Lower Bounds For Algebraic Computation Trees

(Preliminary Report)

Michael Ben-Or [†]

Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Abstract — A topological method is given for obtaining lower bounds for the height of algebraic computation trees, and algebraic decision trees. Using this method we are able to generalize, and present in a uniform and easy way, almost all the known nonlinear lower bounds for algebraic computations. Applying the method to decision trees we extend all the apparently known lower bounds for linear decision trees to bounded degree algebraic decision trees, thus answering the open questions raised by Steele and Yao [20]. We also show how this new method can be used to establish lower bounds on the complexity of constructions with ruler and compass in plane Euclidean geometry.

1. Introduction

Despite the extensive research in algebraic complexity theory in recent years, no general lower bound method has been provided for algorithms that involve arithmetical operations and comparisons. Much less is known if we further allow the operation of root extraction or the algebraic operation of finding the root of a polynomial. Consider the following decision problem:

Example 1. Element Distinctness. *Given $x_1, \ldots, x_n \in \mathbf{R}$, is there a pair i, j with $i \neq j$ and $x_i = x_j$?*

One can solve the element distinctness problem with the help of any efficient sorting algorithm using $O(n \log n)$ comparisons, or by computing the product $\prod_{i \neq j}(x_i - x_j)$ and comparing the computed result to zero (using $O(n \log n)$ mult/div). Allowing linear operations for free, we know of no previous result that indicates why more than $O(1)$ operations are required to solve this problem in the model considered here.

In this paper we provide a new topological method for obtaining lower bounds for this general type of algorithms, formally described as *algebraic computation trees.* Before giving the detailed computational model it is worthwhile to mention a concrete application of the method presented here.

Theorem 1. *Any algebraic computation tree that solves the n-element distinctness problem must have complexity of at least $\Omega(n \log n)$.*

This result extends the lower bounds of Dobkin and Lipton [5] for the linear decision tree model, and the lower bounds of Baur and Strassen [1] for the straight line complexity of the above product.

Our new lower bound method rests heavily on a result from real algebraic geometry due to Milnor [11] and Thom [23] that bounds the "topological complexity" of real algebraic varieties. Except for this result the proofs of our main theorems are elementary and require only basic knowledge of algebra and topology. The new method also provides a unified and easy way to prove nonlinear lower bounds for straight line computations, algebraic decision trees, and other previously untouchable problems such as lower bounds for the complexity of constructions with a ruler and compass in plane Euclidean geometry.

In the next section we rigorously specify our basic computational model. The third section is devoted to a technical result needed for our main theorems that are presented in section four. In section four we also show how to extend our computational model to allow more algebraic operations such as taking k-th roots or computing the roots of a polynomial.

In section five we show how to apply our method to the bounded degree algebraic decision tree model, thus solving the open problems in [20]. Section six is devoted to applications and in particular to the proof of the result on the element distinctness problem (Theorem 1) mentioned above.

[†] Research supported by a Weizmann Postdoctoral fellowship and by NSF grant MCS-8006938.

© 1983 ACM 0-89791-099-0/83/004/0080 $00.75

In the seventh section we show how to use our lower bound method to prove lower bounds on the number of *ruler and compass* operations for certain plane geometry constructions. This solves the open problem posed by Shamos [18].

2. The Algebraic Computation Tree Model

Let $W \subseteq \mathbf{R}^n$ be any set. The *membership problem* for W is the following:

$$\textit{Given} \quad x=(x_1,\dots,x_n)\in\mathbf{R}^n \quad \textit{determine} \quad \textit{if} \quad x\in W.$$

In our example 1 we can set,

$$W=\left\{(x_1,\dots,x_n)\,\middle|\,\prod_{i\neq j}(x_i-x_j)\neq 0\right\}\subseteq \mathbf{R}^n.$$

We are interested in obtaining lower bounds on algorithms for solving the membership problem for W that allow both arithmetic operations and tests. Thus, a step of the computation in our model will be either an *Arithmetic Operation* or a *Test* — comparing a computed result to zero (i.e. $>,\geq,=$), and branching according to the outcome of the test.

