Fast Algorithms for N-Dimensional Restrictions of Hard Problems

FRIEDHELM MEYER AUF DER HEIDE

Johann Wolfgang Goethe Universität Frankfurt, Frankfurt am Main, West Germany

Abstract. Let M be a parallel RAM with p processors and arithmetic operations addition and subtraction recognizing $L \subset N^n$ in T steps. (Inputs for M are given integer by integer, not bit by bit.) Then L can be recognized by a (sequential!) linear search algorithm (LSA) in $O(n^4(\log(n) + T + \log(p)))$ steps. Thus many n-dimensional restrictions of NP-complete problems (binary programming, traveling salesman problem, etc.) and even that of the uniquely optimum traveling salesman problem, which is Δ_2^p-complete, can be solved in polynomial time by an LSA. This result generalizes the construction of a polynomial LSA for the n-dimensional restriction of the knapsack problem previously shown by the author, and destroys the hope of proving nonpolynomial lower bounds on LSAs for any problem that can be recognized by a PRAM as above with $2^{\text{poly}(n)}$ processors in poly(n) time.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*geometrical problems and computations*

General Terms: Algorithms, Theory, Verification

Additional Key Words and Phrases: Linear search algorithms, parallel random-access machines

1. *Introduction*

Linear search algorithms (LSAs) have turned out to be a realistic and comfortable computation model for proving lower time bounds for a large variety of interesting problems. Such an algorithm is an abstraction of a random-access machine (RAM). These RAMs have the capability of executing direct or indirect storage access, conditional branchings, addition, or subtraction in one step (see [1] or, tailored to our use, [5], [7], or [12]). They recognize languages $L \subset N^n$ or $L \subset N^*$ and read the input integer by integer, not bit by bit.

When dealing with LSAs, one usually allows inputs consisting of n real numbers, only counts conditional branchings as computation steps, and assumes that they all are of the form "If $f(\bar{x}) > 0$, then $\cdots$ else $\cdots$," where $\bar{x} \in R^n$ is the input and $f: R^n \to R$ is an affine function, that is, $f(\bar{x}) = \bar{a}\bar{x} - b$ for some $\bar{a} \in R^n$, $b \in R$. A conditional branching as above is said to be defined by f. We present LSAs by rooted binary trees whose root and inner nodes are labeled with predicates of the form "$f(\bar{x}) > 0$" for some f as above. The leaves are labeled with "accept" or "reject." A computation started with some input $\bar{x} \in R^n$ consists of traversing the

This research was done at the IBM Research Laboratory, San Jose, California.

Author's present address: Fachbereich Informatik, Universität Dortmund, Postfach 50 05 00, D-4600 Dortmund 50, West Germany.

tree from the root to a leaf, always choosing the left or right branch of a node according to whether its predicate is fulfilled or not. The LSA accepts the language of all inputs arriving at an accepting leaf.

The reason why LSAs are comfortable for proving lower bounds is their nice geometrical structure. As each conditional branching defined by some affine function $f: R^n \to R$ subdivides the input set in the two halfspaces $\{\bar{x} \in R^n \mid f(\bar{x}) >$ (respectively, $\leq$) $0\}$ of the hyperplane $\{\bar{x} \in R^n \mid f(\bar{x}) = 0\}$, the inputs arriving at some node of an LSA form a (convex) polytope. Lower bounds for LSAs can be found, for example, in [3], [7], or [11].

Furthermore, many lower bounds for LSAs can be carried over to RAMs (see [5], [12], and, for a general lower bound for RAMs, [7, theorem 3]), that is, to an important, realistic computation model.

In [6] it is shown by the author that LSAs are surprisingly strong. They can solve the NP-complete (compare [4]) knapsack problem (input: $\bar{x} \in R_+^n$, query: $\exists \bar{\alpha} \in \{0, 1\}^n$ such that $\bar{\alpha}\bar{x} = 1$) in polynomial time. The reason for this strength of LSAs is that they only deal with inputs consisting of a fixed number n of variables, that is, they only handle n-dimensional restrictions of given problems. Thus, although a program solving, say, the knapsack problem must have constant size (independent of the number of input variables), the size of an LSA for its n-dimensional restriction may depend on n. In fact, the size of the LSA shown in [6] is exponential in n. Its main importance is to pinpoint the limits of proving large lower bounds on LSAs.

