NEARLY SHARP COMPLEXITY BOUNDS FOR MULTIPROCESSOR ALGEBRAIC COMPUTATIONS

Dima Grigoriev*
Department of Computer Science and
Department of Mathematics
Penn State University
University Park, PA 16802 USA
email: dima@cse.psu.edu

The complexity lower bound of $\Omega(\sqrt{\log N})$ was obtained ([MP93],[M94]) for recognizing a semialgebraic set with N connected components by some parallel computational models, like the accepting network or the algebraic PRAM. It is unknown whether a parallel computation has an advantage versus its sequential counterpart for this sort of problems, i.e. whether it could recognize a semialgebraic set faster than within complexity $0(\log N)$ (the complexity lower bound for sequential algebraic computation trees [SY82], [B83]). We introduce a computational model and call it multiprocessor algebraic computation which extends the notions of accepting network and algebraic PRAM. For this model an $\Omega(\sqrt{\log N})$ complexity lower bound still holds. We design a multiprocessor algebraic computation which recognizes a linear complex in $\mathbb{R}^n$ (i.e. a set given by a boolean combination of N linear inequalities) within the complexity $0(\sqrt{\log N \log\log N})$ for small n.

Introduction.

The theory of parallel algebraic computations differs in many aspects from its more common binary (so, based on Turing machines or RAM) counterpart (for a survey see [G86]). In [MP93] (see also [M94]) a complexity lower bound of $\Omega(\sqrt{\log N})$ was discovered

*Supported in part by NSF grant CCR-9424358.

for testing membership to a semialgebraic set with N connected components by means of accepting networks or parallel algebraic computation trees (for the case of linear trees one could get a similar result from [M88]). Comparing this bound with $\Omega(\log N)$ (the complexity lower bound for (sequential) algebraic computation trees [SY82], [B83]), this shows that for the model of algebraic computation trees the speed-up which could be achieved by parallel computations, is limited.

On the other hand it still remains unclear, whether this lower bound $\Omega(\sqrt{\log N})$ could be attained; in other words, whether parallel algebraic computation trees indeed have advantages with respect to their sequential counterparts, that is whether they can recognize a semialgebraic set faster than within complexity $O(\log N)$.

In this paper we introduce a computational model and call it multiprocessor algebraic computation which on the one hand is a special case of the very general concept of arithmetic network [G86]. On the other hand multiprocessor algebraic computations extend the usual models of algebraic PRAM ([M88], [M94]) or accepting network ([MP93]). The crucial new feature for multiprocessor algebraic computation is that we allow the use of polynomials from a shared pool, these polynomials are computed along with the whole computation (one can view these polynomials as reserved in the hard disc), and the number of processors is restricted (one could view the processors as the random access memory). The processors can be used for control needs, say for branching, in order to reach some node in the computation, as is usually done in computation trees (e.g. while recognizing membership to some set).

The main consequences of the paper are that we first show that the lower bound $\Omega(\sqrt{\log N})$ on the depth holds as well for our extended model of multiprocessor algebraic computation (corollary 2 in section 1). Secondly we prove the complexity upper bound $0(\sqrt{\log N \log\log N})$ for our model for the problem of membership to a linear complex (i.e. a set given by a boolean combination of N linear inequalities) for small dimensions n (theorem 2 in section 3). Note that the number of connected components of a linear complex is bounded by N^n (see e.g. [M64] or [G88]). The construction of the multiprocessor algebraic computation relevant for this purpose relies on the basic design of one for the problem of binary search with the depth $0(\sqrt{\log N \log\log N})$ (theorem 1 in section 2); the latter bound is also close to the complexity lower bound $\Omega(\sqrt{\log N})$ for binary search (see corollary 2 in section 1). Thus, multiprocessor algebraic computations can be faster in recognizing a semialgebraic set than the complexity lower bound $\Omega(\log N)$ for the sequential models.

In the last section the recent complexity lower bounds on testing membership to a polyhedron ([GKV95], [GKMS96]), are extended to multiprocessor algebraic computations (proposition 1), including the randomized version (proposition 2); thereby, these lower bounds hold as well for algebraic PRAM's and for parallel algebraic computation trees. Observe that corollary 2 in section 1 and other more general topological methods based on the sum of Betti numbers ([Y94], [MMP96], see also the discussion in section 1) cannot be applied to the problem of membership to a polyhedron because of the trivial topology of a polyhedron. At the end of the paper we introduce a modification of multiprocessor algebraic computations and call this model the multiprocessor algebraic decision-maker. It is in similar relation with respect to multiprocessor algebraic computations as algebraic decision trees are with respect to algebraic computation trees. We extend the complexity lower bounds for the problem of testing membership to a polyhedron to multiprocessor algebraic decision-makers, (proposition 3) including randomized ones (proposition 4).

1. Multiprocessor Algebraic Computation: Concept and Properties

We describe a multiprocessor algebraic computation (MAC), the computational model which we deal with in the paper. MAC is a special case of an arithmetic network [G86] (see also [MP93], [MMP96], [CG96]). An algebraic network is a directed acyclic graph which contains nodes of 3 types:

0) with the indegree 0 called input nodes labeled by either one of n input variables or constants;

(1) with the indegree 1 called sign-nodes and with an output equal to either 0 or 1, depending on the sign of the incoming node (for the case of a ground field being a subfield of $\mathbb{R}$ the sign is usually defined according to $\geq$, $<$, for an arbitrary ground field the sign is defined according to $=$, $\neq$);

(2) with the indegree 2 and labeled by some arithmetic operation.

