Theoretical Computer Science 41 (1985) 325-330
North-Holland

NOTE

SIMULATING PROBABILISTIC BY DETERMINISTIC ALGEBRAIC COMPUTATION TREES

Friedhelm MEYER AUF DER HEIDE
IBM Research Laboratory, San Jose, CA 95193, U.S.A.

Communicated by M.S. Paterson
Received April 1985
Revised July 1985

Abstract. A probabilistic algebraic computation tree (probabilistic ACT) which recognizes $L \subset R^n$ in expected time T, and which gives the wrong answer with probability $\leqslant \varepsilon < \frac{1}{2}$, can be simulated by a deterministic ACT in $O(T^2 n)$ steps. The same result holds for linear search algorithms (LSAs). The result for ACTs establishes a weaker version of results previously shown by the author for LSAs, namely that LSAs can only be slightly sped up by their nondeterministic versions. This paper shows that ACTs can only be slightly sped up by their probabilistic versions. The result for LSAs solves a problem posed by Snir (1983). He found an example where probabilistic LSAs are faster than deterministic ones and asked how large this gap can be.

Introduction

Linear search algorithms (LSAs) and algebraic computation trees (ACTs) are abstractions of random access machines with operations $\{+, -\}$ and $\{+, -, *, /\}$, respectively. Both LSAs and ACTs have turned out to be well suited for proving lower time bounds for many interesting problems. (For LSAs, see [3, 8, 12], and for ACTs, see [2]). On the other hand, many lower bounds for LSAs also hold for random access machines (see [5, 7]), i.e., for a realistic computational model.

Recently, some effort has been made to understand the power of probabilistic versions of LSAs (PLSAs). Manber and Tompa [6] and Snir [13] proved lower bounds for the complexity of PLSAs with one- or two-sided error.

In this paper we show that PLSAs and PACTs (probabilistic ACTs) can only be slightly faster than their deterministic versions. Any PACT or PLSA which recognizes a language $L \subset R^n$ in expected time T, and gives the wrong answer with probability $\leqslant \varepsilon < \frac{1}{2}$ can be simulated by an LSA or ACT in time $O(T^2 n)$.

The result for LSAs solves a generalization of an open problem posed by Snir [13]. He established an example of a language $L \subset R^n$ which needs $\Theta(n)$ steps on LSAs but can for some $\gamma < 1$ be solved in only $O(n^\gamma)$ steps by a PLSA, even if no error is allowed. Snir also mentioned that there are languages for which PLSAs with one-sided error are faster than they are for the complements of these languages.

This motivated him to ask how big such gaps can be. Our result shows that the complexities of languages and their complements are polynomially related.

The result for ACTs is important in the spirit of the papers [8, 9]. LSAs and ACTs are nonuniform computation models. In the above papers it is shown that this property, namely to only handle inputs consisting of a fixed number of variables, i.e., to deal with n-dimensional restrictions of problems, makes LSAs powerful enough to compute n-dimensional restrictions of some NP-complete problems such as the knapsack problem or the traveling salesman problem in polynomial time. Thus, the question arises whether the nonuniformity also makes ACTs very strong. Our result shows that they are at least strong enough to recognize in polynomial time n-dimensional restrictions of many languages $L \subset N^*$, which can be recognized in random polynomial time, namely all those which can be recognized by a probabilistic RAM with operations $+$, $-$, $*$ in (uniform) time polynomial in n.

The proof of our result is based on a technique to simulate probabilistic computations with two-sided error by deterministic computations, if only finitely many inputs are allowed. Such a technique is first used by Bennett and Gill [1] and later by Reif [11]. In order to obtain simulations for our computation models (in which the input set, R^n, is infinite), we have to consider the structure of functions computed in ACTs and LSAs. Here, as in [2], we again make use of Milnor's bound [10] for the number of connected components into which R^n can be subdivided by the set of roots of a polynomial with given degree.

1. A general simulation

Let A be a set, and let F be a family of functions $f: A \to R$. A *probabilistic computation tree* (PCT) D with queries defined by F is a binary computation tree which takes inputs from A. An inner node v of D is either a probabilistic node or a query node. At a probabilistic node, a coin is flipped to determine which branch to follow. At a query node a query $f(x)\ \sigma\ 0$ is asked to determine which branch to follow. Here, $f \in F$, $x \in A$ is the input, and $\sigma \in \{<, >, =\}$. Each leaf is either accepting or rejecting. The complexity of D is the maximum over all $x \in A$ of the expected running time of D on input x. D recognizes $L \subset A$ with threshold $\frac{1}{2}+\varepsilon$, $0<\varepsilon<\frac{1}{2}$, if, for each $x \in A$, Prob(D treats x correctly)$\geq \frac{1}{2}+\varepsilon$, where "$D$ treats x correctly" means D accepts x if $x \in L$ and rejects it otherwise. D is deterministic, i.e., a CT, if it contains no probabilistic nodes. A computation of D is a sequence of functions from F used for queries on some path of D. In the sequel, let D be a PCT with queries defined by F and with complexity T which recognizes $L \subset A$ with threshold $\frac{1}{2}+\varepsilon$.