In this paper we take a closer look at these limits. For this purpose, we consider parallel RAMs (PRAMs). Such a PRAM consists of p RAMs as described above, its processors, and an additional common memory consisting of infinitely many registers, which can be accessed directly or indirectly by the processors. We note here that this model can not be compared with the usual PRAM model used in complexity theory. The main difference lies in the fact that we assume that the input is given integer by integer, not bit by bit, and that we measure the complexity as a function of the number of input variables, not of their binary length. A minor difference is that we allow the processors to be awakened in one step. This makes it possible for more than 2^t processors to execute a computation of length t.

The main result of this paper is the following:

> *If $L \subset N^n$ can be recognized by a PRAM with p processors in T steps, then a (sequential!) LSA can recognize L in time polynomial in n, T, and $\log(p)$.*

Now one can show the existence of polynomial LSAs for given problems by simply designing polynomial parallel algorithms for them using $2^{\text{poly}(n)}$ processors. This method yields, for example, polynomial LSAs for NP-complete problems such as the traveling salesman problem, binary programming, and integer programming with bounded solution size. Furthermore, we obtain a polynomial LSA for the problem of deciding whether an instance of the traveling salesman problem has a unique optimal solution. This problem is shown to be complete in Δ_2^P, the class of all problems that can be solved in polynomial time using oracles from NP (see [10]).

The paper is organized as follows. In Section 2, we define PRAMs in more detail, state our theorem, and show applications. In Section 3, we outline the proof, state the two basic lemmas, and conclude the theorem from them. Sections 4 and 5 contain the proofs of the two lemmas mentioned above. In Section 6, some open problems are discussed.

2. *Definitions and Results*

A PRAM M consists of a finite number p of processors $P_1, \ldots, P_p$ and a common memory. The common memory consists of infinitely many registers $m_0, m_1, \ldots$, each capable of storing one integer. Each processor P_i is a usual RAM with the capabilities of accessing directly or indirectly a common register or one of its infinitely many private registers $m_{i,0}, m_{i,1}, \ldots$, adding or subtracting the contents of two registers, and executing a conditional branching in one step. We assume that the PRAM is synchronized and that all processors are awakened at the beginning in one step. In the following we describe in more detail a computation step of a PRAM. Let $\#m_j$, $\#m_{i,j}$ denote the current contents of the respective registers.

In one step, each processor P_i executes three phases:

(1) *Conditional Branching.* P_i branches according to "$\#m_{i,0} > 0$".

(2) *Write.* P_i writes k or $\#m_{i,0}$ into the private or common register with address j or $\#m_{i,j}$. Here k and j are again constants from $\{0, \ldots, p\}$.

(3) *Read/Arithmetic.* P_i reads a constant k, or $\#m_j$, or $\#m_{i,j}$, or $\#m_{\#m_{i,j}}$, or $\#m_{i,\#m_{i,j}}$ into $m_{i,0}$. Here j and k are again constants from $\{0, \ldots, p\}$. P_i can also execute an addition or subtraction after the read. In this case, the value to be read is added to or subtracted from the old content of $m_{i,0}$.

At the beginning of a computation the input $(x_1, \ldots, x_n) \in N^n$ is stored in $m_1, \ldots, m_n$. All other registers contain 0. A PRAM M computes a function of $f\colon N^n \to N$ in T steps if M started with an input $(x_1, \ldots, x_n)$ has computed $f(x_1, \ldots, x_n)$ after T steps in $m_{1,0}$. M recognizes $L \subset N^n$ in T steps if it computes the characteristic function of L in T steps.

We prove the following theorem in this paper.

