Decision versus Evaluation in Algebraic Complexity

Pascal Koiran

LIP**, École Normale Supérieure de Lyon.
Pascal.Koiran@ens-lyon.fr

Abstract. This is a survey of some of my joint work with Sylvain Périfel. It is focused on transfer theorems that connect boolean complexity to algebraic complexity in the Valiant and Blum-Shub-Smale models.
Keywords: computational complexity, algebraic complexity, Blum-Shub-Smale model, Valiant's model.

1 Introduction

The goal of this note is to highlight some developments in algebraic complexity theory that have occured since my previous survey [11] was written. The scope of the present note is much narrower since I will only attempt to present some of my joint work with Sylvain Périfel.

The survey [11] contained an introduction to the Blum-Shub-Smale theory of computation in rings [3,2] and its extension to "arbitrary" structures by Poizat [23]; a presentation of some then-recent results, including the transfer theorems for the "P=NP ?" problem over the reals with addition and order [8,9]; and some suggestions of future research directions as well as a few concrete steps in those directions. The main message of the paper was that in order to study the "$\mathrm{P}_M = \mathrm{NP}_M$?" problem in a structure M, it is essential to find out whether NP_M problems can be solved by polynomial depth computation trees. Indeed, we know that $\mathrm{P}_M \neq \mathrm{NP}_M$ if there exists some problem X in NP_M that cannot be solved by polynomial depth computation trees. This follows from the fact that computation trees are always at least as powerful as circuits: if problem X cannot be solved by polynomial depth computation trees then it cannot be solved by polynomial size circuits. On the other hand, if all NP_M problems can be solved by polynomial depth computation trees it may be possible to obtain a transfer theorem for the "$\mathrm{P}_M = \mathrm{NP}_M$?" problem. It turns out that the structure M of the real numbers with addition and equality complies with the first branch of this "computation tree alternative". Indeed, it has been known for a long time that this structure satisfies $\mathrm{P}_M \neq \mathrm{NP}_M$ [19]. By contrast, it is known that for the structure of the real numbers with addition and order all NP_M problems can be solved by polynomial depth computation trees. This follows from results of computational geometry due Meiser [20] and Meyer auf

** UMR 5668 ENS Lyon, CNRS, UCBL, INRIA.

der Heide [21,22]. Those results are instrumental in the proof of our transfer theorems [8,9], which show that the "P = NP ?" problem in that structure is equivalent to the classical "P = NP ?" problem[1]. For the ordered field of the real numbers, the "$\mathrm{P}_{\mathbb{R}} = \mathrm{NP}_{\mathbb{R}}$?" problem is still open but the following result was established in [11].

Theorem 1. *If* $\mathrm{NP}_{\mathbb{R}}$ *problems can be solved by constant-free computation trees of polynomial depth, we have the following transfer theorem:* P = PSPACE *implies* $\mathrm{P}_{\mathbb{R}} = \mathrm{NP}_{\mathbb{R}}$.

It is still not known whether the hypothesis that $\mathrm{NP}_{\mathbb{R}}$ problems can be solved by computation trees of polynomial depth holds true, and my guess is that it doesn't[2]. Nevertheless, we recently managed to make some progress on these questions by bringing in a new ingredient, namely, Valiant's own algebraic version of the "P = NP?" problem [25,26,4]. In Valiant's framework one studies the complexity of evaluating polynomials rather than the complexity of decision problems as in the boolean or Blum-Shub-Smale frameworks. In [14] and [13] we showed that a collapse of complexity classes in Valiant's framework implies $\mathrm{P}_{\mathbb{R}} = \mathrm{NP}_{\mathbb{R}}$ and $\mathrm{P}_{\mathbb{C}} = \mathrm{NP}_{\mathbb{C}}$. These results and some relevant background are presented in the remainder of this paper.

2 The Blum-Shub-Smale Model

Although the original definitions of Blum, Shub and Smale [3,2] are in terms of uniform machines, we will follow [23] by using families of algebraic circuits to recognize languages over a field K, that is, subsets of $K^\infty = \bigcup_{n\geq 0} K^n$. The ordinary (boolean) theory of computation is recovered by choosing for K the two-element field $\mathbb{Z}/2\mathbb{Z}$. A similar circuit model was defined by von zur Gathen [28] (see also [1], section 4.1).