A node with the outdegree 0 is called an output node and the output of the whole arithmetic network is defined by a function of its output nodes. If one considers a decision network then its output could be treated as a boolean function $\{0,1\}^m \to \{\text{accept,reject}\}$, where m is the number of all the output nodes which are sign-nodes. The number s of nodes of the network is called its size, the maximal length d of its paths is called the depth. The depth corresponds to parallel complexity.

If no further restrictions are imposed, the network becomes unrealistic as is illustrated by the following easy example showing that the knapsack problem could be accepted in logarithmic parallel time.

Example. Assume for simplicity that $n = 2^k$. Then for $\ell = 0, \dots, k$ the ℓ-th level of the network contains $2^{2^\ell} \cdot 2^{k-\ell}$ arithmetic nodes which compute the linear forms $\sum\limits_{i \in I} X_i$ for all possible subsets $I \subset \{q \cdot 2^\ell + 1, \dots, (q+1)2^\ell\}$ and $0 \leq q < 2^{k-\ell}$. For each of the computed linear forms f the network computes as an output the sign-node $sgn(f^2 - 1)$ (using 3 extra levels). So, in this example, in $0(\log n)$ parallel steps we take into account 2^n output nodes.

The usual restriction imposed on a network (see e.g. [MP93], [MMP96]) is that it contains a single output node (this model is called the accepting network model). This restriction immediately provides a bound on the the size $s \leq 2^d$. A similar uniform version of this model, namely algebraic PRAM's (for different sets of involved arithmetic operations, like $\{+,-,\times\}$ or $\{+,-\}$) was studied in [M88]. Another similar computational model was considered in [M94] .

The model which we introduce, MAC, extends the latter models. We define a MAC with p processors as a slightly modified arithmetic network of the depth d where each node consists of two parts: an algebraic part and an indicator. Each node has indegree either 0 (input node) or 2. The algebraic part contains a certain polynomial and is computed in the usual way, and an indicator is a boolean variables which could be in one of two states: "active" or "passive". To perform the computation, to each node in a MAC an arithmetic operation is attached (or an input variable or a constant in the case of an input node) which computes the algebraic part, and an (activating) boolean function. The value of the activating boolean function is always "passive" when both incoming nodes are "passive", otherwise it is actually a value of a boolean function of (one or two) signs of the algebraic parts of the incoming nodes which are "active". More precisely, one could think of 3 boolean functions for 3 possible situations, respectively: just one incoming node is "active", the other incoming node is "active", or both incoming nodes are "active". For every input node its indicator (so, "passive" or "active" state) is assigned. Thus, extra sign nodes are unnecessary, since they are already incorporated.

The main restriction is that for any input the number of "active" nodes at each step of the computation by a MAC is at most (some parameter) $p \leq 2^d$. Recall that the latter bound holds for the usual parallel computations. Here we need to impose some restriction on p to make the model more realistic.

We assume that the nodes of a MAC are naturally divided into $d+1$ levels. Level 0 consists just of the input nodes. The ℓ-th step of the computation is performed at the ℓ-th level of the MAC and the incoming nodes lie at the $(\ell-1)$-th level.

To complete the description of a MAC we assign an output function which provides an output (it could be "accept" or "reject" when we deal with the decision problems) for each subset of at most p nodes of the last (i.e. d-th) level. The output of the MAC is the value of the output function for the set of all "active" nodes.

The informal idea behind the concept of a MAC on the one hand, is to bound the number p of involved "active" nodes which play the role of the processors (in order to preclude computations like in the above example for the Knapsack problem where 2^n processors were involved), and on the other hand to have access to many functions computed (as the algebraic parts of the nodes of a MAC) independently from the inputs as a kind of preprocessing. One could view these functions as being stored in the hard disc (treated as a reserve) which could be used when necessary, in p processors in the random access memory. Each processor could be viewed like a node in a computation tree (where the next node is chosen as in the usual branching according to the signs of the computed polynomials) and the several processors could be treated as several trees. The destiny of the processors is eventually to reach the output nodes (in which the MAC reads an output).

The following observation shows that the concept of a MAC admits some generalizations. Namely, we could allow the arithmetic operation attached to a node to depend on the signs of the algebraic parts of the incoming nodes which are "active" (similar to the activating function). Thus, the algebraic part would become a piecewise polynomial function. One could transform such a modified MAC into a MAC as we defined it above without increasing the depth. Indeed, we start with the nodes at the last d-th level. For each such node v and every possible arithmetic operation attached to it we construct a new node with this arithmetic operation attached to it, and an activating function which makes the new node "active" if and only if the signs of the algebraic parts of the "active" incoming nodes correspond to the chosen arithmetic operation and besides, v was "active". Obviously, if v was "active" then exactly one node among the newly constructed ones is "active" (otherwise, if v was "passive" then all the constructed nodes are "passive"), hence the number of "active" nodes at the d-th level does not increase. After this transformation the algebraic part of each node at the d-th level becomes a polynomial (rather than a piecewise polynomial function as before the transformation) in the algebraic parts of the incoming nodes from the $(d-1)$-th level (which in their turns still could be piecewise polynomial functions). Reasoning by induction, we can suppose that for some $1 \leq \ell \leq d$ for any $\ell \leq \ell_1 \leq d$ the algebraic part of each node at the ℓ_1-th level is a polynomial in the algebraic parts of the incoming nodes from the $(\ell_1 - 1)$-th level. Now we construct the new nodes for each node at the $(\ell - 1)$-th level as above, thereupon for each node at the ℓ-th level duplicate it the necessary number of times linking the duplicates to the newly constructed nodes at the $(\ell - 1)$-th level, respectively, thereupon duplicate the nodes at the $(\ell + 1)$-th level and so on till the d-th level. This proves the inductive step (for $\ell - 1$).