Formally, an *algebraic computation tree* is a binary tree T with a function that assigns:

- to any vertex v with exactly one son (simple vertex) an operational instruction of the form

  $$f_v := f_{v_1}\circ f_{v_2} \quad \textit{or} \quad f_v := c\circ f_{v_1} \quad \textit{or} \quad f_v := \sqrt{f_{v_1}}$$

  where v_i is an ancestor of v in the tree T, or $f_{v_i}\in\{x_1,\dots,x_n\}$, $\circ\in\{+,-,\times,/\}$, and $c\in\mathbf{R}$ is a constant.
- to any vertex v with two sons (branching vertex) a test instruction of the form

  $$f_{v_1}>0 \quad \textit{or} \quad f_{v_1}\geq 0 \quad \textit{or} \quad f_{v_1}=0$$

  where v_1 is an ancestor of v, or $f_{v_i}\in\{x_1,\dots,x_n\}$.
- to any leaf an output YES or NO.

Given an input $x\in\mathbf{R}^n$, the program traverses a path $P(x)$ in the tree T down from the root. At each simple vertex the arithmetical operation is performed, and at each branching vertex a branching is made according to the test at the vertex. When a leaf is reached the answer YES or NO is returned. We say that "x passes through a vertex v" if v is on the path $P(x)$. We require that if an input x passes through a vertex v with a division instruction $f_v := f_{v_1}/f_{v_2}$ that $f_{v_2}(x)\neq 0$, and if $f_v := \sqrt{f_{v_1}}$ that $f_{v_1}(x)\geq 0$.

We say that the computation tree T solves the membership problem for W if the answer returned is correct for every input $x\in\mathbf{R}^n$. Let $\mathit{cost}(x,T)$ denote the number of vertices that x passes through. The complexity of T, $C(T)$, is given by the maximum of $\mathit{cost}(x,T)$ for any x.

Now let $C(W)$ be the minimum $C(T)$ for any algebraic computation tree T that solves the membership problem for W. The lower bound on $C(W)$ derived by our method will depend heavily on the topology of W. For this purpose we derive in the next section upper bounds on the topological complexity of sets defined by polynomial equalities and inequalities.

3. Counting Connected Components

Let $V\subseteq\mathbf{R}^n$ be a set defined by the following polynomial equations

$$\begin{array}{r}
q_1(x_1,\dots,x_n)=0,\dots,q_m(x_1,\dots,x_n)=0,\\
p_1(x_1,\dots,x_n)>0,\dots,p_s(x_1,\dots,x_n)>0,\\
p_{s+1}(x_1,\dots,x_n)\geq 0,\dots,p_h(x_1,\dots,x_n)\geq 0.
\end{array}\qquad (I)$$

where $q_i,p_j\in\mathbf{R}[x_1,\dots,x_n]$, and $d=\max\{2,\mathit{deg}\,q_i,\mathit{deg}\,p_j\}$.

Denote by $\#V$ the number of connected components V has. For any integers n,h,d we put

$$\beta_d(n,h)=\max\{\#V\mid V\subseteq\mathbf{R}^n \text{ is defined by } (I)\}.$$

Note that we do not bound the number of polynomial equalities defining V in the definition of $\beta_d(n,h)$, but merely bound their degree and the number of inequalities.

Proving upper bounds on $\beta_d(n,h)$ is apparently not an easy matter. Fortunately, we can easily reduce the problem to a similar problem about algebraic varieties (defined by polynomial equalities) for which we can apply the known results of Milnor [11] and Thom [23].

Theorem 2. *For $d\geq 2$,*

$$\beta_d(n,h)\leq d(2d-1)^{n+h-1}$$

Proof. Let $V\subseteq\mathbf{R}^n$ be a set defined by (I). $\#V$ is certainly finite (see [12]), so pick a base point in every connected component of V. Let $v_1,\dots,v_r\in V$ be these points, where $r=\#V$, and set

$$\epsilon=\min\{p_i(v_j)\mid i=1\dots s,\quad j=1\dots r\}.$$

Since all the v_j are in V we know that $\epsilon>0$. Let

$$V_\epsilon=\left\{\mathbf{x}\in\mathbf{R}^n\,\middle|\,\begin{array}{r}q_1(\mathbf{x})=0,\dots,q_m(\mathbf{x})=0,\\ p_1(\mathbf{x})\geq\epsilon,\dots,p_s(\mathbf{x})\geq\epsilon,\\ p_{s+1}(\mathbf{x})\geq 0,\dots,q_h(\mathbf{x})\geq 0.\end{array}\right\}$$

then $V_\epsilon\subseteq V$, and $v_j\in V_\epsilon$ for all j. All the v_j must be in distinct connected components of V_ϵ, because $V_\epsilon\subseteq V$, thus $\#V\leq\#V_\epsilon$. Now let W be the set of all solutions $(x_1,\dots,x_n,y_1,\dots,y_h)\in\mathbf{R}^{n+h}$ to the system

