynomials such as the determinant [5]. Throughout we shall use the unrestricted model of parallelism described by Borodin and Munro [2].

Simultaneous resource bounds in discrete computations are discussed in detail by Cook [3]. An important positive result in that area, due to Ruzzo [8], is that context-free languages can be recognized by Boolean circuits that simultaneously have polynomial size and $\log^2 n$ depth. This is also a corollary of our present result as can be seen as follows: Consider the monotone Boolean circuit defined by the Cocke–Kasami–Younger algorithm. Regard this as an arithmetic program over the reals using the correspondence "and" $\rightarrow \times$ and "or" $\rightarrow +$. This program has linear degree and polynomial program size. An application of our construction therefore gives a fast parallel program using only polynomially many processors. In addition, the parallel program can be reinterpreted as a shallow monotone circuit, as required. The concept of "degree" for Boolean circuits exploited in this argument is discussed formally in [9].

Finally we observe that, in the terminology of [11], the question whether every polynomial family of polynomially bounded degree and program size is a p-projection of the determinant is open. An affirmative answer to this combined with Csanky's result would give an alternative proof of our main result here, at least in the case of fields of characteristic 0.

This paper is a simplification and improvement of an earlier result by Valiant and Skyum [12].

Definitions and the main theorem. Let F be a field and let $F[x_1, \cdots, x_n]$ be the ring of polynomials over indeterminates $x_1, \cdots, x_n$ with coefficients from F. A *program* f over F is a sequence of instructions

$$v_i \leftarrow v_i' \circ v_i'', \qquad i = 1, \cdots, C$$

* Received by the editors May 15, 1981, and in revised form September 16, 1982.
† Aiken Computation Laboratory, Harvard University, Cambridge, Massachusetts 02138.
‡ Computer Science Department, Aarhus University, Aarhus, Denmark.
§ Computer Science Department, University of Toronto, Toronto, Canada M5S 1A7.

where for each value of i

(I) $v_i', v_i'' \in F \cup \{x_1, \cdots, x_n\} \cup \{v_1, \cdots, v_{i-1}\}$ and

(II) $\circ$ is one of the two ring operators $+, \times$.

Note that since $-1 \in F$, subtraction can be easily simulated. The formal polynomial computed at v_i can be defined in the obvious way and is denoted by $f(v_i)$. The degree of $f(v_i)$ in the usual sense is denoted by $d(v_i)$. For convenience, we will assume $d(v_i') \geqq d(v_i'')$. For the moment we will also assume that f is *homogeneous*; by this we mean that if $v_i \leftarrow v_i' + v_i''$, then $d(v_i') = d(v_i'')$. We say that f has size C (the number of instructions) and computes the polynomial $f(v_C)$. The following fact due to Strassen [10] shows that forcing f to be homogeneous is not a serious restriction.

FACT 1. *If a nonhomogeneous program f computes a polynomial p of degree d and has size C, then there is a homogeneous program which has size $O(Cd^2)$ and which computes $d+1$ polynomials whose sum is p.*

Outline of proof. For each line of the original program, we will have $d+1$ lines which compute the first $d+1$ homogeneous parts of the polynomial computed at the original line. If the operation is $+$, then we add similar degree components. If the operation is $\times$, then we can view each set of operands as the coefficients of a polynomial, and perform polynomial multiplication (about d^2 operations) to obtain the coefficients of a new polynomial; only the first $d+1$ of these are needed. □

We will also assume that f is a smallest possible program for computing $f(v_C)$. The following fact is easy to verify.

FACT 2. *If f is a smallest possible homogeneous program for computing $f(v_C)$, then $f(v_i)$ is not the zero polynomial for any i, and $d(v_i) \leqq d(v_C)$ for all i.*

We will denote the set $\{v_i\}$ by V, the set $\{x_i\}$ by X, and the set $V \cup X \cup F$ by $\bar{V}$. The *depth* of $v \in \bar{V}$ is the length of the longest possible sequence $u_1, \cdots, u_D$ such that $u_1 = v$, for each i, $u_{i+1} = u_i'$ or $u_{i+1} = u_i''$, and $u_D \in F \cup X$.

DEFINITION. Let $v, w \in \bar{V}$. We define $f(v; w) \in F[x_1, \cdots, x_n]$ by induction on the depth of w. Firstly, if $v = w$ (that is they are the same node) then $f(v; w) = 1$. Otherwise, if $w \in F \cup X$, then $f(v; w) = 0$. Otherwise, if $w \leftarrow w' + w''$ then $f(v; w) = f(v; w') + f(v; w'')$ and if $w \leftarrow w' \times w''$ then $f(v; w) = f(w'') \cdot f(v; w')$. (The motivation for this definition is that if $d(w) < 2d(v)$, then $f(v; w)$ turns out to be the coefficient of v' in $f(w)$ in a modified program obtained by replacing node v by a new indeterminate v'. In fact, the proof below can be rewritten so as to only use the value of $f(v; w)$ in this case.)

