Are Lower Bounds Easier over the Reals ?*

Hervé Fournier and Pascal Koiran
[hfournie,koiran]@ens.ens-lyon.fr

October 10, 1997

Abstract

We show that proving lower bounds in algebraic models of computation may not be easier than in the standard Turing machine model. For instance, a superpolynomial lower bound on the size of an algebraic circuit solving the real knapsack problem (or on the running time of a real Turing machine) would imply a separation of P from PSPACE. A more general result relates parallel complexity classes in boolean and real models of computation. We also propose a few problems in algebraic complexity and topological complexity.

Keywords: algebraic complexity, decision trees, range searching, lower bounds.

1 Introduction

One important motivation for the study of algebraic complexity is the search for better lower bounds than in boolean models of computation. This hope has been fulfilled to a large extent. In particular, there is a large body of work on lower bounds for linear or algebraic decision trees. Here, one of the seminal papers is the quadratic lower bound for the knapsack problem by Dobkin and Lipton [8] (other early results can be found in [18] and [20]). Nevertheless, the ultimate goal of proving superpolynomial lower bounds for natural problems has remained elusive. Sometimes this is due simply to the fact there *is* no superpolynomial lower bound: Meyer auf der Heide [14] has constructed linear decision trees (or *linear search algorithms* in his terminology) of polynomial depth for Knapsack. As he puts it, this "destroys the hope of proving nonpolynomial lower bounds for this NP-complete problem in the model of linear search algorithms." One can still try to prove a superpolynomial lower bound for Knapsack in more realistic

*A part of this work was done while the authors were visiting the Liu Bie Ju Center for Mathematical Sciences at the City University of Hong-Kong.

(i.e., less powerful) computation models, e.g., arithmetic circuits, or, if one wants a uniform model of computation, the real Turing machine of Blum, Shub and Smale [4]. This has remained an open problem to this date.

In this paper, we show that Meyer auf der Heide's result effectively destroys the hope of proving a superpolynomial lower bound for Knapsack in these less powerful models. Indeed, we show that a superpolynomial lower bound on the circuit size (or *a fortiori* on the time on a real Turing machine) for Knapsack implies $\mathrm{P} \neq \mathrm{PSPACE}$. In other words:

Proposition 1 *If* $\mathrm{P} = \mathrm{PSPACE}$*, Knapsack can be solved in polynomial time on a real Turing machine.*

Although widely believed to be true, the separation of P from PSPACE is a notorious open problem. This shows in a precise sense that lower bounds over the reals are not easier than in boolean models of computation.

Our proof is based on Meyer auf der Heide's construction. In fact there is a more general result, based on a subsequent paper by the same author [15]. The main result in that paper implies that problems in $\mathrm{PAR}^0_{\mathbb{R}_{ovs}}$ can be solved by linear decision trees of polynomial depth. Here PAR stands for "parallel polynomial time" and the notation $\mathbb{R}_{ovs}$ is meant to recall that we consider $\mathbb{R}$ as an ordered vector space, i.e., the only legal operations are $+$, $-$ and $<$ (see [2, 7] for more information on parallel complexity classes in the BSS model). The superscript 0 means that real parameters are not allowed in a machine's program (0 and 1 are the only allowed constants). Meyer aud der Heide used a somewhat different model of computation: he worked with parallel random access machines performing arithmetic operations on integers at unit cost. One can check that the result (and its proof) still hold for real inputs.

The class of problems that can be solved in polynomial time by parameter-free real Turing machines is denoted $\mathrm{P}^0_{\mathbb{R}_{ovs}}$. It also makes sense to work with real Turing machine which can use arbitrary real parameters. The corresponding classes are denoted $\mathrm{P}_{\mathbb{R}_{ovs}}$ and $\mathrm{PAR}_{\mathbb{R}_{ovs}}$. We shall prove the following.

Theorem 1 $\mathrm{P}_{\mathbb{R}_{ovs}} = \mathrm{PAR}_{\mathbb{R}_{ovs}}$ *if and only if* $\mathrm{P/poly} = \mathrm{PSPACE/poly}$*, and* $\mathrm{P}^0_{\mathbb{R}_{ovs}} = \mathrm{PAR}^0_{\mathbb{R}_{ovs}}$ *if and only if* $\mathrm{P} = \mathrm{PSPACE}$.

In the theory of computation over the reals, PAR plays the same role as PSPACE in the classical theory (and in fact $\mathrm{PAR} = \mathrm{PSPACE}$ for the standard structure $\{0, 1\}$). Theorem 1 can therefore be viewed as a transfer result for the problems $\mathrm{P} = \mathrm{PSPACE}$ and $\mathrm{P/poly} = \mathrm{PSPACE/poly}$. This is similar in spirit (but technically very different) to the transfer theorem for the problem $\mathrm{P} = \mathrm{NP}$ due to Blum, Cucker, Shub and Smale [3].