$$\begin{array}{r}
q_1(x_1,\dots,x_n)=0,\dots\dots\dots,q_m(x_1,\dots,x_n)=0\\
p_1(x_1,\dots,x_n)=y_1^2+\epsilon,\dots,p_s(x_1,\dots,x_n)=y_s^2+\epsilon\\
p_{s+1}(x_1,\dots,x_n)=y_{s+1}^2,\dots,p_h(x_1,\dots,x_n)=y_h^2
\end{array}$$

and let $\pi : \mathbf{R}^{n+h} \longrightarrow \mathbf{R}^n$ be the projection $\pi(x_1, \ldots, x_n, y_1, \ldots, y_h) = (x_1, \ldots, x_n)$. The function $\pi|_W$ is continuous and $\pi(W) = V_\epsilon$, so clearly $\#V_\epsilon \leq \#W$.

Since $W \subseteq \mathbf{R}^{n+h}$ is an algebraic variety defined by polynomials of degree $\leq d$, we can apply a theorem of Milnor [11, Theorem 2] to bound its number of connected components. Thus

$$\#V \leq \#V_\epsilon \leq \#W \leq d(2d-1)^{n+h-1}. \qquad \blacksquare$$

Remark: Milnor [11, Theorem 3] proves that $\beta_d(n, h) = O((dh)^n)$ which is asymptotically better when h tends to infinity but this bound is not useful here.

4. The Main Theorem

Now we are prepared to prove our main theorems that establish the connection between the complexity of the membership problem for W and its topological complexity.

Theorem 3. *Let $W \subseteq \mathbf{R}^n$ be any set, and let T be a computation tree that solves the membership problem for W. If N is the number of disjoint connected components of W, and $h = C(T)$, then*

$$2^h 3^{n+h} \geq N.$$

Proof. Let $\pi = (v_1, \ldots, v_t)$, $t \leq h$, be a path from the root $r = v_1$ of T to a leaf $l = v_t$ with the answer YES, and let V be the set of inputs $\mathbf{x} \in \mathbf{R}^n$ leading to l. We now use the elegant method of reducing the constraints degree by adding new variables (see [14]). Traversing the tree down from the root to l, we set a system of equations Γ according to the operations (or tests) on the vertices of the path π, by the following rules:

Operation	Equation
$f_{v_i} := f_{v_j} \pm f_{v_k}$	$f_{v_i} = f_{v_j} \pm f_{v_k}$
$f_{v_i} := f_{v_j} \times f_{v_k}$	$f_{v_i} = f_{v_j} f_{v_k}$
$f_{v_i} := f_{v_j} / f_{v_k}$	$f_{v_k} f_{v_i} = f_{v_j}$
$f_{v_i} := \sqrt{f_{v_j}}$	$f_{v_i}^2 = f_{v_j}$

and if v_i is a branching vertex with a test

$$f_{v_j} > 0 \quad or \quad f_{v_j} \geq 0 \quad or \quad f_{v_j} = 0$$

then add this equation to Γ if it should be satisfied, and add the negated equation

$$-f_{v_j} \geq 0 \quad or \quad -f_{v_j} > 0 \quad or \quad f_{v_i} f_{v_j} - 1 = 0$$

accordingly otherwise.

Let $f_{u_1}, \ldots, f_{u_r}$ be the set of new variables in Γ, and let s be the number of inequalities in Γ. Then $r + s \leq t$, since each step adds at most one new variable or one inequality. Let U be the set of solutions $(x_1, \ldots, x_n, f_{u_1}, \ldots, f_{u_r}) \in \mathbf{R}^{n+r}$ to the system Γ. It is easily seen that the projection of U on the x coordinate is exactly V so by the same argument given in the proof of theorem 2, we have $\#V \leq \#U$. Since the degree of Γ is ≤ 2, we know by theorem 2 that

$$\#V \leq \#U \leq \beta_2(n + r, s) \leq 2 \cdot 3^{n+r+s-1} \leq 3^{n+h}$$

Since each leaf of T is correctly labeled, each connected component of V must be completely contained in some connected component of W. Since the number of leaves of $T \leq 2^h$, and each leaf has at most 3^{n+h} connected components, we have $2^h 3^{n+h} \geq N$. $\blacksquare$

From theorem 3 we immediately have

Theorem 4. *For any $W \subseteq \mathbf{R}^n$,*

$$C(W) \geq \frac{\log N}{1 + \log 3} - \frac{\log 3}{1 + \log 3} n \approx 0.38 \log N - 0.61 n$$

where $N = \max\{\#W, \#(\mathbf{R}^n - W)\}$.