FACT 3. *It is easy to see that each $f(v)$ computed in the homogeneous program f is also homogeneous, that is, all of its monomials have the same degree. It is also easy to prove by induction on the depth of w that each nonzero $f(v; w)$ is homogeneous and satisfies*: $\text{degree}(f(v; w)) = d(w) - d(v)$.

DEFINITION. If $a > 0$, define $V_a = \{t \in V \mid d(t) > a, t \leftarrow t' \times t'', d(t') \leqq a\}$.

LEMMA 1. *Say that $v, w \in V$, $d(v) \leqq a < d(w)$. Then*

$$f(v; w) = \sum_{t \in V_a} (f(v; t) \cdot f(t; w)) \quad \textit{and} \quad f(w) = \sum_{t \in V_a} f(t) \cdot f(t; w).$$

Proof. Notice that $v \neq w$ since $d(v) < d(w)$. We will prove the lemma by induction on the depth of w, keeping v and a fixed.

Case 1. $w \in F \cup X$. This cannot happen since we can't have $d(w) > a > 0$.

Case 2. $w \leftarrow w' + w''$, $d(w') = d(w'') = d(w)$. Assume the lemma holds for w', w''.

$$f(v; w) = f(v; w') + f(v; w'') = \sum_{t \in V_a} (f(v; t) \cdot f(t; w')) + \sum_{t \in V_a} (f(v; t) \cdot f(t; w''))$$

$$= \sum_{t \in V_a} (f(v; t) \cdot (f(t; w') + f(t; w''))) = \sum_{t \in V_a} (f(v; t) \cdot f(t; w)).$$

Similarly, $f(w) = \sum_{t \in V_a} f(t) \cdot f(t; w)$.

Case 3. $w \leftarrow w' \times w''$, $a \geqq d(w') \geqq d(w'')$. (Recall that $d(w') \geqq d(w'')$ by definition.) Then $w \in V_a$. Clearly $f(v; w) = f(v; w) \cdot f(w; w)$. Let s be any other element of V_a. Then $f(s; w) = f(w'') \cdot f(s; w')$. But $f(s; w') = 0$ by Fact 3, since $d(s) > a \geqq d(w')$. So $f(s; w) = 0$. So $f(v; w) = \sum_{t \in V_a} (f(v; t) \cdot f(t; w))$. Similarly, $f(w) = \sum_{t \in V_a} f(t) \cdot f(t; w)$.

Case 4. $w \leftarrow w' \times w''$, $d(w) \geqq d(w') > a$. Assume the lemma holds for w'.

$$f(v; w) = f(w'') \cdot f(v; w') = f(w'') \cdot \sum_{t \in V_a} (f(v; t) \cdot f(t; w'))$$

$$= \sum_{t \in V_a} (f(v; t) \cdot f(w'') \cdot f(t; w')) = \sum_{t \in V_a} f(v; t) \cdot f(t; w).$$

Similarly, $f(w) = \sum_{t \in V_a} f(t) \cdot f(t; w)$. □

THEOREM 1. *Let f be a homogeneous program of size C which computes a polynomial p of degree d. Then there is a program f' of size $O(C^3)$ which computes p, such that the largest depth of any node is $O(\log C \log d)$.*

Proof. The construction of f' will proceed in $\lceil \log_2 d \rceil$ stages; each stage will add at most $\log C$ to the depth of the program.

At stage 0 we compute all $f(w)$ and $f(v; w)$ that have degree at most $2^0 = 1$. Since these are linear forms in n indeterminates and $C \geqq n - 1$ if f is minimal, depth $2 + \lceil \log_2 C \rceil$ is sufficient for this. At stage $i+1$ we compute all $f(w)$ and $f(v; w)$ that have degree in the range $(2^i, 2^{i+1}]$. By Fact 2 we are done after stage $\lceil \log_2 d \rceil$.

Say that $2^{i+1} \geqq d(w) > 2^i$. Let $a = 2^i$. Then by Lemma 1, $f(w) = \sum_{t \in V_a} f(t) \cdot f(t; w) = \sum_{t \in V_a} f(t') \cdot f(t'') \cdot f(t; w)$. By definition of V_a, each $f(t')$, $f(t'')$, $f(t; w)$ has already been computed. So $f(w)$ can be computed adding only $O(\log C)$ depth.

Say that $2^{i+1} \geqq d(w) - d(v) = \text{degree}\,(f(v; w)) > 2^i$. Let $a = d(v) + 2^i$. By Lemma 1, $f(v; w) = \sum_{t \in V_a} f(v; t) \cdot f(t; w) = \sum_{t \in V_a} f(t'') \cdot f(v; t') \cdot f(t; w)$. Each $f(v; t')$ and $f(t; w)$ has already been computed. It is possible, however, that $d(t'')$ is very large, say that $d(t') \geqq d(t'') > 2^{i+1}$. If $f(v; t') = 0$, then we're okay, since $f(t'') \cdot f(v; t') \cdot f(t; w)$ can still be computed. So say that $d(t'') > 2^{i+1}$ and $f(v; t') \neq 0$. Then $d(t') \geqq d(v)$, so $d(t) = d(t') + d(t'') > d(v) + 2^{i+1} \geqq d(w)$. So $f(t; w) = 0$. So $f(v; w)$ can be computed adding only $O(\log C)$ depth.