Theorem 1 implies Proposition 1 since Knapsack is in $\mathrm{PAR}^0_{\mathbb{R}_{ovs}}$. For $\mathrm{PAR}_{\mathbb{R}_{ovs}}$-complete problems there is a more precise statement: such a problem is in $\mathrm{P}_{\mathbb{R}_{ovs}}$ if and only if $\mathrm{P}_{\mathbb{R}_{ovs}} = \mathrm{PAR}_{\mathbb{R}_{ovs}}$, that is, if and only if $\mathrm{P/poly} = \mathrm{PSPACE/poly}$. This applies for instance to DTRAO, the Digital Theory of the Reals with Addition and Order [7]. For Knapsack we have seen that one direction of this implication holds, but the converse is not known to be true because Knapsack is presumably not $\mathrm{PAR}_{\mathbb{R}_{ovs}}$-complete. Nevertheless, there is a weak converse to Proposition 1: it can be shown that if Knapsack is in $\mathrm{P}_{\mathbb{R}_{ovs}}$ then the standard knapsack problem (in the Turing model) is in P/poly. This would imply P/poly = NP/poly since the standard knapsack problem is NP-complete (there is a similar remark in [11]).

Note that under the (quite unlikely) hypothesis P/poly = PSPACE/poly, Theorem 1 is actually a strengthening of Meyer auf der Heide's result since for any input size, a polynomial-time real machine can always be unwinded into a polynomial depth decision tree. This somehow suggests that one cannot avoid using his result. Note also that Theorem 1 does not hold in all structures: Knapsack is still in PAR (even in NP) over the reals with addition and equality, but it is known that it does not admit polynomial-size decision trees in that structure [10].

The remainder of this paper is organized as follows. In section 2 we give a refinement of the main result of [15] which concerns the size of coefficients in a linear decision tree. We also answer in Theorem 4 a question left open in that paper. Section 3 is devoted to the proof of Theorem 1. Finally, section 4 discusses the generalization of these results to models of computation with multiplication.

2 Coefficient Size in Decision Trees

As explained in the introduction, the following result is essentially established in [15].

Theorem 2 *Any problem in* $\mathrm{PAR}^0_{\mathbb{R}_{ovs}}$ *can be solved by a family of linear linear decision of polynomial depth.*

We recall that the internal nodes in a linear decision tree (also called a linear search algorithm, or LSA for short) are labeled by tests of the form "$l(x) \geq 0$?" where l is an affine function and $x \in \mathbb{R}^n$ is the input. Leaves are labeled 0 (reject) or 1 (accept).

Note that Theorem 2 actually holds for any problem in $\mathrm{PAR}_{\mathbb{R}_{ovs}}$: if $A \in \mathrm{PAR}^k_{\mathbb{R}_{ovs}}$, there exists $B \in \mathrm{PAR}^0_{\mathbb{R}_{ovs}}$ such that $A \cap \mathbb{R}^n$ is the restriction of $B \cap \mathbb{R}^{n+k}$ (obtained by fixing the values of the k parameters). Since B can be solved by a family of polynomial depth LSA, the same is true for its restriction.

In this section we present a refinement of Theorem 2.

Theorem 3 *Any problem in* $\mathrm{PAR}^0_{\mathbb{R}_{ovs}}$ *can be solved by a family of LSA of polynomial depth in which the test functions have integer coefficients of polynomial size.*

Before explaining the proof, we recall a few definitions. Let $H = \{h_1, \ldots, h_m\}$ a set of hyperplanes in $\mathbb{R}^n$. We denote by h_i^+ and h_i^- the two open halfspaces defined by h_i. For a point x in $\mathbb{R}^n$, let $\mathrm{pv}_i(x) = +$ if $x \in h_i^+$, $\mathrm{pv}_i(x) = -$ if $x \in h_i^-$ and $\mathrm{pv}_i(x) = 0$ if x is on h_i. The *position vector* of x is $\mathrm{pv}(x) = (\mathrm{pv}_1(x), \ldots, \mathrm{pv}_m(x))$. The set of points that have a given position vector, if not empty, is called a face. The partition of $\mathbb{R}^n$ into faces is called the arrangement of H, and is denoted $\mathcal{A}(H)$. We define the dimension of a face f to be the dimension of its affine closure. A face of dimension k is called a k-face. A n-face is called a cell, and a 0-face a vertex.

Let A be a language in $\mathrm{PAR}^0_{\mathbb{R}_{ovs}}$: $A \cap \mathbb{R}^n$ is recognized in parallel time $p(n)$ by a constant-free real machine, where p is a polynomial. It is shown in [15] that $A \cap \mathbb{R}^n$ is a union of faces of $\mathcal{A}(H)$, where H is a set of hyperplanes in $\mathbb{R}^n$ defined by linear equations with integer coefficients in $[-2^{p(n)}, 2^{p(n)}]$. It is therefore sufficient to prove the following result.

Theorem 4 *Let $H = \{h_1, \ldots, h_m\}$ be a set of hyperplanes in $\mathbb{R}^n$.*

(i) The range searching problem for H can be solved by a LSA of depth $(n \log m)^{O(1)}$.

(ii) Moreover, if the coefficients of $h_1, \ldots, h_m$ are integers in $[-q, q]$, the test functions in this LSA have integer coefficients of size $(n \log q)^{O(1)}$.