Examining the proof of theorem 3, one can see that all we need for the proof to work is that the degree of Γ will not be higher than 2. Thus we can allow linear operations for free and count only $\times, /, \sqrt{\ }$ operations and comparisons. Moreover, we can allow any bilinear operation on precomputed results and count it as one operation since this still gives an equation of degree 2.

Using the same type of argument, we can allow the operation of taking k-th roots and handle it in the following way: Let

$$s_0 := y, s_1 = s_0^2, \ldots, s_t := s_i \cdot s_j$$

be a straight line computation of y^k of minimal length. On encountering the operation

$$f_{v_i} := (f_{v_j})^{\frac{1}{k}}$$

we add to Γ the set of equations (of degree ≤ 2)

$$s_0 = f_{v_i}, s_1 = s_0^2, \ldots, f_{v_j} = s_i \cdot s_j$$

which introduces t new variables. Thus the cost of this operation should be t. More generally we can allow the algebraic operation of taking roots of a polynomial (of any degree) at the cost of evaluating this polynomial at a given point.

Formally we associate with each operation a *cost.* Thus addition, subtraction, and multiplication by constants have cost 0, multiplication, division, taking square roots, any bilinear operation and comparisons all have cost 1, taking k-th roots costs $O(\log k)$, and solving a polynomial has the cost of the complexity of evaluating it at a given point.

Let T be a computation tree (with or without the new operations), and let $M(\mathbf{x}, T)$ denote the sum of the costs of the operations along the path $P(\mathbf{x})$. The multiplicative complexity of T, $M(T)$, is the maximum of $M(\mathbf{x}, T)$ for any $\mathbf{x} \in \mathbf{R}^n$, and the (multiplicative) complexity of W, $M(W)$, is the minimum $M(T)$ for any algebraic computation tree that solves the membership problem for W.

Theorem 5. *For any $W \subseteq \mathbf{R}^n$,*

$$M(W) = \Omega(\log N - n)$$

where $N = \max\{\#W, \#(\mathbf{R}^n - W)\}$.

Since our method applies to computations with real numbers we can easily extend our lower bound technique to computations with complex numbers by representing the complex number z as $x + iy$, with $x, y \in \mathbf{R}$. This way, multiplication is represented by two bilinear operations. Let $Re(z)$ and $Im(z)$ denote the real and imaginary parts of the complex number z.

Theorem 6. *Let $W \subseteq \mathbf{C}^n$ be any set, and let T be an algebraic computation tree that solves the membership problem for W, using the functions Re,Im with cost 0 and comparisons on real numbers with cost 1, then*

$$M(T) = \Omega(\log N - n)$$

where $N = \max\{\#W, \#(\mathbf{C}^n - W)\}$.

By continuity arguments we can extend our method to deal also with the rational numbers.

Theorem 7. *Let $W \subseteq \mathbf{Q}^n$ be any set, and let T be a computation tree that solves the membership problem for W. Then*

$$M(T) = O(\log N - n)$$

where N is the number of connected components of $\overline{W}$ in $\mathbf{R}^n$ with non null interior.

5. The Decision Tree Model

Another model of computation used to prove worst-case lower bounds is the decision tree model. Although this model is less interesting from the computational point of view, many worst-case lower bounds have been proved for this model. In this model algorithms are presented as trees, in which every vertex of the tree has the form of a comparison

$$f(inputs) : 0$$

where f is some function from a class of allowed functions.

For *linear* decision trees several powerful techniques are known (e.g. Reingold [15], Dobkin [3], Dobkin and Lipton [4,5], Yao [24], and Yao and Rivest [26]). Less is known on *algebraic* decision trees, and the results of Rabin [13], Steele and Yao [20], and Jaromczyk [8,9], fail to give non-linear lower bounds for some of the problems discussed here. Following Steele and Yao [20], we define:

A *d-th order decision tree* T for testing if $\mathbf{x} \in W \subseteq \mathbf{R}^n$, is a decision tree where the functions allowed are polynomials of degree at most d, each leaf of T contains the answer YES or NO, and for any $\mathbf{x} \in \mathbf{R}^n$, T decides correctly if $\mathbf{x} \in W$. Denote by $C_d(W)$ the minimum height for any d-th order decision tree for the set W.