THEOREM. *Let $L \subset N^n$ be recognized by a PRAM with p processors in T steps. Then L can also be recognized by an LSA in $6n^4(\log(n) + T + \log(p)) + O(n^3)$ steps.*

We now give some applications of this theorem. All algorithms for PRAMs mentioned below are trivial, and are not explained in this paper.

First we consider some NP-complete problems (compare [4]).

> *Integer Programming with Solutions $\leq k$* (n variables, m inequalities). (Input: m linear inequalities with variables $x_1, \ldots, x_n$. Query: $\exists (x_1, \ldots, x_n) \in \{0, \ldots, k\}^n$ which fulfills all the inequalities.)

(Note that, for $k = 1$, we have defined the *Binary Programming* problem.)

This problem can be solved by a PRAM with $(k+1)^n$ processors in $O(nm)$ steps and thus by an LSA (for inputs from $R^{(n+1)m}$) in $O((nm)^4(nm + n\log(k+1)))$ steps. (Note that this remains polynomial in n and m as long as k is at most $2^{\text{poly}(n,m)}$.)

We note here that in [7] an $\Omega(n^2\log(k+1))$ lower bound for $m = 1$ on LSAs and RAMs is shown generalizing the bounds from [3] and [5] for $m = 1$, $k = 1$.

> *Traveling Salesman Problem* (n *towns*) (Input: Distances between all pairs of towns, and a number k. Query: Is there a roundtrip visiting each town exactly once with total length at most k?)

This problem can be solved by a PRAM with $n!$ processors in $O(n\log(n))$ steps and thus by an LSA (for inputs from R^{n^2}) in $O(n^9\log(n))$ steps.

We now consider the following generalization of the traveling salesman problem:

Uniquely Optimum Traveling Salesman Problem (n *towns*). (Input: $n \times n$-matrix of distances between all pairs of towns. Query: Is there a unique shortest round trip visiting each town exactly once?)

This problem is shown in [10] to be Δ_2^p-complete. Also this problem can obviously be solved by a PRAM with $n!$ processors in $O(n\log(n))$ steps and thus by an LSA in $O(n^9\log(n))$ steps.

3. *Proof of the Theorem*

The proof of the theorem is not done by simulating a PRAM step by step by an LSA. Instead we do the following: We first show that languages recognized by a PRAM with p processors in T steps have a certain structure; we call them *q-languages* where q denotes a parameter dependent on t and p. Then we show that q-languages can be recognized fast by LSAs.

In order to define q-languages, let $F = \{f_1, \ldots, f_m\}$ be a set of affine functions f_i: $R^n \to R$. Then the following languages are called *F-languages*:

(1) $\{\bar{x} \in R^n \mid f_i(\bar{x}) : 0\}$, $i = 1, \ldots, m$, ":" $\in \{=, <, >\}$, are F-languages.
(2) Unions and intersections of F-languages are F-languages.

In order to interpret F-languages geometrically and for later use, we now give some geometrical definitions. Let f: $R^n \to R$ be an affine function. The hyperplane $\{\bar{x} \in R^n \mid f(\bar{x}) = 0\}$ and its halfspaces are said to be defined by f. The intersection of a finite number of halfspaces and hyperplanes is a (convex) polytope. Let F be the set of affine functions described above, and let $H_1, \ldots, H_m$ be the hyperplanes defined by $f_1, \ldots, f_m$. Let H_i^+, H_i^- denote the left and the right halfspace of H_i. For each disjoint partition $A \cup B \cup C$ of $\{1, \ldots, m\}$ (A, B, or C may be empty), $\bigcap_{i\in A} H_i \cap \bigcap_{i\in B} H_i^+ \cap \bigcap_{i\in C} H_i^-$ is called a face of F, or a face of the language $\bigcup_{i=1}^m H_i$.

With these definitions it is easily seen that each F-language is a union of some faces of F.

Now let F_q denote the set of all affine functions f: $R^n \to R$ with coefficients from $\{-q, \ldots, q\}$. An F_q-language is called a q-language.

The proof of the theorem is based on the following two lemmas.