We say that a CT strongly simulates D if it recognizes L and if its computations are concatenations of computations of D.

First we notice, as already shown in [6], that it suffices to consider the depth of D rather than its complexity.

Lemma 1.1 (Manber and Tompa [6]). *The PCT derived from D by replacing the internal nodes of D at depth $(2/\varepsilon)T$ by accepting leaves and removing all deeper nodes recognizes L with threshold $\frac{1}{2}+\frac{1}{2}\varepsilon$.*

From now on we assume w.l.o.g. that D has depth T. Next we show how to simulate D by a CT if A is finite. Similar forms of the following lemma are already implicitly used in [1, 11].

Lemma 1.2. *If A is finite, then D can be strongly simulated by a CT with depth* $O(T\log(|A|))$.

Proof. For some positive integer s let D_s be the PCT of depth sT which started with input x, runs D on x for s independent trials, and accepts x if x is accepted by D in at least $\frac{1}{2}s$ runs. We need the following claim.

Claim 1.3. *D_s recognizes L with threshold $\alpha = 1-(1-4\varepsilon^2)^{s/2}$.*

Proof. Let $x \in A$. D_s treats x correctly, if x is treated correctly in at least $\frac{1}{2}s$ of the s independent runs of D executed by D_s. Thus, Prob(D_s treats x incorrectly) $\leqslant \sum_{i=s/2}^{s}\binom{s}{i}(\frac{1}{2}-\varepsilon)^i(\frac{1}{2}+\varepsilon)^{s-i} \leqslant (1-4\varepsilon^2)^{s/2}$ by Chernoff's bound, see [4]. □

Proof of Lemma 1.2 (*continued*). Now, let $\bar{e}=(e_1,\ldots,e_{sT})$ be a sequence of outcomes of coin flips, and let $D_{\bar{e}}$ be the CT arising from D_s by removing on each path at each ith probabilistic node v the branch which is not chosen, if the outcome of the coin flip at v is e_i. The computation of D_s can be looked upon as follows. Started with input x, D_s first randomly chooses an $\bar{e}$, and then runs $D_{\bar{e}}$ with input x. Thus, "Prob(D_s treats x correctly) $\geqslant \alpha$" means that in $\alpha 2^{sT}$ of the 2^{sT} $D_{\bar{e}}$'s, x is treated correctly. Thus, there is an $\bar{e}$ such that $D^* := D_{\bar{e}}$ treats at least $\alpha|A|$ inputs correctly. Therefore, if $\alpha|A| > |A|-1$, D^* recognizes L. Inserting the expression for α from Claim 1.3 in this inequality, we have that D^* recognizes L if $(1-4\varepsilon^2)^{s/2} < 1/|A|$. As $0<\varepsilon<\frac{1}{2}$, $\log(1-4\varepsilon^2)$ exists and is negative. Therefore, the above inequality yields that D^* recognizes L if

$$s \geqslant \frac{2\log(|A|)}{-\log(1-4\varepsilon^2)} = O(\log(|A|)). \qquad \square$$

Here we note that this construction makes the deterministic computation nonuniform (even if the probabilistic one were uniform), because of the need to choose a suitable $\bar{e}$.

Lemma 1.2 shows an efficient simulation of probabilistic by deterministic computations if the input set is finite. But for the computational models we are interested in the case that the input set is infinite, namely R^n. On the other hand, in Lemma 1.2 we have not used any properties of the set F of functions. We shall now take into account the 'structure' of F in order to get results similar to Lemma 1.2 when the input set is infinite.

Let $F=\{f_1,\ldots,f_m\}$, and let $\bar{\sigma}=(\sigma_1,\ldots,\sigma_m)\in\{<,>,=\}^m$. Then, $I_{\bar{\sigma}}:=\{x\in A,$ $f_i(x)\ \sigma_i\ 0$ for $i=1,\ldots,m\}$.

Theorem 1.4. *Let* $k=|\{\bar{\sigma}\in\{<,>,=\}^m, I_{\bar{\sigma}}\neq\emptyset\}|$. *Then D can be strongly simulated by a CT with depth* $O(T\log(k))$.

Proof. For $\bar{\sigma}\in\{<,>,=\}^m$ with $I_{\bar{\sigma}}\neq\emptyset$ let $x_{\bar{\sigma}}\in I_{\bar{\sigma}}$. By Lemma 1.2 we know that there is a CT D' with depth $O(T.\log(k))$ which strongly simulates D if we only allow inputs from $\{x_{\bar{\sigma}}, I_{\bar{\sigma}}\neq\emptyset\}$. The proof of the following claim will also prove the theorem.