Theorem 8. *Let $W \subseteq \mathbf{R}^n$ be any set, and let T be a d-th order algebraic decision tree that solves the membership problem for W. If N is the number of disjoint connected components of W, and h is the height of T, then*

$$2^h \beta_d(n, h) \geq N.$$

Thus for fixed d, $\quad C_d(W) = \Omega(\log N - n)$.

6. Applications

As a first example to the strength of our method let us return to our example 1, the element distinctness problem. It is easy to see that for the W defined in section 2, $\#W = n!$, since each region

$$\{(x_1, \ldots, x_n) \mid x_{\sigma(1)} < x_{\sigma(2)} < \ldots < x_{\sigma(n)}\}$$

is a maximal connected component of W for each permutation σ. Thus by theorems 4, 5 and 8 we have that $C(W)$, $M(W)$, and $C_d(W)$ are all at least $\Omega(n \log n)$.

Example 2. Set Equality and Inclusion. *Given two sets $A = \{x_1, \ldots, x_n\}$, $B = \{y_1, \ldots, y_n\}$, determine whether or not (a) $A = B$, or (b) $A \subseteq B$.*

Any computation tree that solves any of these problems will correctly decide the case when $B = \{1, 2, \ldots, n\}$. Now for problem (a) set

$$W_a = \{(\sigma(1), \sigma(2), \ldots, \sigma(n)) \mid \sigma \in S_n\}.$$

Since W contains $n!$ distinct points $\#W = n!$. Thus $C(W_a)$,$M(W_a)$,$C_d(W_a) = \Omega(n \log n)$.

For problem (b) set

$$W_b = \{(x_1, \ldots, x_n) \mid \{x_1, \ldots, x_n\} \subseteq \{1, 2, \ldots, n\}\}.$$

$\#W = n^n$ so again $C(W_b), M(W_b), C_d(W_b) = \Omega(n \log n)$.

Example 3. Set Disjointness. *Given two sets $A = \{x_1, \ldots, x_n\}$ and $B = \{y_1, \ldots, y_n\}$, determine whether or not $A \bigcap B = \emptyset$.*

For this problem set

$$W = \left\{ (x_1, \ldots, x_n, y_1, \ldots, y_n) \middle| \prod_{i,j} (x_i - y_j) \neq 0 \right\}$$

It is easy to see that $\#W \geq (n!)^2$, so again we know that $C(W), M(W), C_d(W) = \Omega(n \log n)$.

These lower bounds extend the lower bounds under the linear decision model due to Reingold [15].

Example 4. The Measure problem. *Given a list of* $2n$ *real (or rational) numbers* $a_1, \ldots, a_n$ *and* $b_1, \ldots, b_n$, *compute the measure of* $\bigcup_i [a_i, b_i]$.

Any algorithm that solves the measure problem can be used to decide whether

$$\bigcup_i [x_i, x_i + \epsilon] = n\epsilon$$

for any $x_1, \ldots, x_n \in \mathbf{R}$ and $\epsilon > 0$. For this problem set

$$W = \{(x_1, \ldots, x_n) \mid |x_i - x_j| \geq \epsilon \quad \text{for all} \quad i \neq j\}.$$

Again $\#W = n!$, thus $C(W), M(W), C_d(W) = \Omega(n \log n)$. This extends the lower bound under the linear decision tree model due to Fredman and Weide [6].

Example 5. Extreme Points. *Given n points in the plane does the convex hull formed by them possess n vertices?*

It has been shown by Steele and Yao [20], that this set $W \subseteq \mathbf{R}^{2n}$ satisfies $\#W \geq (n-1)!$, thus $C(W)$, $M(W)$, $C_d(W) = \Omega(n \log n)$. This generalizes the result of Yao [25] where he showed that $C_2(W) = \Omega(n \log n)$.

Example 6(a). The Knapsack Problem. *Given real numbers* $x_1, \ldots, x_n$, *decide if there exists some subset* $S \subseteq \{1, 2, \ldots, n\}$ *such that* $\sum_{i \in S} x_i = 1$.

In this case,

$$W = \left\{ (x_1, \ldots, x_n) \middle| \prod_S (\sum_{i \in S} x_i - 1) \neq 0 \right\}$$

It was shown in Dobkin and Lipton [4] that $\#W \geq 2^{n^2/2}$, thus $C(W), M(W), C_d(W) = \Omega(n^2)$.