Here we say that a LSA solves the range searching problem if two points of $\mathbb{R}^n$ arriving to the same leaf always belong to the same face. Part (i) answers a question of [15]. This question was almost answered in a paper by Meiser [13], the main caveat being that multiplications are used in his algorithm (thus the implicit computation model is the algebraic decision tree instead of the LSA). Range searching has been studied by many other authors, see e.g. [6] and the references there.

Part (ii) follows from a fairly straightforward analysis of the proof of (i). It is shown in [9] that this bound on coefficients can also be obtained from an analysis of the constructions in [14] and [15].

By a lemma from [15], it suffices to recognize the union $h_1 \cup \cdots \cup h_m$.

Lemma 1 (Meyer auf der Heide) *Let H be a set of hyperplanes in $\mathbb{R}^n$. If the union of these hyperplanes can be decided by a LSA $\mathcal{T}_1$ of depth T, then the range searching problem for H can be solved by a LSA $\mathcal{T}_2$ of depth $2T$. Moreover, the hyperplanes appearing in the nodes of $\mathcal{T}_1$ and $\mathcal{T}_2$ are the same.*

In the remainder of this section, we present our algorithm for recognizing $h_1 \cup \cdots \cup h_m$. This algorithm is a modification of Meiser's, and we refer to his paper for any unexplained notion.

Given a finite set R of hyperplanes in $\mathbb{R}^n$, $\Delta\mathcal{A}(R)$ denotes the *triangulated arrangement* of R. The importance of triangulations stems from the following fact.

Lemma 2 (Meiser) *For any set H of m hyperplanes in $\mathbb{R}^n$, and any ε, $0 < \varepsilon < 1$, there is a set $R \subseteq H$ of $r = O((n^2/\varepsilon)\log^2(n/\varepsilon))$ hyperplanes such that no cell of $\Delta\mathcal{A}(R)$ is intersected by more than εm hyperplanes of H.*

The algorithm works as follows: let R be the subset of H given by Lemma 2 for $\varepsilon = 1/2$, and $r = |R|$. The position of an input x in $\mathcal{A}(R)$ can be computed in depth $2r$ by testing in turn the position of x with respect to each hyperplane of R. The r leaves corresponding to the hyperplanes of R can be labeled *accept*. If x does not lie on an hyperplane of R, it belongs to a cell c of $\mathcal{A}(R)$. We now describe a method for locating x in the triangulation of c.

2.1 Structure of Triangulations

We first recall how the triangulation Δp of a bounded polytope p of $\mathbb{R}^n$ is built. Let z^0 be a vertex of p and $\{f_1, \ldots, f_s\}$ the set of $(d-1)$-faces of p that are not adjacent to z^0. The collection of cells forming Δp is $\{\mathrm{conv}(z^0, f), f \in \Delta f_1 \cup \ldots \cup \Delta f_n\}$, where $\mathrm{conv}(z^0, f)$ is the interior of the convex hull of $z^0 \cup f$.

The case of an unbounded polytope p is a little more involved. For $x \in p$, $\mathrm{cc}_x(p)$ is defined as $\{y;\ \forall\lambda \geq 0\ x + \lambda y \in p\}$. This set does not depend on x. It is called the *characteristic cone* of p and denoted $\mathrm{cc}(p)$. First, we apply the triangulation algorithm for bounded polytopes to p. However, the cone $C = z^0 + \mathrm{cc}(p)$ will not be covered by the elementary cells obtained. We can make sure that C is line-free by adding to H the set of hyperplanes $\{x_1 = 0, \ldots, x_d = 0\}$. Then C can be triangulated as follows: let h be a hyperplane such that $h \cap C$ is a bounded polytope p' of h. We set $\Delta C = \{\mathrm{cone}(z^0, f), f \in \Delta p'\}$, where $\mathrm{cone}(z^0, f)$ is the interior of the cone of apex z^0 and base f.

2.2 Point Location in a Triangulated Polytope

We now explain how to locate a point x in the triangulation of a cell c of $\mathcal{A}(R)$. More precisely, we want to find a (possibly unbounded) simplex s of $\Delta\mathcal{A}(c)$ such that x belongs to the closure of s. Let z^0 be the vertex that has been used to triangulate c, and $\{f_1^0, \ldots, f_{n_0}^0\}$ the set of $(n-1)$-faces of c that are not adjacent to z^0. In the first step, we compute i such that $x \in \mathrm{conv}(z^0, f_i^0)$. Since the n_0 faces under consideration are bounded by at most r hyperplanes, and $n_0 \leq r$, this can be done in time $O(r^2)$. Let z^1 be the vertex that has been used to triangulate f_i^0, and $\{f_1^1, \ldots, f_{n_1}^1\}$ the set of $(n-2)$-faces of f_i^0 that are not adjacent to z^1. In the second step, we determine in time $O(r^2)$ a face f_i^1 such that $x \in \mathrm{conv}(z^0, z^1, f_i^1)$. One can keep going down the hierarchy of triangulations in the same way, and eventually determine after $n-1$ steps a simplex s of $\Delta\mathcal{A}(c)$ such that x belongs to the closure of s (if c is unbounded, we also have to test if $x \in \mathrm{cone}(z^0, p')$ in the first step; if this is the case, the following steps consist of going down the hierarchy of cones of apex z^0 induced by the triangulation of p'). Since each step can be performed in depth $O(r^2)$, the depth of the LSA locating x in $\Delta\mathcal{A}(R)$ is $O(nr^2)$.