The size of the transformed MAC (the number of its nodes) could grow considerably, but first we are interested in bounds on the depth, and secondly we could prune a MAC diminishing its size (see corollary 1 below).

Notice also that in case when $p = 1$ we get an extension of the usual (sequential) algebraic computation tree.

The bound on the size s of a MAC could be considerably greater than 2^d as it is the case for the computational models studied in the papers [MP93], [M94] (see the constructions in sections 2,3). In order to bound s we make use of the argument from [MP93], [MMP96] (see also [M94]; for the case of a linear decision trees this argument was earlier introduced in [M88]). Under the computation pattern of a MAC (for a particular input) we understand the list of "active" nodes along the computation at all levels for this input). Since the "active" nodes at any level together with the signs of the polynomials at these "active" nodes determine uniquely the "active" nodes at the next level, we obtain the following lemma based on [MP93], [MMP96], [M94], [M88].

Lemma 1. *The number of computation patterns does not exceed* $\min\{2^{O(d^2 n)}, 2^{pd}\}$.

The idea of the proof of the bound for the first term in the min consists in an observation that the number of possible sets of "active" nodes at the next level $\ell+1$ (see the above discussion) is bounded by $(p2^\ell)^{O(n)}$ due to [M64], herewith 2^ℓ is the obvious upper bound on the degrees of the polynomials computed at ℓ-th level. Then induction on ℓ and the imposed bound $p \le 2^d$ proves the lemma for first term in the min. The proof for the second term is trivial, but sometimes (when p is significantly less than dn) could be better than the bound for the first term.

Corollary 1. *One could prune some nodes from a MAC in such a way that the size of the resulting equivalent MAC does not exceed* $\min\{2^{O(d^2n)}, 2^{pd}\}$.

Proof. Observe that in each particular computation pattern at most pd "active" nodes are involved. For the computation of the algebraic part of each (including "active") node at most 2^d nodes of the MAC are used. Thus, at most $pd2^d$ nodes are involved in each computation pattern, which is dominated by the number of patterns due to lemma 1. It remains to notice that if a node of the MAC was not used in any computation pattern (so, this is a dum node), we could prune it.

Another consequence of lemma 1 which was exploited in [MP93], [M94], is the possibility to apply "connected components counting" technique (well known for the applications to complexity lower bounds for the sequential computation trees [SY82], [B83]) to complexity lower bounds for parallel computation trees. Following [MP93], [M94] we get a complexity lower bound for MAC's.

Corollary 2. *If a MAC accepts a semialgebraic set with N connected components then its depth* $d \ge \Omega\left(\max\left\{\sqrt{\frac{\log N}{n}}, \frac{\log N}{p+n}\right\}\right)$.

The proof is based on the standard counting argument that the semialgebraic set accepted by a particular computation pattern, could contain at most $(pd2^d)^{O(n)}$ connected components because of [M64]; then we apply lemma 1. A similar bound where N denotes the sum of Betti numbers obtained in [MMP96] (its sequential version was obtained in [Y94]), could be also literally extended to MAC's. Thus, the statement of corollary 2 is valid replacing N by the sum of the Betti numbers.

Comparing corollary 2 with the well known sequential lower bound $\Omega(\log N)$ [SY92], [B83]), one sees the gap between $\sqrt{\log N}$ and $\log N$ (ignoring the factor of $\sqrt{n}$). It is an open question, whether one could attain $\sqrt{\log N}$ for the depth of the accepting network [MP93] or an algebraic PRAM [M88], [M94]? In this paper we almost (up to a factor $\sqrt{\log\log N}$) attain this bound for MAC's (see sections 2,3). Moreover, for MAC's which we design below, both $d, p \leq 0(\sqrt{\log N \log\log N})$ (the dimension n is a constant), thus the bounds in corollary 2 are "almost" attained for both terms in the max.

2. Multiprocessor algebraic computation for binary search.

Let the reals $x_1 < \cdots < x_N$. Our purpose is to design a MAC for the binary search problem. Namely, given an input $x \in \mathbb{R}$ to find $1 \leq i \leq N$ such that either $x = x_i$, or $x_{i-1} < x < x_i$ or $x > x_N$ (we agree that $x_0 = -\infty$, $x_{N+1} = \infty$). Actually, a MAC will have a node distinguishing each of these cases, but as we decided to consider just accepting problems, we assume that the problem is to accept a set $V \subset \mathbb{R}$ being a union of several intervals $x_i < x < x_{i+1}$, $0 \leq i \leq N$ and of the points x_i. For simplicity of notation suppose that $N = 2^k - 1$ where $k = \left\lfloor \frac{m^2 \log_2 m}{4} \right\rfloor - 1$ for a certain m.