LEMMA 1. *Let M be a PRAM with p processors recognizing $L \subset N^n$ in T steps. Then there is a $p2^t$-language L' with $L = L' \cap N^n$.*

LEMMA 2. *Each q-language can be recognized by an LSA in $6n^4(\log(n) + \log(q)) + O(n^3)$ steps.*

The proof of the theorem is now done by inserting the bound for q from Lemma 1 into Lemma 2. Therefore it remains to prove these two lemmas.

4. *Proof of Lemma* 1

Let M, L, T, p be as in the lemma. A computation pattern of M started with some input $\bar{x}$ consists of the description of the results of the conditional branchings, the information whether $\bar{x} \in L$ or not, and the communication pattern of M started with $\bar{x}$. The communication pattern specifies for each $t \in \{1, \ldots, T\}$, $i \in \{1, \ldots, p\}$, which processor wrote at which time what P_i reads in step t, or which input variable or which constant from $\{1, \ldots, p\}$ P_i reads in step t if M is started with input $\bar{x}$.

Let $C_1, \ldots, C_r$ be all possible computation patterns, and let $U_1, \ldots, U_r$ denote the sets of inputs with the respective computation pattern.

CLAIM 1. *Each U_i is a $p2^t$-language.*

Before we prove this claim, we derive Lemma 1 from it.

Since the U_i's form a disjoint partition of N^n, and since by definition either $U_i \subset L$ or $U_i \cap L = \emptyset$, L is the union of all U_i's with $U_i \subset L$. By claim 1, all U_i's are $p2^t$-languages; therefore L is one, too. Q.E.D.

PROOF OF CLAIM 1. Let $C = C_i$ be one of the computation patterns, $U = U_i$. For $t \in \{1, \ldots, T\}$, let U^t denote the set of inputs following the computation pattern C for t steps.

In a computation pattern the sequence of instructions is fixed because the outcomes of the conditional branches are fixed. Therefore, for inputs from U^t, whether a processor reads or writes in some step in the private or common memory is fixed, as well as whether it uses a constant or an indirect address. Whether a constant, an input variable, or a previously written value is read is also fixed, as well as whether an addition or subtraction is executed.

Now let $f_{i,t}(\bar{x})$ denote the content of $m_{i,0}$ before step t, and let $r_{i,t}(\bar{x})$, $w_{i,t}(\bar{x})$ denote the addresses used for reading and writing in step t by P_i, if M is started with $\bar{x}$.

We now note the following: $f_{i,0} \equiv 0$ and for $t > 0$, $f_{i,t} \mid_{U^{t-1}}$ is either a constant from $\{0, \ldots, p\}$, or some x_j, or of the form $f_{j,t'} \mid_{U^{t-1}}$ or $(f_{i,t} + f_{j,t'}) \mid_{U^{t-1}}$ for some j, t' fixed by C. Since $r_{i,t}$, $w_{i,t}$ were also previously computed as some $f_{j,t'}$ for some j, t' fixed by C, a straightforward induction shows:

(*) For each $i, t, f_{i,t} \mid_{U^{t-1}}, r_{i,t} \mid_{U^{t-1}}, w_{i,t} \mid_{U^{t-1}} \in F_{p2^{t-1}}$.

(For a function $g: N^n \to N$ and a set $V \subset N^n$, $g \mid_V \in F_q$ means that there is $g' \in F_q$ with $g \mid_V \equiv g' \mid_V$.)

Now we can clarify the structure of the sets U^t.

(**) Each U^t is a $p2^t$-language.

PROOF. By induction on t.

$U^0 = N^n$ is a p-language.

Let $t > 0$. U^t is the maximal set fulfilling the following properties:

(a) $U^t \subset U^{t-1}$

(b) For each $i \in \{1, \ldots, p\}$, $U^t \subset V_i = \{\bar{x} \in N^n \mid f_{i,t}(\bar{x}) > (\leq)\ 0\}$, where $>$, $\leq$ is chosen according to C. (C fixes the outcomes of conditional branchings.)