Claim 1.5. *D' recognizes L (for all inputs from A).*

Proof. Let $x\in A$. As the nonempty $I_{\bar{\sigma}}$'s partition A, there is a unique $\bar{\sigma}$ with $x\in I_{\bar{\sigma}}$. By the definition of $I_{\bar{\sigma}}$, no query defined by a function from F can distinguish between x and $x_{\bar{\sigma}}$. Thus, both in D and D', the same computation paths are followed by x and $x_{\bar{\sigma}}$. Therefore, D accepts x if and only if it accepts $x_{\bar{\sigma}}$, i.e., $x\in L$ if and only if $x_{\bar{\sigma}}\in L$. By the same argument we get that D' accepts x if and only if it accepts $x_{\bar{\sigma}}$. Thus, D' accepts L. □

2. The main result

In this section we prove our results for algebraic computation trees and linear search algorithms. A *probabilistic algebraic computation tree* (PACT) D is a tree with degree 0, 1, or 2. To each node v with degree 1 a function $f_v: R^n\to R$ is attached. f_v is either a projection on one of the input variables $x_1,\ldots,x_n$, or a constant, or $f_v=f_{v'}\ \$\ f_{v''}$ for some nodes v', v'' on the path to v, and for $\$\in\{+,-,*,/\}$. A node v with degree 2 is either a probabilistic node or a query node. Probabilistic nodes work as in PCTs. At a query node v, a query $f_{v'}(\bar{x})\ \sigma\ 0$ is asked to determine which branch to follow, where v' is a node on the path to v and $\sigma\in\{<,>,=\}$. The leaves are accepting or rejecting. The recognized language, the complexity, and the threshold of D, as well as the deterministic version (ACT) are defined as for PCTs. Note that, in contrast to the ACTs introduced by Ben-Or [2], we do not allow the extraction of roots as arithmetic operation.

A *probabilistic or deterministic linear search algorithm* (PLSA or LSA) is a PACT or ACT, in which only the arithmetic operations $+$, $-$, and multiplication with real constants are allowed, and in which only queries and coin flips count for the complexity. (We shall see in the proof of the Main Theorem that we need a bound on the degrees of the computed rational functions. Therefore arithmetic operations have to count for the complexity, if multiplication or division are allowed. But with addition and subtraction only linear functions can be computed. Hence, in this case, we even have a degree bound (namely 1), if we do not count arithmetic operations, as in LSA's.)

We are now ready to state the main result of this paper.

Main Theorem 2.1. *Let D be a PACT or PLSA with complexity T recognizing $L \subset R^n$ with threshold $\frac{1}{2}+\varepsilon$. Then there is an ACT or LSA recognizing L in $O(T^2 n)$ steps.*

Proof. We only prove the result for ACTs. A simplified version of this proof already yields the result for LSAs.

Let D be an ACT as in the Main Theorem 2.1. By Lemma 1.1 we may assume w.l.o.g. that D has depth T. Let $F = \{f_1, \dots, f_m\}$ be the set of functions computed at the nodes of D. Then $m \leq 2^T$, because D is a binary tree. Furthermore, because of the arithmetic operations allowed in D, each f_i is a rational function, $f_i = r_i/q_i$, where r_i and q_i are polynomials of degree at most 2^T.

For $\bar{\sigma} \in \{<, >, =\}^m$ let $I_{\bar{\sigma}}$ be defined as in the previous section. We need the following claim.

Claim 2.2. *Let $k = |\{\bar{\sigma} \in \{<, >, =\}^m, I_{\bar{\sigma}} \neq \emptyset\}|$. Then there is an ACT recognizing L in $O(T \log(k))$ steps.*

Proof. By Theorem 1.4, D can be strongly simulated by a CT D' of depth $O(T \log(k))$. The definition of 'strongly simulating' guarantees that D' is an ACT. □

Proof of Theorem 2.1 (*continued*). Now it remains to bound k. For this purpose we first note that for $\sigma \in \{<, >, =\}$ and for $\bar{x} \in R^n$ such that $f_i(\bar{x})$ is defined, $f_i(\bar{x}) \, \sigma \, 0$ holds if and only if $p_i \, \sigma \, 0$, where $p_i := r_i q_i$. Thus, for $\bar{\sigma} \in \{<, >, =\}^m$, $I_{\bar{\sigma}} = \{\bar{x} \in R^n, f_i(\bar{x}) \, \sigma_i \, 0 \text{ for } i = 1, \dots, m\} = \{\bar{x} \in R^n, p_i(\bar{x}) \, \sigma_i \, 0 \text{ for } i = 1, \dots, m\}$. We need the following lemma.