We'll appeal in the next construction to a (sequential) binary search tree T of the depth $k+1$. We assume that the vertices of the first k levels of T are labeled by the linear polynomials $x - x_i$, $1 \leq i \leq N$ in x (where x is an input of the binary search) and the search branches according to the signs of these linear polynomials. Thus, the root (0 level) is labeled with $x - x_{2^{k-1}}$, its two sons (1-st level) are labeled with $x - x_{2^{k-2}}$ and $x - x_{3 \cdot 2^{k-2}}$, respectively, and so on. Finally, after k levels the vertices of the tree correspond bijectively to the semi-open intervals of the form $x_{i-1} < x \leq x_i$. So, in order to distinguish also the points x_i, at the $(k+1)$-st level we label the corresponding vertex with the linear polynomial $x_i - x$. The linear polynomial attached to the vertex w at the i-th level we denote by $g_i^{(w)}$. Thus, the left subtree of the root corresponds to the (unbounded) interval on the left side from $x_{2^{k-1}}$ (including this point), the right subtree corresponds to the open interval on the right side from x_{2^k-1}. Actually, any subtree $T^{(v)}$ of T (consisting of all the descendants of a vertex v of T, being the root of this subtree), corresponds to a certain interval $I^{(v)}$ (which is either semi-open up to k-th level except the right-most unbounded interval being open, or it is either open or a single point for the $(k+1)$-st level).

For any $1 \leq r \leq m-1$, any vertex v of T at the level $\ell = \sum_{1 \leq j \leq r-1} \lfloor j \log_2 j \rfloor$ and any $0 \leq t \leq \lfloor r \log_2 r \rfloor - 1$, the MAC computes the product $G_t^{(v)}$ of the linear polynomials $g_{\ell+t}^{(w)}$ for all the vertices w of T at the level $\ell+t$ which are descendants (in T) of v. Thus $\deg G_t^{(v)} = 2^t$ and the MAC can compute each $G_t^{(v)}$ independently in t step (multiplications).

Simultaneously for any $1 \leq r \leq m-1$ at its level $L = 3 \sum_{1 \leq j \leq r-1} \lceil \log_2 j \rceil$ the MAC arranges (by recursion on r) 2^ℓ nodes which correspond bijectively to the vertices of T at the level ℓ. We refer to this as a result of $r-1$ rounds of the MAC and describe the r-th round. At the level L just one node ∂ of the MAC is "active". This node ∂ corresponds to a certain vertex v of T at the level ℓ which would be virtually reached if the binary search was applied to x.

At the level $L+1$ the MAC arranges $\lfloor r \log_2 r \rfloor$ nodes $\partial_0, \ldots, \partial_{\lfloor r \log_2 r \rfloor - 1}$ with algebraic parts $G_0^{(v)}, \ldots, G_{\lfloor r \log_2 r \rfloor - 1}^{(v)}$, respectively. These nodes are just the "active" nodes at the level $L+1$, and they are linked with the node ∂ at the level L. At the level $L+2$ the MAC arranges $2\lfloor r \log_2 r \rfloor$ nodes partitioned into pairs. Each pair corresponds to one of the polynomials $G_j^{(v)}$, $0 \leq j \leq \lfloor r \log_2 r \rfloor - 1 - 1$, and both nodes are linked with the node ∂_j at the previous level $L+1$. Each of these nodes from a pair of nodes corresponds to one of two possible signs $\leq, >$ of the polynomial $G_j^{(v)}$ and exactly one of these two nodes which corresponds to the correct sign, is "active" at the level $L+2$. Thereby, the activating boolean functions (see the previous section) for these $2\lfloor r \log_2 r \rfloor$ nodes are described. Observe that exactly $\lfloor r \log_2 r \rfloor$ nodes are "active" at the level $L+2$.

Now the crucial observation enters the game. Consider a subtree $T_1^{(v)}$ of T with the root at v consisting of all its descendants with the levels between ℓ and $\ell + \lfloor r \log_2 r \rfloor - 1$.

Each leaf w of $T_1^{(v)}$ corresponds to a certain subinterval $I^{(w)}$. I claim that the partition of $I^{(v)}$ into $2^{\lfloor r \log_2 r \rfloor}$ intervals of the type $I^{(w)}$ is determined completely by the signs of the polynomials $G_0^{(v)}, \dots, G_{\lfloor r \log_2 r \rfloor -1}^{(v)}$. Actually, the claim holds for any q where $T_1^{(v)}$ is replaced by a subtree $T^{(v,q)}$ of $T_1^{(v)}$ with the root at v consisting of all the descendants of v with the levels between ℓ and $\ell + q$. Then the signs of the polynomials $G_0^{(v)}, \dots, G_q^{(v)}$ determine the partition of the interval $I^{(v)}$ into 2^{q+1} subintervals which correspond to the leaves of the tree $T^{(v,q)}$. The claim can be easily proved by induction on q. The main point in the proof is that for any leaf w of $T^{(v,q)}$ its interval $I^{(w)}$ is divided at the next level just by the polynomial $G_{q+1}^{(w_1)}$ since all the linear polynomials $g_{\ell+q+1}^{(w_1)}$ occurring in the product $G_{q+1}^{(v)}$ have constants signs on $I^{(w)}$ except the polynomial $g_{\ell+q+1}^{(w)}$.

Now following the construction from the example in the previous section, at each level $L+t+2$, $1 \le t \le \lceil \log_2 \lfloor r \log_2 r \rfloor \rceil = R$ we arrange $2^{2^t} \lceil \lfloor r \log_2 r \rfloor / 2^t \rceil$ nodes in MAC which correspond bijectively to all possible signs of the polynomials in the block of polynomials $B = \{G_{q \cdot 2^t + 1}^{(v)}, \dots, G_{(q+1)2^t}^{(v)}\}$ where $0 \le q < \lceil \lfloor r \log_2 r \rfloor / 2^t \rceil$. Each node has two links with the nodes at the level $L + t + 1$ which correspond to the signs of the polynomials in the blocks $B_1 = \{G_{2q2^{t-1}+1}^{(v)}, \dots, G_{(2q+1)2^{t-1}}^{(v)}\}$ and $B_2 = \{G_{(2q+1)2^{t-1}+1}^{(v)}, \dots, G_{(2q+2)2^{t-1}}^{(v)}\}$, respectively. Notice that $B = B_1 \cup B_2$.