(c) For each $i \in \{1, \ldots, p\}$, $U^t \subset W_i = \{\bar{x} \in N^n \mid r_{i,t}(\bar{x}) - w_{j,t'}(\bar{x}) = 0\}$, where $j \in \{1, \ldots, p\}$ and $t' \in \{1, \ldots, t-1\}$ are fixed by C. (C fixes the communication pattern.)

By (*), each V_i is a $p2^{t-1}$-language. Also, by (*), $r_{i,t} \mid_{U^{t-1}}$ and $w_{j,t'} \mid_{U^{t-1}} \in F_{2p^{t-1}}$. Therefore $(r_{i,t} - w_{j,t'}) \mid_{U^{t-1}} \in F_{p2^t}$ and W_i is a $p2^t$-language. Thus $U^t = U^{t-1} \cap \bigcap_{i=1}^{p} V_i \cap \bigcap_{i=1}^{p} W_i$ is a $p2^t$-language. Q.E.D.

The lemma now can be concluded as follows. Since $U = U^T \cap \{\bar{x} \in N^n \mid f_{i,T}(\bar{x}) = 1 \text{ (or } 0)\}$, ($f_{1,t}(\bar{x})$ is the output), and since by (*), $f_{i,T} \mid_{U^T} \in Fp2^{T-1}$, U is a $p2^T$-language. Q.E.D.

5. *Proof of Lemma* 2

This proof is based on [6, theorem 2].

THEOREM [6]. *Let $H_1, \ldots, H_m$ be hyperplanes in R^n, $H_i = \{\bar{x} \in R^n \mid f_i(\bar{x}) = 0\}$ for some $f_i \in F_q$, and $L = \bigcup_{i=1}^m H_i$. Then $[-1, 1]^n \cap L$ can be recognized by an LSA in $3n^4(\log(n) + \log(q)) + O(n^3)$ steps.*

In order to prove Lemma 2, we first show

CLAIM 2. *The above theorem also holds, if L instead of $[-1, 1]^n \cap L$ has to be recognized.*

PROOF. For an affine function $f: R^n \to R$ with $f(\bar{x}) = \bar{a}\bar{x} - b$ for some $\bar{a} \in R^n$, $b \in R$, let $\tilde{f}: R^{n+1} \to R$ be defined by $\tilde{f}(\bar{x}, x_{n+1}) = \bar{a}\bar{x} - bx_{n+1}$. Now let $f_1, \ldots, f_m$, $H_1, \ldots, H_m$ be as in the above theorem from [6], $\tilde{H}_1, \ldots, \tilde{H}_m$ be the linear hyperplanes in R^{n+1} defined by $\tilde{f}_1, \ldots, \tilde{f}_m$, and $\tilde{L} = \bigcup_{i=1}^m \tilde{H}_i$. It suffices to prove the theorem for $\tilde{L}$, because an LSA recognizing $\tilde{L}$ recognizes L if we substitute 1 for x_{n+1}. $\tilde{L}$ consists of linear hyperplanes; that is, all $\tilde{H}_i$'s contain $\bar{0}$.

Now let $P_j^+(P_j^-) = \{\bar{x} \in R^{n+1} \mid x_j > (<) 0$ and $|x_j| = \max\{|x_1|, \ldots, |x_{n+1}|\}\}$ and $\tilde{L}_j^{\pm} = \tilde{L} \cap P_j^{\pm}$. Then by Lemma 1 from [6], $\tilde{L}_j^{\pm}$ can be recognized as fast as $\tilde{L}_j^{\pm} \cap \{\bar{x} \in R^n \mid x_j = \pm 1\} = L_j^{\pm} \cap [-1, 1]^n$, where $L_j^{\pm}$ is the union of the hyperplanes in R^n $(= R^{n+1}|_{x_j=\pm 1})$ defined by $f_{ij}^{\pm} = \tilde{f}_i|_{x_j=\pm 1}$. Since f_i belongs to F_q, $f_{ij}^{\pm}$ does, too, Thus $L_j^{\pm}$, and therefore $\tilde{L}_j^{\pm}$, too, can be recognized in $3n^4(\log(n) + \log(q)) + O(n^3)$ steps because of the theorem from [6]. Therefore, the following LSA recognizes L as fast as desired in claim 2:

—Decide in which $P_j^{\pm}$ the input $\bar{x}$ lies. Suppose $\bar{x} \in P_j^{\pm}$.
—Use the above LSA for recognizing $L \cap P_j^{\pm}$.