The *size* of the new program is dominated by the time to compute the $f(v; w)$. There are C^2 choices of pairs (v, w) and the computation of each $f(v; w)$ takes $O(C)$ steps. The overall size is therefore $O(C^3)$. □

By combining Theorem 1 and Fact 1 we have:

THEOREM 2. *Let f be a nonhomogeneous program of size C which computes a polynomial p of degree d. Then there is a program f' of size $O((Cd^2)^3)$ and depth $O((\log C + \log d) \log d)$ which computes p.*

There is another method for handling nonhomogeneous programs without first making them homogeneous. Given a nonhomogeneous program f, we first define the degree of a node v, $d(v)$, differently than above. The degree of a field member is 0; the degree of an indeterminate is 1; the degree of a multiplication node is the sum of the degrees of its inputs; the degree of an addition node is the maximum degree of its inputs. We define the degree of f to be the maximum degree of any node.

For $a > 0$, define $V'_a = \{t \in V \mid d(t) > a, t \leftarrow t' + t'', d(t'') \leqq a\}$. We state the following lemma and theorem without proof.

LEMMA 2. *Say that $v, w \in V$, $d(v) \leqq a < d(w)$. Then*

$$f(v; w) = \sum_{t \in V_a} (f(v; t) \cdot f(t; w)) + \sum_{t \in V'_a} (f(v; t'') \cdot f(t; w))$$

and

$$f(w) = \sum_{t \in V_a} (f(t) \cdot f(t; w)) + \sum_{t \in V'_a} (f(t'') \cdot f(t; w)).$$

THEOREM 3. *Let f be a nonhomogeneous program of size C and degree d. Then there is a program f' of size $O(C^3)$ and depth $O(\log C \log d)$ which computes the same polynomial.*

Remarks. 1. Strassen [10] showed that for infinite fields Fact 1 holds even when divisions are allowed in the original program but not in the transformed one. It follows that Theorem 2 holds under the same hypothesis. Recently Borodin, von zur Gathen and Hopcroft [1] have shown that the infinity assumption is inessential.

2. It is easy to verify that the above theorems hold for structures much less restricted than fields. For example, the constructions work for monotone arithmetics (i.e., with constants from the nonnegative reals). As observed in the introduction, the same results then follow for monotone Boolean circuits when the notion of degree is suitably interpreted. More formally, it can be verified that it is sufficient for F to be a semiring in the sense of Jerrum and Snir [7].

3. Using Lemmas 1 and 2, analogues of Theorems 1 and 3 can be proved with a size bound for f' of $C^\alpha \log d$ where α is such that $n \times n$ matrices can be multiplied in n^α operations ($\alpha < 2.496$ [4]).

REFERENCES

[1] A. BORODIN, J. VON ZUR GATHEN AND J. HOPCROFT, *Fast parallel matrix and GCD computations*, Proc. 23rd IEEE Symposium on Foundations of Computer Science, 1982, pp. 65–71.

[2] A. BORODIN AND I. MUNRO, *The Computational Complexity of Algebraic and Numeric Problems*, American Elsevier, New York, 1975.

[3] S. A. COOK, *Deterministic CFL's are accepted simultaneously in polynomial time and log squared space*, Proc. 11th ACM Symposium on Theory of Computing, 1979, pp. 338–345.

[4] D. COPPERSMITH AND S. WINOGRAD, *On the asymptotic complexity of matrix multiplication*, Proc. 22nd IEEE Symposium on Foundations of Computer Science, 1981, pp. 82–90.

[5] L. CSANKY, *Fast parallel inversion algorithms*, this Journal, 5 (1976), pp. 618–623.

[6] L. HYAFIL, *On the parallel evaluation of multivariate polynomials*, Proc. 10th ACM Symposium on Theory of Computing, 1978, pp. 193–195.

[7] M. JERRUM AND M. SNIR, *Some exact complexity results for straight-line computations over semirings*, J. Assoc. Comput. Mach., 1982, to appear.

[8] W. L. RUZZO, *On uniform circuit complexity*, Proc. 20th IEEE Symposium on Foundations of Computer Science, 1979, pp. 312–318.

[9] S. SKYUM AND L. G. VALIANT, *A complexity theory based on Boolean algebra*, Proc. 22nd IEEE Symposium on Foundations of Computer Science, 1981, pp. 244–253.

[10] V. STRASSEN, *Vermeidung von Divisionen*, J. Reine Angew. Math., 264 (1973), pp. 182–202.

[11] L. G. VALIANT, *Completeness classes in algebra*, Proc. 11th ACM Symposium on Theory of Computing, 1979, pp. 249–261.

[12] L. G. VALIANT AND S. SKYUM, *Fast parallel computation of polynomials using few processors*, Lecture Notes in Computer Science, 118, Springer-Verlag, New York, 1981, pp. 132–139.