The described nodes of the MAC are yielded by induction on t. Remark that in the described nodes only their indicators matter, the algebraic parts of these nodes we could ignore. More precisely, the activating boolean function computing the indicator of any such node is the same for all of them and gives the value "active" if and only if both incoming to it nodes are "active". At the end of this construction, at the level $L + R$ the MAC contains 2^{2^R} nodes which correspond, in particular, to all the possible signs of the polynomials $G_0^{(v)}, \dots, G_{\lfloor r \log_2 r \rfloor}^{(v)}$ and thereby, to all the leaves of the tree $T_q^{(v)}$, as we have proved above in the claim.

Observe that at the level $L + t + 2$, $1 \le t \le R$ the MAC has $\lceil \lfloor r \log_2 r \rfloor / 2^t \rceil$ "active" nodes which correspond to all the compatible signs of the polynomials in each block.

Thus, at the end, at the level $L+R$, there is just one "active" node which corresponds to the vertex of the tree T at the level $\ell + \lceil r \log_2 r \rceil$ which is reached by the binary search being applied to x. This completes the description of the r-th round of the MAC.

Now let us prove that the MAC runs correctly and estimate its depth. We start the next $(r+1)$-st round at the level $L + 3\lceil \log_2 r \rceil$ of the MAC so this additional $3\lceil \log_2 r \rceil$ levels would be enough to realize the described r-th round because $R + 2 < 3\lceil \log_2 r \rceil$. Besides, before the level L the MAC has enough depth (parallel time) to compute all $G_0^{(v)}, \dots, G_{\lfloor r \log_2 r \rfloor}^{(v)}$, taking into account that every $G_q^{(v)}$ is the product of 2^q linear polynomials and $q \le r \log_2 r \le L$. After $m-1$ rounds MAC simulates $\sum_{1 \le j \le m-1} \lfloor j \log_2 j \rfloor \ge \left\lfloor \frac{m^2 \log m}{4} \right\rfloor = k+1$ levels of the tree T, whence the whole binary search tree T. Thus, the depth of MAC is bounded by $d \le O(\sum_{1 \le j \le m-1} \log_2 j) \le O(m \log m) \le O(\sqrt{k \log k}) \le O(\sqrt{\log N \log\log N})$. Notice that the number of processors $p \le O(m \log m) \le O(\sqrt{\log N \log \log N})$, and the size of the designed MAC $s \le O(N)$, which is close to the bound in corollary 1 from section 1.

To accept a set $V \subseteq \mathbb{R}$ (see the beginning of the section) we simply assign the "accept" or "reject" output in the necessary way to each of the nodes corresponding to every particular open interval (x_i, x_{i+1}) or to a point x_i.

Finally, we formulate the main result of this section.

Theorem 1. *There exists a multiprocessor algebraic computation which solves the problem of binary search with depth $d \le O(\sqrt{\log N \log\log N})$ and with a similar bound on the number p of processors.*

If we take a set $V \subseteq \mathbb{R}$ in such a way that it contains N connected components then applying corollary 2 from the previous section we obtain the lower bound $d \geqslant \Omega(\sqrt{\log N})$ on the depth. So, there is still a gap within a factor $O(\sqrt{\log\log N})$ between the upper and lower bounds.

3. Recognizing linear complexes by multiprocessor algebraic computations.

By a linear complex $U \subset \mathbb{R}^n$ we mean any language which could be represented by a quantifier-free formula with linear inequalities as its atomic subformulae. Or in a geometrical language one could think of a given family of hyperplanes $H_1, \dots, H_m \subset \mathbb{R}^n$. Then the number of cells into which $H_1, \dots, H_m$ partition $\mathbb{R}^n$ is at most $m^{O(n)}$ (e.g. see [M64], also [G88]). Denote by $C(H_1, \dots, H_m)$ the set of all these cells. A linear complex could be defined as a union of some of the cells from $C(H_1, \dots, H_m)$.

In this section we design a MAC which recognizes a linear complex, relying on theorem 1 from the previous section. The complexity bound is nontrivial for small dimensions n.

There are quite sophisticated methods for recognizing linear complexes by means of (sequential) linear decision trees with complexity $O(n^{O(1)} \log m)$ (see [M88], [M93]). This bound is sharp (due to [SY82], [B83]) ignoring n (so, for small n relative to m). Unfortunately, it is unclear how to adjust these methods for MAC's. Therefore, we make use of a much more general method of cylindrical algebraic decomposition [C75], which provides a worse dependency on the dimension n. But the main issue will be the improvement of dependency of the depth d on m, we'll get the bound $d \leq O(\sqrt{\log m \log\log m})$ (cf. theorem 1) which is close to the lower bound in corollary 2 from section 1 (for small n).

We design by recursion on n a MAC for $C(H_1, \dots, H_m)$, which outputs a cylindrical algebraic decomposition for $C(H_1, \dots, H_m)$, i.e. a certain partition of $\mathbb{R}^n$ into polyhedra (so, there is a node in the MAC for each of these polyhedra), being finer than the partition $C(H_1, \dots, H_m)$. Denote by $\pi : \mathbb{R}^n \to \mathbb{R}^{n-1}$ a linear projection. We assume for simplicity that $\mathbb{R}^{n-1} = \pi(\mathbb{R}^n)$ is embedded in $\mathbb{R}^n$ and π is the orthogonal projection onto $\mathbb{R}^{n-1}$. The pairwise projections $\pi(H_i \cap H_j)$, $1 \leq i < j \leq m$ provide a family of hyperplanes in $\mathbb{R}^{n-1}$ (choosing π in a suitable way we could suppose that $\dim \pi(H_i \cap H_j) = n-2$, $1 \leq i < j \leq m$). Then for any cell $C \in C(H_1, \dots, H_m)$ its projection $\pi(C)$ is a union of several cells from $C(\{\pi(H_i \cap H_j)\}_{1 \leq i < j \leq m})$ [C75].