2.3 Recursion

According to Lemma 2, the set H_1 of hyperplanes of H intersecting s has at most $|H|/2$ elements. There are two cases.

(a) If x belongs to s, we call the algorithm recursively with H replaced by H_1.

(b) If x is on the boundary of s, let h be the affine closure of a $(n-1)$-face of s such that $x \in h$. If $h \in H$ we accept x. Otherwise, we proceed as in (a).

2.4 Analysis of the Algorithm

Let $T(m)$ be the depth of the LSA deciding the union of m hyperplanes. We have $T(m) \leq O(nr^2) + T(m/2)$ for $m > r$ for r as in Lemma 2, $T(m) = O(r)$ otherwise. Thus $T(m) = O(nr^2 \log m) = O(n^5 \log^4 n \log m)$. This completes the proof of (i).

Let us now assume that the hyperpanes of H have integer coefficients in $[-q, q]$. Each hyperplane appearing as a test function in the LSA is the affine closure of a union of faces of $\mathcal{A}(H)$. Every such hyperplane is therefore the affine closure of n vertices $p_1, \ldots, p_n$ of $H' = H \cup \{x_1 = 0, x_1 = 1, \ldots, x_n = 0, x_n = 1\}$. By Cramer's rule, p_i is a rational point $(a_{i1}/u_i, \ldots, a_{in}/u_i)$, with $|a_{ij}|, |u_i| \leq B = q^n n^{n/2}$. An equation of the hyperplane h containing $p_1, \ldots, p_n$ is $\det(x - p_1, p_2 - p_1, \ldots, p_n - p_1) = 0$. Multiplying the columns of this determinant by $u_1, (u_1 u_2), \ldots, (u_1 u_n)$ respectively, we obtain an equation for h with integer coefficients no larger than $n!(2B)^{2n}$. Thus the length of each coefficient is bounded by $n^2 \log n + 2n^2 \log q + O(n^2)$. This completes the proof of Theorem 4.

3 Proof of the Main Result

Let us recall that a real language (or *problem*) is a subset of $\mathbb{R}^\infty = \bigcup_{n \geq 0} \mathbb{R}^n$, and that the boolean part $\mathcal{BP}(\mathcal{C})$ of a class $\mathcal{C}$ of real problems is the set of boolean problems of the form $A \cap \{0,1\}^*$ where $A \in \mathcal{C}$.

3.1 Boolean Parts

We need some characterizations of boolean parts. In the parameter-free case there is almost nothing to prove; see [10] and [7] for the general (parameters allowed) case.

Fact 1 $\mathcal{BP}(\mathrm{P}_{\mathbb{R}_{ovs}}) = \mathrm{P/poly}$ *and* $\mathcal{BP}(\mathrm{P}^0_{\mathbb{R}_{ovs}}) = \mathrm{P}$*;* $\mathcal{BP}(\mathrm{PAR}_{\mathbb{R}_{ovs}}) = \mathrm{PSPACE/poly}$ *and* $\mathcal{BP}(\mathrm{PAR}^0_{\mathbb{R}_{ovs}}) = \mathrm{PSPACE}$.

Fact 2 *Let B be a problem in $\mathrm{PAR}^0_{\mathbb{R}_{ovs}}$. For every $n \geq 0$ there exists a quantifier-free formula F_n in the theory of the reals with addition and order defining $B \cap \mathbb{R}^n$. The atomic predicates in F_n are of the form $l(x) \geq 0$ or $l(x) > 0$, where l is an affine function with integer coefficients of polynomial size.*

It is also possible to give a single-exponential bound on the number of atomic predicates in F_n, but we do not need this here.

It is shown in the next result that nondeterminism does not increase the power of parallel polynomial time. We work with the following definitions: a problem $A \subseteq \mathbb{R}^\infty$ is in $\mathrm{NPAR}_{\mathbb{R}_{ovs}}$ (respectively, $\mathrm{NPAR}^0_{\mathbb{R}_{ovs}}$) if there exists a *corresponding problem* $B \in \mathrm{PAR}_{\mathbb{R}_{ovs}}$ (respectively, $B \in \mathrm{PAR}^0_{\mathbb{R}_{ovs}}$) and a polynomial p such that for any $n \geq 0$ and $x \in \mathbb{R}^n$, $x \in A$ iff

$$\exists y \in \mathbb{R}^{p(n)} \langle x, y \rangle \in B. \tag{1}$$

Theorem 5 $\mathrm{NPAR}_{\mathbb{R}_{ovs}} = \mathrm{PAR}_{\mathbb{R}_{ovs}}$ *and* $\mathrm{NPAR}^0_{\mathbb{R}_{ovs}} = \mathrm{PAR}^0_{\mathbb{R}_{ovs}}$.