The first part of this algorithm needs $2(n + 1)$ steps, and the second part needs $3n^4(\log(n) + \log(q)) + O(n^3)$ steps, as shown above. Q.E.D.

Claim 2 already proves Lemma 2 for special q-languages, namely, those that consist of hyperplanes defined by functions from F_q. We shall now construct LSAs for arbitrary q-languages from the above LSAs for q-languages consisting of hyperplanes. For this purpose let $H_1, \ldots, H_m$ be arbitrary hyperplanes in R^n and $L = \bigcup_{i=1}^m H_i$. We say that an LSA partitions R^n according to L if for each leaf v of the LSA the set of inputs arriving at v is a subset of a face of L (cf. Section 3).

CLAIM 3. *If L can be recognized by an LSA in T steps, then R^n can be partitioned according to L in $2T$ steps.*

Claims 2 and 3 imply Lemma 2 as follows. Let L be a q-language and L_q the language consisting of all hyperplanes defined by functions from F_q. Then, by Claim 2, L_q can be recognized by an LSA in $T = 3n^4(\log(n) + \log(q)) + O(n^3)$ steps. Thus, by Claim 3, R^n can be partitioned according to L_q in $2T$ steps by some LSA. By the definition of a q-language, L is the union of faces of L_q. Therefore we obtain an LSA of depth $2T$ for L by attaching "accept" to all leaves v of D for which the set of inputs arriving at v is a subset of a face of L_q belonging to L, and attaching "reject" to the other leaves of D. Thus it remains to prove Claim 3.

PROOF OF CLAIM 3. A 3-way LSA D is a 3-ary tree whose nodes are labeled with affine functions $f: R^n \to R$. An input $\bar{x} \in R^n$ arriving at such a node chooses the left (middle, right) branch, if $f(\bar{x}) > (=, <) 0$. The leaves are labeled with "accept" or "reject." The set of inputs arriving at a node v is called $c(v)$. D accepts the union of all sets $c(v)$ for accepting leaves v of D.

SUBCLAIM. *Let D be a 3-way LSA accepting L in T steps. Then there is a 3-way LSA D' with depth T and the following property: For each leaf v of D' with*

$c(v) \subset L$, there is a leaf v' of D' for which $c(v') \cap L$ is empty, such that $c(v)$ is a face of (the polytope) $c(v')$.

PROOF. Let D be as above. We may assume without loss of generality that v is a leaf whenever $c(v) \subset L$. Then an accepting leaf is always reached via the middle branch of its father. The following algorithm constructs D' from D.

(1) Mark all fathers of accepting leaves.
(2) As long as there is a marked node v whose right son v'' is no leaf, replace its middle son by a copy of the subtree with root v''.
(3) Remove all labels (accept or reject) from the leaves.

This algorithm stops because it changes neither the maximum degree nor the depth of the tree but adds at least one node to it in each step of (2).

Now let v be a leaf of D' with $c(v) \subset L$. Traverse the path from the root to v until the first time it chooses the middle branch of a marked node. At this point choose the right branch instead, and go on in the copy of the path to v in the subtree of this branch in the same way. Let this path lead to the leaf v'. Then $c(v)$ and $c(v')$ differ exactly by the fact that restrictions from marked nodes of the form $f(\bar{x}) = 0$ defining $c(v)$ are replaced by $f(\bar{x}) < 0$ in the definition of $c(v')$. But this just means that $c(v)$ is a face of $c(v')$. Furthermore $c(v') \cap L$ is empty because inputs from L choose a path in D' on which at least at one marked node the middle branch is chosen. By construction this is not the case in the path to v'. Q.E.D.

To prove Claim 3, we now show that the above 3-way LSA D' partitions R^n according to L. Since D' can obviously be simulated by an LSA in $2T$ steps, this implies Claim 3.