Example 6(b). The ϵ-Approximation Knapsack Problem. *Given real (or rational) numbers* $x_1, \ldots, x_n$ *and* $\epsilon > 0$, *decide if there exists some subset* $S \subseteq \{1, 2, \ldots, n\}$ *such that*

$$|\sum_{i \in S} x_i - 1| < \epsilon$$

For any $\epsilon > 0$, define

$$W_\epsilon = \left\{ (x_1, \ldots, x_n) \middle| |\sum_{i \in S} x_i - 1| > \epsilon \quad \text{for all} \quad S \right\}$$

If ϵ is small enough $\#W_\epsilon = \#W_0 \geq 2^{n^2/2}$, thus any algorithm for our problem must have complexity $\Omega(n^2)$.

Examples 6 (a) and (b) extend the results under the bounded degree algebraic decision tree model due to Steele and Yao [20].

Example 7. Sign of an Ordering Permutation. *Given* $x_1, \ldots, x_n \in \mathbf{R}$, *is there a permutation of odd parity that orders the* x_i*?*

The set defined by this problem is connected but we can derive lower bounds by looking at its complement. Define

$$W = \{(x_1, \ldots, x_n) \mid x_{\sigma(1)} < \cdots < x_{\sigma(n)} \text{ for some } \sigma \in A_n\}$$

Once again $\#W = n!/2$ so any algorithm for this problem has complexity $\Omega(n \log n)$.

Example 8. Symmetric Functions. *Given* $x_1, \ldots, x_n \in \mathbf{R}$, *Compute the elementary symmetric functions*

$$\sigma_1(x_1, \ldots, x_n), \ldots, \sigma_n(x_1, \ldots, x_n).$$

Let $a_i = \sigma_i(1, 2, \ldots, n)$. Any algorithm that computes the elementary symmetric function can be used to test, using n more steps, whether $\sigma_i(x_1, \ldots, x_n) = a_i$ for all i. Since this is true if and only if $\{x_1, \ldots, x_n\} = \{1, \ldots, n\}$, we know from example 2 that the algorithm requires $\Omega(n \log n)$ steps. This extends the result due to Strassen [21], since checking whether $\sigma_i(\mathbf{x}) = a_i$ may actually be easier than computing the values of $\sigma_i(\mathbf{x})$.

Example 9. Discriminant. *Given* $x_1, \ldots, x_n \in \mathbf{R}$, *compute the discriminant* $\prod_{i \neq j}(x_i - x_j)$.

Any algorithm for this problem can, in one more step, test if the discriminant $\neq 0$, and this happens if and only if all the x_i are distinct. So by example 1 the algorithm must make $\Omega(n \log n)$ steps. This extends the result due to Baur and Strassen [1].

Example 10. Resultant. *Given* $x_1, \ldots, x_n, y_1, \ldots, y_n \in \mathbf{R}$, *compute the resultant* $\prod_{i,j}(x_i - y_j)$.

Any algorithm for this problem can, in one more step, test if the resultant $\neq 0$, and this happens if and only if the sets $\{x_i\}$ and $\{y_i\}$ are disjoint. So by example 3 the algorithm must make $\Omega(n \log n)$ steps. This extends another result due to Baur and Strassen [1].

Example 11. Interpolation polynomial. *Given* $(x_1, y_1), \ldots, (x_n, y_n) \in R^2$, *compute the unique interpolating polynomial through those points.*

One can prove an $\Omega(n \log n)$ lower bound for this problem by reducing the problem to the symmetric function computation, because the coefficients of the interpolation polynomial through the points

$$(x_1, 0), (x_2, 0), \ldots, (x_n, 0), (0, \pm x_1 x_2 \cdots x_n)$$

are the elementary symmetric functions of $x_1, \ldots, x_n$.

To show how to prove this directly by our method we note that even if we restrict the input to satisfy $x_1 < \cdots < x_n$, a straight line algorithm that correctly computes the coefficients of the interpolating polynomial for this type of input must give the correct answers even when the x_i are any complex numbers because of analytic continuation. In particular the algorithm gives the correct answers when the x_i are not ordered. Let $p(t) = a_{n-1}t^{n-1} + \cdots + a_0$ be the interpolation polynomial through the points $(1,2),(2,-2),\ldots,(n,\pm 2)$ and let

$$W = \left\{(x_1,y_1,\ldots,x_n,y_n) \middle| \begin{array}{c} y_i^2 = 1 \quad \text{and the interpolation} \\ \text{polynomial} = p(t) \end{array}\right\}$$

Since $p(t) = \pm 1$ has $n-1$ distinct real roots it is easy to see that W contains $(2n-2)!/(n-2)!$ distinct points. Since any algorithm that computes the interpolation polynomial can be used, with n more steps, to recognize W it must require $\Omega(n \log n)$ arithmetical operations. This extends another result due to Strassen [21].