Proof. Obviously, $\mathrm{PAR}_{\mathbb{R}_{ovs}} \subseteq \mathrm{NPAR}_{\mathbb{R}_{ovs}}$ and $\mathrm{PAR}^0_{\mathbb{R}_{ovs}} \subseteq \mathrm{NPAR}^0_{\mathbb{R}_{ovs}}$. Let us show the converse inclusion $\mathrm{NPAR}^0_{\mathbb{R}_{ovs}} \subseteq \mathrm{PAR}^0_{\mathbb{R}_{ovs}}$. Thus, let $A \in \mathrm{NPAR}^0_{\mathbb{R}_{ovs}}$, and let $B \in \mathrm{PAR}^0_{\mathbb{R}_{ovs}}$ be the corresponding problem. We claim that there exists a polynomial q such that this existential formula is satisfied iff it is satisfied by a point $y \in \mathbb{R}^{p(n)}$ all of whose components are of the form $y_i = l_i(x)$ where l_i is an affine function with rational coefficients of size at most $q(n)$. This will show that $A \in \mathrm{PAR}^0_{\mathbb{R}_{ovs}}$ since one can then try in parallel all such values of y to decide whether $x \in A$ (divisions can be avoided by storing separately numerators and denominators).

To prove the claim, assume that (1) holds for a given x. By Fact 2, $\langle x, y \rangle \in B$ iff (x, y) satisfies a formula $F_n(x, y)$ with coefficients of polynomial size. We can assume that F_n is a disjunction $\bigvee_{i=1}^{m_n} C_{i,n}$ of conjunctions of linear inequalities. Since (1) is valid, one conjunction $C_{i,n}$ must be valid. From the existence of "small" points in polyhedra (Theorem 2 in [7]; note that the number of inequations does not appear in that bound) we conclude that $C_{i,n}$, and hence F_n, is satisfied by a point y of the required form. This completes the proof that $\mathrm{NPAR}^0_{\mathbb{R}_{ovs}} = \mathrm{PAR}^0_{\mathbb{R}_{ovs}}$ (note the similarity with the proof of Theorem 3 in [7]).

Finally, we show that $\mathrm{NPAR}_{\mathbb{R}_{ovs}} \subseteq \mathrm{PAR}_{\mathbb{R}_{ovs}}$. For any $A \in \mathrm{NPAR}_{\mathbb{R}_{ovs}}$ there exists $k > 0$, $\alpha \in \mathbb{R}^k$ and $A' \in \mathrm{NPAR}^0_{\mathbb{R}_{ovs}}$ such that $x \in A$ iff $\langle x, \alpha \rangle \in A'$. But we have just seen that, in fact, $A' \in \mathrm{PAR}^0_{\mathbb{R}_{ovs}}$. This shows that $A \in \mathrm{PAR}_{\mathbb{R}_{ovs}}$ since this problem is the restriction of a $\mathrm{PAR}^0_{\mathbb{R}_{ovs}}$ problem (one could also use a version of Fact 2 adapted to $\mathrm{PAR}_{\mathbb{R}_{ovs}}$ problems to prove that result). $\square$

Corollary 1 $\mathcal{BP}(\mathrm{NPAR}_{\mathbb{R}_{ovs}}) = \mathrm{PSPACE/poly}$ *and* $\mathcal{BP}(\mathrm{NPAR}^0_{\mathbb{R}_{ovs}}) = \mathrm{PSPACE}$.

3.2 Exploring a Linear Decision Tree

To a problem A of $\mathbb{R}^\infty$ we associate a boolean problem $\tilde{A}$. An instance of $\tilde{A}$ is described by three integers n, L, d (given in unary) and a (possibly empty) system $\mathcal{S}$ of affine inequalities of the form $l(x) \geq 0$ or $l(x) < 0$. The coefficients of these inequalities are integers written in binary, and the variable x lives in $\mathbb{R}^n$. The system defines a polyhedron $P_\mathcal{S} \subseteq \mathbb{R}^n$. An instance of $\tilde{A}$ is positive if there exists a LSA T of depth at most d with coefficients of bit size at most L such that T recognizes A on $P_\mathcal{S}$ (i.e., $A \cap P_\mathcal{S} = E \cap P_\mathcal{S}$, where E is the subset of $\mathbb{R}^n$ recognized by T).

We need an algorithm to solve $\tilde{A}$, and for positive instances of this problem we also need to compute the label l_r of the root of a corresponding tree T (this tree may not be unique, but any solution will do). Thus l_r is just a boolean value if T is reduced to a leaf, and an affine inequality of the form $l(x) \geq 0$ otherwise.

Lemma 3 *If* $A \in \mathrm{PAR}^0_{\mathbb{R}_{ovs}}$ *then* $\tilde{A} \in \mathrm{PSPACE}$. *Moreover, for a positive instance* l_r *can be constructed in polynomial space.*

Proof. We first determine whether T can be of depth 0, i.e., reduced to a leaf. In that case, T recognizes either $\mathbb{R}^n$ or $\emptyset$ depending on the label of that leaf. Label 1 is *not* acceptable iff

$$\exists x \in \mathbb{R}^n \; x \in P_\mathcal{S} \setminus A \tag{2}$$

Since $A \in \mathrm{PAR}_{\mathbb{R}_{ovs}}$, the problem of deciding whether a given x and $\mathcal{S}$ satisfy $x \in P_\mathcal{S} \setminus A$ is $\mathrm{PAR}_{\mathbb{R}_{ovs}}$. Hence by Corollary 1, deciding (2) is a PSPACE problem. Label 0 is *not* acceptable iff $\exists x \in \mathbb{R}^n \; x \in P_\mathcal{S} \cap A$. This is also a PSPACE problem.