To design MAC for $C(H_1, \dots, H_m)$ with an input $x \in \mathbb{R}^n$ first we design (recursively on n) a MAC for $C(\{\pi(H_i \cap H_j)\}_{1 \leq i < j \leq m})$ with an input $\pi(x)$. Let the point $\pi(x)$ reach an element c of the cylindrical algebraic decomposition for $C(\{\pi(H_i \cap H_j)\}_{1\le i<j\le m})$, thus the MAC reaches some node ∂ for the input $\pi(x)$. Then the intersections $h_1 = H_1 \cap \pi^{-1}(c), \dots, h_m = H_m \cap \pi^{-1}(c)$ are pairwise either disjoint or coincide and hence linearly ordered $h_{i_1}\sigma_1 h_{i_2} \cdots \sigma_{m-1} h_{i_m}$ with respect to the coordinate in $\mathbb{R}^n$ orthogonal to $\pi(\mathbb{R}^n)$ where each sign $\sigma_j, 1 \le j \le m-1$ is either $>$ or $=$ (for an appropriate permutation $\{i_1, \dots, i_m\}$ of $\{1, \dots, m\}$). The elements into which $h_i, \dots, h_{im}$ partition the cylinder $\pi^{-1}(c)$ constitute just the elements of the cylindrical algebraic decomposition for $C(H_1, \dots, H_m)$ [C75]. Thereupon, to the node ∂ we paste a MAC which performs a binary search among $h_{i_1}, \dots, h_{i_m}$; so the linear functions (considered as functions in the coordinate orthogonal to $\pi(\mathbb{R}^n)$ determining the hyperplanes $H_{i_1}, \dots, H_{i_m}$, play the role of the linear functions $x - x_1, \dots, x - x_N$ in the construction of the MAC in the previous section. Actually, the latter MAC cares as well of the situation we encounter in the present section, when some pairs of adjacent points could coincide, i.e. $x_{i-1} = x_i$. The resulting MAC has a node for every element of the cylindrical algebraic decomposition, thus at least one node for each cell from $C(H_1, \dots, H_m)$. To complete designing the MAC which accepts a linear complex $V \subset \mathbb{R}^n$, attach "accept" or "reject" output to each of these nodes.

To estimate the complexity of the designed MAC observe that the number of hyperplanes in $\mathbb{R}^{n-1}$ after the projection π is bounded by m^2. Hence after $(n-1)$ projections it could increase as $m^{2^{n-1}}$. Thus, the application of theorem 1 from section 2 gives the bound on the depth and the number of processors of the designed MAC. We summarize in the following theorem the results obtained in the present section.

Theorem 2. *There is a multiprocessor algebraic computation which accepts a linear complex given by m hyperplanes in $\mathbb{R}^n$ with depth and number of processors both $d, p \le O((n\,2^n)^{1/2}\sqrt{\log m \log\log m})$.*

Thus, there is still a gap within the factor $\sqrt{\log\log m}$ with the lower bound provided by corollary 2 from section 1 (for small n), taking into account that the number of connected components in a linear complex could be $m^{\Omega(n)}$. It would be interesting to obtain similar (to theorem 2) bounds for nonlinear cylindrical algebraic decomposition ([C75]).

4. Complexity lower bounds on testing membership to a polyhedron by a MAC and a MAD.

The known topological methods for obtaining complexity lower bounds for decision and computation algebraic trees, based on the number of connected components ([SY82], [B83], [MP93], [M94], see also the discussion in section 1) or more generally, the sum of Betti numbers ([Y94], [MMP96]), cannot be applied to the problem of membership to a polyhedron because of the trivial topological structure of the latter.

Therefore, alternative approaches were developed, which allowed one to obtain complexity lower bounds for testing membership to a polyhedron by algebraic decision and computation trees [GKV95], and for randomized algebraic decision trees [GKMS96]. Notice that the similar question for randomized algebraic computation trees remains open (see the discussion in [GKMS96]). The lower bounds in both papers have the form $\Omega(\log N)$ where N is the number of the faces of all the dimensions of a polyhedron, the similar lower bound in the case of the linear decision trees was ascertained in [YR80].

The purpose of this section is to extend the mentioned results to MAC's (as well as randomized MAC's) and to the weaker computational model of the multiprocessor algebraic decision-maker.

First consider a MAC with the depth d and p processors, which tests membership to a polyhedron $P \subset \mathbb{R}^n$ with N faces. As we have shown in lemma 1 of section 1 there are at most $\min\{2^{O(d^2n)}, 2^{pd}\}$ computation patterns. Each pattern either accepts or rejects a semialgebraic set $W \subset \mathbb{R}^n$. Let a pattern accept W, then $W \subset P$. In [GKV95] we say that W "touches" a face F of P if $\dim(W \cap F) = \dim F$. Theorem 2 [GKV95] implies that the number of faces "touched" by W does not exceed $(n2^d)^{O(n)}$. Hence $N \leq \min\{2^{O(d^2n)}, 2^{pd}\}(n2^d)^{O(n)}$. This entails the inequality $N \leq 2^{O(d^2n)}n^{O(n)}$, therefore, there exists a constant $c_1 > 0$ such that if $N \geq n^{c_1 n}$ then $d \geq \Omega\left(\sqrt{\frac{\log N}{n}}\right)$ (cf. corollary 2 in section 1). Also, the inequality entails that $N \leq n^{O(n)} 2^{O(p+n)d}$ and hence $d \geq \frac{\log N}{p+n}$. Thus, we obtain the proposition (cf. corollary 2 in section 1).