Example 12. Sum of Powers. *Given $z_1,\ldots,z_n \in \mathbf{C}$, compute the sum $z_1^k + \cdots + z_n^k$.*

let $z_i = x_i + y_i$, and set

$$W = \left\{(z_1,\ldots,z_n) \in \mathbf{C}^n \middle| \begin{array}{c} z_1^k + \cdots + z_n^k = n \\ \text{and} \quad x_i^2 + y_i^2 = 1 \end{array}\right\}$$

An algorithm to compute our function can be used with $O(n)$ more steps to solve the membership problem for W. It is easy to see that $(z_1,\ldots,z_n) \in W$ if and only if all the z_i are k-th roots of unity, so W contains k^n discrete points. Since $\#W = k^n$ the complexity of the algorithms is $\Omega(n \log k)$. This extends another result due to Baur and Strassen [1], and the results of Schnorr [17].

Example 13. Integer parts. *Given $x_1,\ldots,x_n \in [0,M]$, compute the sum $[x_1] + \cdots + [x_n]$, where $[x]$ is the integer part of x.*

Let

$$W = \left\{(x_1,\ldots,x_n) \middle| \begin{array}{c} [x_1] + \cdots + [x_n] = x_1 + \cdots + x_n \\ \text{and} \quad 0 \le x_i \le M \end{array}\right\}$$

Any algorithm for our problem can solve the membership problem for W using $O(n)$ more steps. $\#W = (M+1)^n$, so the algorithm has complexity $\Omega(n \log M)$. This extends the result of Schmitt [17] where he showed that computing the integer part of x requires $\Omega(\log M)$ operations.

Remark: All the above lower bounds are tight (up to constant factors) except for examples 6(a) and (b) where the best upper bound is $O(n^4 \log n)$, a recent result due to Meyer auf der Heide (these proceedings).

7. Constructions in Euclidean geometry

The questions of constructibility by the Euclidean ruler and compass (such as trisecting an angle) were raised in ancient times by the Greek mathematicians. With the advent of Galois theory in the early nineteenth century a complete characterization of those problems solvable with ruler and compass became available. Hilbert, in his *Foundations of Geometry* [7], explains how to reduce the constructibility problem to an algebraic problem. By introducing a coordinate system in the plane, he shows how the elementary geometric operations correspond with the operations of addition, subtraction, multiplication, division, and square root extraction.

While elegant and simple constructions were always regarded as desirable, the first systematic study of the complexity of Euclidean constructions was undertaken only early in this century by Lemoine [10]. His work is the only known attempt to count operations in geometry, but he was unable to prove any lower bounds.

More recently Shamos in his work on computational geometry [18] studied a number of fundamental problems in this area, and was able to give upper and lower bounds for problems involving set of points, lines, and polygons in the plane. The lower bounds in Shamos's work were all under the linear decision tree model and were proved by reduction to some of the problems we gave above. Since our algebraic computation tree model can handle the operation of taking square roots most of the lower bounds from Shamos's work can be extended to lower bounds on the complexity of solving the problems with the aid of a ruler and compass.

We allow the following elementary operations:

1. Drawing a line through two points.
2. Drawing a circle (with or without a given radius).
3. Intersecting a circle/line with a circle/line.
4. Determine whether a point is on the right/left side or on a directed line.
5. Determine whether a point is in/out or on a circle.

Thus for example we can prove

Theorem 9. *Any algorithm using the above elementary operation that solves the Extreme Points problem (example 5) has worst-case complexity of at least $\Omega(n \log n)$ operations.*

Theorem 10. *Any algorithm that determines for any $n+1$ given points, $P_0,\ldots,P_n$, whether P_0 is colinear with any two other points has worst-case complexity of $\Omega(n \log n)$ elementary operations.*

Other lower bounds for the problems mentioned in [19] can be proved as well.

8. Remarks

1. Since our lower bound theorems are based on [11,23] it is worthwhile noting that the bounds provided by Milnor and Thom actually bound the sum of the betti numbers of algebraic varieties and not only the number of connected components. Thus it may be possible to use the dimension of the higher cohomology groups to establish lower bounds on straight line computations.