If there is a solution in depth 0, we accept the instance of $\tilde{A}$ and output the corresponding label. Otherwise, for $d > 0$ we look for solutions of depth between 1 and d (for $d = 0$ we exit and reject the instance). To do this we enumerate (e.g. in lexicographic order) all possible linear inequalities of the form $l(x) \geq 0$ where the coefficients of l are of bit size at most L. For each such inequality we do the following.

1. Decide by a recursive call whether $(n, L, d-1, \mathcal{S} \cup \{l(x) \geq 0\})$ is a positive instance of $\tilde{A}$.

2. Decide by a recursive call whether $(n, L, d-1, \mathcal{S} \cup \{l(x) < 0\})$ is a positive instance of $\tilde{A}$.

3. In case of a positive answer to both questions, exit the loop, accept $(n, L, d, \mathcal{S})$ and output $l_r = l$.

The instance is rejected if it is not accepted in the course of this enumeration procedure.

In addition to the space needed to solve the depth 0 case, we just need to maintain a stack to keep track of recursive calls. Hence this algorithm runs in polynomial space, showing that $\tilde{A} \in$ PSPACE. For positive instances, the algorithm also outputs l_r as needed. $\square$

Before proving Theorem 1 we need an intermediate result.

Theorem 6 *Let A be a problem of* $\mathrm{PAR}^0_{\mathbb{R}_{ovs}}$ *which can be solved by a polynomial-depth LSA with coefficients of polynomial size.* P/poly = PSPACE/poly *implies* $A \in \mathrm{P}_{\mathbb{R}_{ovs}}$, *and* P = PSPACE *implies* $A \in \mathrm{P}^0_{\mathbb{R}_{ovs}}$.

Proof. For inputs of size n, A can be solved by a tree of depth and coefficient size bounded by an^b, where a and b are constants. The idea is to use Lemma 3 to move down that tree. The hypothesis P/poly = PSPACE/poly implies that $\tilde{A} \in$ P/poly. Moreover, for positive instances l_r can be constructed in polynomial time with polynomial advice (one should argue that each bit of l_r is in PSPACE, and therefore in P/poly). Thus we set $L = d = an^b$ and $\mathcal{S} = \emptyset$. By hypothesis $(n, L, d, \mathcal{S})$ is a positive instance of $\tilde{A}$ and therefore l_r can be computed in polynomial time (with polynomial advice). If l_r is a boolean value we stop and output that value. Otherwise l_r is an affine function, and we can determine in polynomial time whether the input $x \in \mathbb{R}^n$ to A satisfies $l_r(x) \geq 0$. If so, we set $\mathcal{S}' = \mathcal{S} \cup \{l_r \geq 0\}$. Otherwise, we set $\mathcal{S}' = \mathcal{S} \cup \{l_r < 0\}$. In any case, we set $d' = d - 1$, and feed $(n, L, d', \mathcal{S}')$ to the algorithm for $\tilde{A}$. This process continues until a leaf is reached. This requires at most an^b steps.

The above algorithm runs in polynomial time with polynomial advice (in fact, the only advice used is the advice needed to solve instances of $\tilde{A}$ of the form $(n, L, d, \mathcal{S})$ where $L, d \leq an^b$ and $\mathcal{S}$ is a system of at most an^b inequalities of coefficient size bounded by an^b). That advice can be encoded in the digits of a real constant and retrieved in polynomial time, showing that $A \in \mathrm{P}_{\mathbb{R}_{ovs}}$. If P = PSPACE then no advice is needed, hence $A \in \mathrm{P}^0_{\mathbb{R}_{ovs}}$. $\square$

Proof of Theorem 1. Let us do the "easy" direction first. The boolean problem QBF is a well-known PSPACE-complete problem. It is clearly in $\mathrm{PAR}^0_{\mathbb{R}_{ovs}}$, hence $\mathrm{P}^0_{\mathbb{R}_{ovs}} = \mathrm{PAR}^0_{\mathbb{R}_{ovs}}$ implies QBF $\in \mathrm{P}^0_{\mathbb{R}_{ovs}}$. Thus QBF $\in$ P since $\mathcal{BP}(\mathrm{P}^0_{\mathbb{R}_{ovs}}) = \mathrm{P}$. We conclude that P = PSPACE by the completeness of QBF. If we only assume that $\mathrm{P}_{\mathbb{R}_{ovs}} = \mathrm{PAR}_{\mathbb{R}_{ovs}}$, we obtain QBF $\in$ P/poly since $\mathcal{BP}(\mathrm{P}_{\mathbb{R}_{ovs}}) = \mathrm{P/poly}$. This implies PSPACE $\subseteq$ P/poly (or equivalently, P/poly = PSPACE/poly).

For the converses, we know from Theorem 3 that any problem in $\mathrm{PAR}^0_{\mathbb{R}_{ovs}}$ can be solved by a polynomial-depth LSA with coefficients of polynomial size. Thus P = PSPACE implies $\mathrm{P}^0_{\mathbb{R}_{ovs}} = \mathrm{PAR}^0_{\mathbb{R}_{ovs}}$ by Theorem 6.