Lemma 2.3. $k = |\{\bar{\sigma} \in \{<, >, =\}^m, I_{\bar{\sigma}} \neq \emptyset\}| \leq (2d+2)(2d+1)^{n-1}$, *where d is the degree of $\prod_{i=1}^m p_i$.*

Before we prove Lemma 2.3 we conclude the Main Theorem 2.1 from it. As the p_i's have degree at most 2^{T+1} and as $m \leq 2^T$, $d \leq 2^{2T+1}$. Thus $k = 2^{O(Tn)}$. Inserting this in Claim 2.2 yields the Main Theorem for ACTs. □

Proof of Lemma 2.3. This proof is based on the following theorem due to Milnor [10].

Theorem 2.4 (Milnor [10]). *Let $p: R^n \to R$ be a polynomial with degree d'. Then $c(p) := \{\bar{x} \in R^n, p(\bar{x}) \neq 0\}$ has at most $(d'+2)(d'+1)^{n-1}$ connected components.*

Now, in order to prove Lemma 2.3, let $A \subset R^n$ contain exactly one element of each nonempty $I_{\bar{\sigma}}$. Then, $|A| = k$. Let $\delta > 0$ be chosen such that $\delta < \min\{|p_i(\bar{x})|, i = 1, \dots, m, \bar{x} \in A, p_i(\bar{x}) \neq 0\}$. Let $\tilde{p} := \prod_{i=1}^m (p_i + \delta)(p_i - \delta)$. We need the following claim.

Claim 2.5. *Each connected component of $c(\tilde{p})$ contains at most one element from A.*

Proof. Let $\bar{x}, \bar{y} \in A$. Then, as $\bar{x}$ and $\bar{y}$ belong to different $I_{\bar{\sigma}}$'s, there is some p_i such that $p_i(\bar{x}) > 0$, $p_i(\bar{y}) \leq 0$ or $p_i(\bar{x}) \geq 0$, $p_i(\bar{y}) < 0$. In the first case, by the definition of δ, we get that $p_i(\bar{x}) - \delta > 0$ and $p_i(\bar{y}) - \delta < 0$. As $p_i - \delta$ is continuous, each continuous path from $\bar{x}$ to $\bar{y}$ contains a root of $p_i - \delta$, and therefore of $\tilde{p}$. Thus, $\bar{x}$ and $\bar{y}$ belong to different connected components of $c(\tilde{p})$. The second case is handled analogously with the help of $p_i + \delta$. □

Proof of Lemma 2.3 (*continued*). Since $d' \leq 2d$ (recall that d is the degree of p), by Milnor's theorem, $c(\tilde{p})$ has at most $(2d+2)(2d+1)^{n-1}$ connected components. As, by Claim 2.5, $k = |A| \leq$ (number of connected components of $c(\tilde{p})$), Lemma 2.3 follows. □

References

[1] C.H. Bennett and J. Gill, Relative to a random oracle, $P^A \neq NP^A \neq$ co-NP^A with probability 1, *SIAM J. Comput.* **10** (1981) 96-113.

[2] M. Ben Or, Lower bounds for algebraic computation trees, *15th ACM STOC* (1983) 80-86.

[3] D. Dobkin and R.J. Lipton, A lower bound of $\frac{1}{2}n^2$ on linear search algorithms for the knapsack problem, *J. Comput. System. Sci.* **16** (1978) 413-416.

[4] W. Feller, *An Introduction to Probability Theory and its Applications* (Wiley, New York, 1957).

[5] P. Klein and F. Meyer auf der Heide, A lower time bound for the knapsack problem on random access machines, *Acta Inform.* **19** (1983) 385-395.

[6] U. Manber and M. Tompa, Probabilistic, nondeterministic, and alternating decision trees, *14th ACM STOC* (1982) 234-244.

[7] F. Meyer auf der Heide, Lower bounds for solving Diophantine equations on random access machines, *J. ACM* **32**(4) (1985) 929-937.

[8] F. Meyer auf der Heide, A polynomial linear search algorithm for the n-dimensional knapsack problem, *J. ACM* **31** (3) (1984) 668-676.

[9] F. Meyer auf der Heide, Fast algorithms for n-dimensional restrictions of hard problems, *17th ACM STOC* (1985) 413-420.

[10] J. Milnor, *Singular Points of Complex Hypersurfaces* (Princeton Univ. Press, 1968).

[11] J.H. Reif, On synchronous parallel computations with independent probabilistic choice, *SIAM J. Comput.* **13** (1) (1984) 46-56.

[12] E. Reingold, On the optimality of some set algorithms, *J. ACM* **19** (1972) 649-659.

[13] M. Snir, Lower bounds for probabilistic linear decision trees, Res. Rept. 83-6, Dept. of Computer Science, Hebrew Univ. of Jerusalem, Israel, 1983.