2. In some of the applications in section 6 we can easily extend the lower bound to the average case complexity. Thus for example we can prove:

Theorem 11. *Let $x_1, \dots, x_n$ be independent random variables uniformly distributed in the interval $[0, 1]$, then the expected complexity for solving the element distinctness problem for the x_i by any algebraic computation tree is at least $\Omega(n \log n)$.*

3. A basic limitation to our method is the fact that it is a "degree" based method. Given a polynomial p of degree d in n variables, the best lower bound that can be derived by our method to the complexity of evaluating p is of order $n \log d$. Thus for example the $O(n^2)$ lower bound for the knapsack problem follows because the degree of the polynomial there is $2^n - 1$. Any general method that can pass this limitation would be of great interest.

Finally, we presented a fairly general and realistic model of computation and provided basic tools for proving lower bounds for a large variety of problems under this model. Our method provides a uniform way to deal with straight line computations, decision trees, and algebraic computation trees. We hope that together with the results of Rabin [13] it serves to clarify the tradeoffs involved between arithmetical operations and comparisons.

References

[1] W. Baur and V. Strassen, *The complexity of partial derivatives.* to appear (1982).

[2] A. Borodin and I. Munro, *Computational complexity of algebraic and numeric problems.* American Elsevier, 1975.

[3] D. Dobkin, *A nonlinear lower bound on linear search tree programs for solving knapsack problems.* JCSS 13, (1976) 69–73.

[4] D.P. Dobkin and R.J. Lipton, *A lower bound of $\frac{1}{2}n^2$ on linear search programs for the knapsack problem.* JCSS 16, (1978) 413–417.

[5] D.P. Dobkin and R.J. Lipton, *On the complexity of computations under varying sets of primitives.* JCSS 18, (1979) 86–91.

[6] M.L. Fredman and B. Weide, *On the complexity of computing the measure of $\bigcup[a_i, b_i]$.* CACM 21, (1978) 540–544.

[7] D. Hilbert, *Foundations of geometry,* 1899. Edited and reprinted by Open Court, 1971.

[8] J.W. Jaromczyk, *Lower bounds for problems defined by polynomial inequalities.* Inter. FCT conference, Hungary, August 1981, F. Gecseg Ed. (Lecture Notes in Computer Science 117), Springer—Verlag, 165–172.

[9] J.W. Jaromczyk, *An extension of Rabin's complete proof concept.* Mathematical Foundations of Computer Science 1981, J. Gruska and M. Chytill Ed. (Lecture Notes in Computer Science 118), Springer—Verlag, 321–326.

[10] Lemoine, *Géométrographie,* 1907.

[11] J. Milnor, *On the betti numbers of real algebraic varieties.* Proc. AMS 15, (1964) 275–280.

[12] J. Milnor, *Singular points of complex hypersurfces.* Princeton Univ. Press, 1968.

[13] M.O. Rabin, *Proving simultaneous positivity of linear forms.* JCSS 6, (1972) 639–650.

[14] M.O. Rabin, unpublished lecture notes (1977).

[15] E.M. Reingold, *On the optimality of some set algorithms.* JACM 19, (1972) 649–659.

[16] A. Schmitt, *On the computational power of the floor function.* Info. Proc. Let. 14, (1982) 1–3.

[17] C.P. Schnorr, *An extension of Strassen's degree bound.* SIAM J. Comput. 10, (1981) 371–382.

[18] M.I. Shamos, *Geometric complexity.* Proc. 7th ACM STOC, Albuqueque, New Mexico, (May 1975) 224–233.

[19] M.I. Shamos, *Problems in computational geometry,* 1975.

[20] J.M. Steele and A.C. Yao, *Lower bounds for algebraic decision trees.* J. Algorithms 3, (1982) 1–8.

[21] V. Strassen, *Die Berechnungskomplexität von elementarsymetrischen funktionen und von interpolationskoeffizienten.* Numer. Math. 20, (1973) 238–251.

[22] V. Strassen, *The computational complexity of continued fractions.* Proc. of the 1981 ACM symposium on symbolic and algebraic computation, Utah, (August 1981) 51–67.

[23] R. Thom, *Sur l'homologie des variétés algébriques réelles.* Differential and Combinatorial Topology, Ed. S.S. Cairns, Princeton Univ. Press, 1965.

[24] A.C. Yao, *On the complexity of comparison problems using linear functions.* Proc. 16th STOC, Berkeley 1975, 85–89.

[25] A.C. Yao, *A lower bound to finding convex hulls.* Stanford Computer Science Report STAN-CS-79-733, (1979).

[26] A.C. Yao and R.L. Rivest, *On the polyhedral decision problem.* SIAM J. Comput. 9, (1980) 343–347.