Proposition 1. *Let a MAC with depth d and number p of the processors accept a polyhedron in $\mathbb{R}^n$ with N faces. There exists a constant $c_1 > 0$ such that if $N \geq n^{c_1 n}$ then $d \geq \Omega(\max\{\sqrt{\frac{\log N}{n}},\ \frac{\log N}{p+n}\})$.*

Denote by m the number of faces of P of the highest dimension $n-1$, so the hyperfaces. Comparing proposition 1 with the upper bound in theorem 2 from the previous section, we see that the upper and lower bounds differ by a factor of $\sqrt{\log\log N}$ (for small n), taking into account the obvious inequalities $m \leq N \leq m^n$.

Now consider a randomized MAC which could be defined as a family $\{M_\alpha\}$ where each MAC M_α is chosen with a probability $p_\alpha \geq 0$, $\sum\limits_\alpha p_\alpha = 1$. For any input a randomized MAC should give a correct output with a probability $> \frac{2}{3}$. Applying [GKMS96] we take as inputs a finite set of sample (infinitesimal) points of a special type, and choose M_{α_0} in such a way that M_{α_0} gives the correct outputs for $> \frac{2}{3}$ deal of these inputs. As above the number of computation patterns of M_{α_0} does not exceed $\min\{2^{O(d^2 n)},\ 2^{pd}\}$. For each pattern the number of faces of P which this pattern represents (in [GKMS96] the pattern represents a face by means of a flag, i.e. a sequence of the hyperplanes being the highest dimension faces of P, whose intersection coincides with this face; more precisely, this flag could contain at most $\frac{1}{3}$ deal of gaps in a sequence of hyperplanes due to the probabilistic origination of M_{α_0}), is bounded from above by $m^{O(n)} \cdot \binom{2^d}{n}^{O(1)}$ [GKMS96]. Observe that a constant c hidden in the notation $m^{O(n)}$ (i.e. m^{cn}) depends on the error (in the above setting $\frac{1}{3}$) of the randomized MAC. One could make the error to be a constant arbitrarily close to zero (at the expense of increasing randomized MAC [M85]) and thereby, to make c also to be arbitrarily close to zero.

Thus, $N \leq \min\{2^{O(d^2 n)},\ 2^{pd}\} m^{O(n)} \binom{2^d}{n}^{O(1)}$. Because of the latter remark it suffices to impose a condition $N \geq m^{c_2 n}$ for arbitrary small $c_2 > 0$. This condition implies the lower bound $d \geq \Omega\left(\sqrt{\frac{\log N}{n}}\right) \geq \Omega(\sqrt{\log m})$. Also we get the bound $d \geq \Omega\left(\frac{\log N}{p+n}\right) \geq \Omega\left(\frac{n \log m}{p+n}\right)$. Let us summarize the obtained above in the follow proposition.

Proposition 2. *Let a randomized MAC with depth d and number P of processors accept a polyhedron in $\mathbb{R}^n$ with N faces including m faces of the highest dimension $n-1$. For any constant $c_2 > 0$ if $N \geq m^{c_2 n}$ then $d \geq c_2^{'}\left(\max\left\{\sqrt{\log m}\,, \frac{n \log m}{p+n}\right\}\right)$ for a suitable constant $c_2^{'}$ depending on c_2.*

Observe that proposition 2 implies proposition 1 when $m \leq n^{O(1)}$.

Now we introduce a modification of MAC and call it the multiprocessor δ-algebraic decision-maker (δ-MAD). It relates to MAC in the similar manner how decision trees relate to computation trees. δ-MAD is defined similarly to MAC (see section 1) with the difference that the polynomials in the algebraic parts of the nodes have degrees less of equal to δ. Since we discuss lower bounds we can assume that these polynomials are not actually computed by δ-MAD, but rather are preprocessed in the nodes (it is a usual assumption for the algebraic decision trees). It is reasonable to suppose that $\delta \leq 2^d$. Observe that the device from the example in section 1 could be viewed as 1-MAD. Let δ-MAD accept a polyhedron P. Then the number of computation patterns of δ-MAD does not exceed $\min\{(\delta p)^{O(dn)}, 2^{pd}\}$ (cf. lemma 1 in section 1). The semialgebraic set accepted by any computation pattern cannot "touch" more than $(\delta p d n)^{O(n)}$ faces of P due to theorem 2 [GKV95] (see the above discussion before proposition 1). Therefore, $N \leq \min\{(\delta p)^{O(dn)}, 2^{pd}\}(\delta p d n)^{O(n)}$ and we get the following lower bound.

Proposition 3. *Let a multiprocessor δ-algebraic decision-maker with depth d and number p of the processors accept a polyhedron P with N faces. Then there exists a constant $c_3 > 0$ such that*

a) $d \geqslant \Omega\left(\frac{\log N}{n \log(\delta p)}\right)$, *provided that* $N \geq n^{c_3 n}$;

b) $d \geqslant \Omega\left(\frac{\log N}{p}\right)$, *provided that* $N \geq (\delta p n)^{c_3 n}$.