Now, let A be a problem in $\mathrm{PAR}_{\mathbb{R}_{ovs}}(\alpha_1, \ldots, \alpha_k)$. This problem is the restriction of a higher-dimensional $\mathrm{PAR}^0_{\mathbb{R}_{ovs}}$ problem. That is, there exists $B \in \mathrm{PAR}^0_{\mathbb{R}_{ovs}}$ such that for any $x \in \mathbb{R}^n$, $x \in A$ if and only if $(x_1, \ldots, x_n, \alpha_1, \ldots, \alpha_k) \in B$. By Theorem 6, P/poly = PSPACE/poly implies $B \in \mathrm{P}_{\mathbb{R}_{ovs}}$. Thus $A \in \mathrm{P}_{\mathbb{R}_{ovs}}$ since it is in fact the restriction of a $\mathrm{P}_{\mathbb{R}_{ovs}}$ problem. $\square$

4 Multiplicative Models

This section is made mostly of problems and conjectures.

4.1 Algebraic Complexity

Extending Theorem 1 to models of computation with multiplication seems to be a challenging problem. In that context, it is natural to work with algebraic decision trees instead of LSA. We recall that the internal nodes in an algebraic decision tree are divided into *computation nodes* and *branch nodes*. A computation node c has a single child and is labeled by an expression of the form $\alpha := \beta \perp \gamma$ where $\perp \in \{+, -, \times\}$. Here α is the *node variable*; β and γ are constants from $\mathbb{R}$ or variables above c (a variable is said to be above c if it is one of the n input variables or labels a computation node on the path from the root of the tree to c). A branch node b has two children and is labeled by an expression of the form "$\alpha \geq 0$?" where α is a variable above b. Finally, the leaves of the tree are labeled 0 or 1. The tree computes a boolean-valued function on $\mathbb{R}^n$ in the usual way. We propose the following conjecture.

Conjecture 1 *Any problem in* $\mathrm{PAR}_{\mathbb{R}}$ *can be solved by a family of polynomial-depth algebraic decision trees.*

It would already be interesting to solve this conjecture for specific problems in $\mathrm{PAR}_{\mathbb{R}}$, e.g., for the linear programming problem (feasibility of systems of the form "$Ax \geq b$").

One can also consider algebraic decision trees over $\mathbb{C}$. In this case, tests are of the form "$\alpha = 0$?". We conjecture that the situation is significantly different than in the real case.

Conjecture 2 *Hilbert's Nullstellensatz, Twenty Questions and* $\mathrm{Knapsack}_{\mathbb{C}}$ *cannot be solved by families of polynomial-depth algebraic decision trees.*

We recall that an input to Hilbert's Nullstellensatz (HN) is a system of polynomial equations in several complex variables. The input is accepted if this system has a solution. Twenty Questions was introduced in [16]: an input $x \in \mathbb{C}^n$ is accepted if the first component x_1 is an integer between 1 and 2^n.

These three problems are in $\mathrm{PAR}_{\mathbb{C}}$, and even in $\mathrm{NP}_{\mathbb{C}}$. Thus the conjecture implies $\mathrm{P}_{\mathbb{C}} \neq \mathrm{NP}_{\mathbb{C}}$. It follows from the $\mathrm{NP}_{\mathbb{C}}$-completeness of HN [2, 4] that if the conjecture is true for Twenty Questions or $\mathrm{Knapsack}_{\mathbb{C}}$, it is also true for HN. Note that the restrictions of $\mathrm{Knapsack}_{\mathbb{C}}$ and Twenty Questions to $\mathbb{R}$ *can* be solved by real algebraic decision trees of polynomial depth. In fact, any finite subset $\{a_1, \ldots, a_m\}$ of $\mathbb{R}$ can be recognized in depth $O(\log m)$ by the obvious binary search algorithm. This is not so over the complex numbers.

Proposition 2 *A finite subset* $\{a_1, \ldots, a_m\} \subseteq \mathbb{C}$ *cannot be recognized in depth* $m - 1$ *if the* a_i*'s are algebraically independent over* $\mathbb{Q}$.

This result appears in [5] as a corollary to a much more general theorem. We give a proof below since it can be sketched from scratch in a few lines. First, recall that the *canonical path* in an algebraic decision tree is the path followed by a Zariski dense subset of the inputs. It is obtained by answering *no* to each test "$\alpha = 0$?" encountered during a computation (we assume without loss of generality that all the α's represent non-constant polynomials in the input variables).

Proof of Proposition 2. Consider a tree of depth d recognizing a finite subset $E \subseteq \mathbb{C}$. Let $\alpha_1, \ldots, \alpha_j$ be the complex parameters appearing on the canonical path. We may assume that at most one new parameter is introduced at each node, so $j \leq d$. Each element of E must be a root of some polynomial computed along the canonical path. These polynomials have coefficients in $\mathbb{Q}[\alpha_1, \ldots, \alpha_j]$ and roots in $K = \overline{\mathbb{Q}[\alpha_1, \ldots, \alpha_j]}$. This field has transcendence degree at most j, hence $j \geq m$ if K is to include $\{a_1, \ldots, a_m\}$. $\Box$

As a side remark, we note that a similar but even simpler argument shows that Twenty Questions is a witness to the separation of P from NP in the BSS model with addition and equality only. This is a simpler problem than Knapsack (used in [10]) or its multidimensional version (used in [12] for the original proof of this separation).