Let v be a leaf of D'. If $c(v) \cap L$ is empty, then $c(v)$ belongs to a connected component of $R^n - L$, which is by definition a face of L. If $c(v) \subset L$, then by the subclaim $c(v)$ is a face of $c(v')$ for some leaf v' for which $c(v') \cap L$ is empty. As each face of a polytope contained in one of the polytopes $R^n - L$ consists of is a subset of a face of L, Claim 3 follows. Q.E.D.

6. *Conclusion*

We have shown in this paper that the n-dimensional restriction of many languages can be recognized surprisingly fast. This result motivates the following questions.

(1) The set F_q consists of $m_q = O(q^n)$ functions, that is, L_q, the union of all the hyperplanes defined by functions from F_q, consists of m_q hyperplanes. Our result shows that L_q can be recognized in poly(log(m_q)) steps. In order to understand the power of LSAs, it is interesting to find out whether LSAs can recognize every union of m hyperplanes in R^n in poly(log(m), n) steps, or whether there exist "hard" versions of such languages.

(2) By Ben Or's result from [2], most known lower bounds for LSAs also hold for algebraic computation trees (ACTs) in which multiplication and division are allowed. Also, ACTs only deal with n-dimensional restrictions of problems. For ACTs it is shown by the author in [9] that at least their deterministic and probabilistic versions are polynomially related. Is this also true for the relation between their (deterministic) sequential and parallel versions (as shown for LSAs in this paper)?

(3) The reason why LSAs are so fast is because the length of LSAs may depend on n, the number of input variables, whereas the length of "usual" programs is bounded independently of the input. This means that for a "bounded program

length" version of LSAs both the tree and the set of functions attached to its nodes have a lot of structure. It would be of greatest interest to explore this structure and derive lower bound arguments from it.

REFERENCES

1. AHO, A. V., HOPCROFT, J. E., AND ULLMAN, J. D. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, Mass., 1974.
2. BEN OR, M. Lower bounds for algebraic computation trees. In *Proceedings of the 15th ACM Symposium on Theory of Computing* (Boston, Mass., Apr. 25–27). ACM, New York, 1983, pp. 80–86.
3. DOBKIN, D. AND LIPTON, R. J. A lower bound of $\frac{1}{2}n^2$ on linear search programs for the knapsack problem. *J. Comput. Syst. Sci. 16* (1978), 413–416.
4. GAREY, M. R., AND JOHNSON, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, 1979.
5. KLEIN, P., AND MEYER AUF DER HEIDE, F. A lower time bound for the knapsack problem on random access machines. *Acta. Inf. 19* (1983), 385–395.
6. MEYER AUF DER HEIDE, F. A polynomial linear search algorithm for the n-dimensional knapsack problem. *J. ACM 31*, 3 (July 1984), 668–676.
7. MEYER AUF DER HEIDE, F. Lower bounds for solving linear Diophantine equations on random access machines. *J. ACM 32*, 4 (Oct. 1985), 929–937.
8. MEYER AUF DER HEIDE, F. Lower time bounds for testing the solvability of Diophantine equations on several parallel computational models. *Inf. Control 67* (1985), 195–211.
9. MEYER AUF DER HEIDE, F. Simulating probabilistic by deterministic algebraic computation trees. *Theoret. Comput. Sci. 41* (1985), 325–330.
10. PAPADIMITRIOU, C. H. On the complexity of unique solutions. *J. ACM 31*, 2 (Apr. 1984), 392–400.
11. REINGOLD, E. On the optimality of some set algorithms. *J. ACM 19*, 4 (Oct. 1972), 649–659.
12. SIMON, J., AND PAUL, W. J. Decision trees and random access machines. In *Monographic 30, L'Enseignement Mathematique, Logic et Algorithmic.* Univ. Geneve, Switzerland, 1982, pp. 331–340.

RECEIVED MARCH 1985; REVISED OCTOBER 1986, AUGUST 1987; ACCEPTED AUGUST 1987