Notice that plugging $\delta = 2^d$ (which is, we remind, the upper bound on the degrees of the algebraic parts in a MAC), we get proposition 1.

Observe that the lower bounds in proposition 3 remain valid replacing a polyhedron P by an arbitrary semialgebraic set and replacing N by the number of connected components even in a stronger setting in which we get rid of the supposition $N \geq n^{c_3 n}$ in both a) and b) (one could treat this statement as an analogue of corollary 2 in section 1 for the multiprocessor algebraic decision-makers).

Finally, we proceed to considering randomized δ-MAD's, $\{D_\alpha\}$. As above applying again [GKMS96] we choose D_{α_0} and bound the number of its computation patterns by $\min\{(\delta p)^{O(dn)}, 2^{pd}\}$. The number of the faces of P represented by a computation pattern (by means of the flags, see above) does not exceed $m^{O(n)} \cdot \binom{\delta p d}{n}^{O(1)}$ [GKMS96]. Arguing as in the proof of proposition 2 we get the following lower bound (taking into account the inequality $\binom{a}{b} \leq \left(\frac{3a}{b}\right)^b$).

Proposition 4. *Let a randomized multiprocessor δ-algebraic decision-maker with depth d and number p of the processors accept a polyhedron P with N faces, including m faces of the highest dimension $n-1$. For any constant $c_4 > 0$ if $N \geq m^{c_4 n}$ then*

a) $d \geqslant c_4^{'} \left(\frac{\log m}{\log(\delta p)}\right)$;

b) $d \geqslant c_4^{'} \left(\min\left\{\frac{n \log m}{p}, N^{1/n}\right\}\right)$, *provided that* $N \geq \left(\frac{3\delta p}{n}\right)^n$,

for an appropriate constant $c_4^{'}$ depending on c_4.

Notice (like after proposition 3) that plugging $\delta = 2^d$ leads to proposition 2.

Remark. 1) Propositions 1–4 remain true replacing a polyhedron by an arrangement of m hyperplanes with N faces. Then propositions 1,3 follow in fact from corollary 2 (section 1), the proofs of propositions 2,4 for arrangements are the same as above for polyhedra (see [GKMS96]).

2) Proposition 1 holds a fortiori for the weaker (than MAC) computational models of the algebraic PRAM and accepting network ([M88], [MP93], [M94], see also the discussion

in section 1). Proposition 2 holds for the randomized version of these computational models (cf. [M94]). By the same token proposition 3 (respectively, 4) holds for the parallel algebraic decision trees (respectively, randomized).

Acknowledgements. The author is thankful to Felipe Cucker and Vitya Pan, who encouraged him to look at parallel computations.

References

[B83] M. Ben-Or, *Lower bounds for algebraic computation trees*, Proc. ACM Symp. on Th. Comput., 1983, p. 80–86.

[C75] G.Collins, *Quantifier elimination for real closed fields by cylindrical algebraic decomposition*, Lect. Notes Comput. Sci. v. 33, 1975, p. 134–183.

[CG96] F. Cucker, D. Grigoriev, *On the power of real Turing machines over binary inputs*, SIAM J. Comput., v. 26, 1, 1997, p. 243–254.

[G86] J. von zur Gathen, *Parallel arithmetic computations: a survey*, Lect. Notes Comput. Sci., v. 233, 1986. p. 93–112.

[G88] D. Grigoriev, *Complexity of deciding Tarski algebra*, J. Symb. Comput., v. 5, 1988, p. 65–108.

[GKV95] D. Grigoriev, M. Karpinski, N. Vorobjov, *Improved lower bound on testing membership to a polyhedron by algebraic decision trees*, Proc. IEEE FOCS, 1995, p. 258–265.

[GKMS96] D. Grigoriev, M. Karpinski, F. Meyer auf der Heide, R. Smolensky, *A lower bound for randomized algebraic decision trees*, Proc. ACM STOC, 1996, p. 612–619.

[M93] S. Meiser, *Point location in arrangements of hyperplanes*, Information and Computation, v. 106, 1993, p. 286–303.

[M85] F. Meyer auf der Heide, *Simulating Probabilistic by Deterministic Algebraic Computation Trees*, Theor. Comput. Sci., 1985, 41, p. 325-330.

[M88] F. Meyer auf der Heide, *Fast algorithms for N-dimensional restrictions of hard problems*, J. ACM, v. 35, 1988. p. 740–747.

[M64] J. Milnor, *On the Betti numbers of real varieties*, Proc. AMS , v. 15, 1964, p. 275–280.

[MP93] J. Montana, L. Pardo, *Lower bounds for arithmetic networks*, Appl. Algebra in Eng., Communic. and Comput., v. 4, 1993, p. 1–24.

[MMP96] J. Montana, J. Morais, L. Pardo, *Lower bounds for arithmetic network II: sum of Betti numbers*, Appl. Algebra in Eng., Communic. and Comput., v. 7, 1996, p. 41–51.

[M94] K. Mulmuley, *Lower bounds for parallel linear programming and other problems*, Proc. ACM Symp. on Th. Comput., 1994, p. 603–614.

[SY82] M. Steele, A. Yao, *Lower bounds for algebraic decision trees*, J. Algorithms, v. 3, 1982, p. 1–8.

[Y94] A. Yao, *Decision tree complexity and Betti numbers*, Proc. ACM Symp. Th. on Comput., 1994, p. 615–624.

[YR80] A. Yao, R. Rivest, *On the polyhedral decision problem*, SIAM J. Comput., v. 9, 1980, p. 343–347.