4.2 Topological Complexity

As a first step towards a positive solution to Conjecture 1, one may attempt to work with decision trees in which internal nodes are labeled by tests of the form "$P(x) \geq 0$?" where P can be any polynomial (thus one assumes that computing an arbitrary polynomial in the input variables takes unit time). This leads to the subject of topological complexity as studied by Smale [17] and Vassiliev [19]. The corresponding trees will be called *topological decision trees* to distinguish them from ordinary algebraic decision trees.

It is not clear whether any problem in $\mathrm{PAR}_{\mathbb{R}}$ can be solved by a family of polynomial-depth topological decision trees. We have seen that the corresponding problem has a positive answer for the reals with addition and order. An important ingredient of the proof was the fact that the range searching problem for arrangements of hyperplanes can be solved in polynomial depth. It seems therefore natural to investigate the complexity of range searching in semi-algebraic sets.

Problem 1 *Is it possible to solve the range searching problem for m polynomials of degree d in n variables by a topological decision tree of depth $(nd \log m)^{O(1)}$? or even depth $(n \log(md))^{O(1)}$?*

Here we say that a tree solves the range searching problem for the polynomials $P_1, \ldots, P_m$ if two input points arriving to the same leaf always belong to the same face. As in the linear case, a face is the set of points satisfying one of the 3^m "sign conditions" of the form $P_1(x)\ \sigma_1\ 0; \ldots ; P_m(x)\ \sigma_m\ 0$ where $\sigma_i \in \{<, =, >\}$.

Range searching in semi-algebraic sets has been studied mostly for polynomials of bounded degree [1]. In this case, it is not hard to see that algebraic decision trees of depth $O(n \log m)^{O(1)}$ can solve the problem (one can make a reduction to the linear case by introducing new variables representing all monomials of degree at most d).

Acknowledgment

Meiser's paper was pointed out to the authors by Friedhelm Meyer auf der Heide.

References

[1] P. K. Agarwal and J. Matoušek. On range searching with semialgebraic sets. *Discrete and Computational Geometry*, 11(4):393–418, 1994.

[2] L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation.* Springer Verlag, to appear.

[3] L. Blum, F. Cucker, M. Shub, and S. Smale. Algebraic settings for the problem "P≠NP?". In J. Renegar, M. Shub, and S. Smale, editors, *The Mathematics of Numerical Analysis*, volume 32 of *Lectures in Applied Mathematics*, pages 125–144. American Mathematical Society, 1996.

[4] L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bulletin of the American Mathematical Society*, 21(1):1–46, July 1989.

[5] P. Bürgisser, T. Lickteig, and M. Shub. Test complexity of generic polynomials. *Journal of Complexity*, 8:203–215, 1992.

[6] B. Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete and Computational Geometry*, 9:145–158, 1993.

[7] F. Cucker and P. Koiran. Computing over the reals with addition and order: Higher complexity classes. *Journal of Complexity*, 11:358–376, 1995.

[8] D. Dobkin and R. J. Lipton. A lower bound of $(1/2)n^2$ on linear search programs for the knapsack problem. *Journal of Computer and System Sciences*, 16:413–417, 1978.

[9] H. Fournier. Localisation de points dans un arrangement d'hyperplans par un arbre de décision. Rapport de stage de MIM 2ème année, Ecole Normale Supérieure de Lyon, 1997.

[10] P. Koiran. Computing over the reals with addition and order. *Theoretical Computer Science*, 133(1):35–48, 1994.

[11] P. Koiran. A weak version of the Blum, Shub & Smale model. *Journal of Computer and System Sciences*, 54:177–189, 1997.

[12] K. Meer. A note on a $P \neq NP$ result for a restricted class of real machines. *Journal of Complexity*, 8:451–453, 1992.

[13] S. Meiser. Point location in arrangements of hyperplanes. *Information and Computation*, 106(2):286–303, 1993.

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

[15] F. Meyer auf der Heide. Fast algorithms for n-dimensional restrictions of hard problems. *Journal of the ACM*, 35(3):740–747, 1988.

[16] M. Shub and S. Smale. On the intractability of Hilbert's Nullstellensatz and an algebraic version of "P=NP". *Duke Mathematical Journal*, 81(1):47–54, 1996.

[17] S. Smale. On the topology of algorithms. I. *Journal of Complexity*, 3:81–89, 1987.

[18] J.M. Steele and A. Yao. Lower bounds for algebraic decision trees. *Journal of Algorithms*, 3:1–8, 1982.

[19] V. A. Vassiliev. *Complements of Discriminants of Smooth Maps: Topology and Applications.* Translations of Mathematical Monographs **98**. American Mathematical Society, 1994.

[20] A. Yao. On parallel computation for the knapsack problem. *Journal of the ACM*, 29(3):898–903, 1982.