An algebraic circuit is a directed acyclic graph whose vertices, called gates, have indegree 0, 1 or 2. An input gate is a vertex of indegree 0. An output gate is a gate of outdegree 0. We assume that there is only one such gate in the circuit. Gates of indegree 2 are labelled by a symbol from the set $\{+, -, \times\}$. Gates of indegree 1, called test gates, are labelled "$= 0$?" if K is unordered (e.g., if $K = \mathbb{C}$), or "≤ 0?" if K is ordered (e.g., if $K = \mathbb{R}$). The size of a circuit C, in symbols $|C|$, is the number of vertices of the graph.

A circuit with n input gates computes a function from K^n to K. On input $\bar{u} \in K^n$ the value returned by the circuit is by definition equal to the value of its output gate. The value of a gate is defined in the usual way. Namely, the value of

[1] Strictly speaking, equivalence to the classical problem only holds true in the constant-free version of the Blum-Shub-Smale model; in the full-fledged model arbitrary real constants are allowed, and we have proved equivalence to the non-uniform problem "P/poly = NP/poly ?".

[2] I will be proved right once since we made the opposite conjecture in [8]. Even for the linear programming problem (feasibility of sytems of linear inequalities), it seems that the answer to this question is not known.

input gate number i is equal to the i-th input u_i. The value of other gates is then defined recursively: it is the sum of the values of its entries for a $+$-gate, their difference for a $-$-gate, their product for a $\times$-gate. If K is unordered, the value taken by a test gate is 1 if the value of its entry is equal to 0, and 0 otherwise. If K is ordered, the value taken by a test gate is 1 if the value of its entry is ≤ 0 and 1 otherwise. We assume without loss of generality that the output is a test gate. The value returned by the circuit is therefore 0 or 1.

The class P_K is the set of languages $L \subseteq K^\infty$ such that there exists a tuple $\bar{a} \in \mathbb{R}^p$ (independent of n) and a P-uniform family of polynomial-size circuits (C_n) satisfying the following condition: C_n has exactly $n+p$ inputs, and for any $\bar{x} \in \mathbb{R}^n$, $\bar{x} \in L \Leftrightarrow C_n(\bar{x}, \bar{a}) = 1$. The P-uniformity condition means that C_n can be built in time polynomial in n by an ordinary (discrete) Turing machine. Note that $\bar{a}$ plays the role of the machine constants of [2,3].

As in [6], we define the class PAR_K as the set of languages over K recognized by a PSPACE-uniform family of algebraic circuits of polynomial depth (and possibly exponential size). The same tuple of constants $\bar{a}$ is used for every input size n, as in the definition of P_K. Note at last that we could also define similar classes without the constants $\bar{a}$. We will use the superscript 0 to denote these constant-free classes, for instance P_K^0 and PAR_K^0.

3 Valiant's Model

In Valiant's model, one computes polynomials instead of recognizing languages. We thus use arithmetic circuits instead of algebraic circuits. A book-length treatment of this topic can be found in [4].

An arithmetic circuit is the same as an algebraic circuit but test gates are not allowed. That is to say we have indeterminates $x_1, \ldots, x_{u(n)}$ as input together with arbitrary constants of K; there are $+$, $-$ and $\times$-gates, and we therefore compute multivariate polynomials.

The polynomial computed by an arithmetic circuit is defined in the usual way by the polynomial computed by its output gate. Thus a family (C_n) of arithmetic circuits computes a family (f_n) of polynomials, $f_n \in K[x_1, \ldots, x_{u(n)}]$. The class $\mathrm{VP}_{\mathrm{nb}}$ defined in [17,18] is the set of families (f_n) of polynomials computed by a family (C_n) of polynomial-size arithmetic circuits, i.e., C_n computes f_n and there exists a polynomial $p(n)$ such that $|C_n| \leq p(n)$ for all n. We will assume without loss of generality that the number $u(n)$ of variables is bounded by a polynomial function of n. The subscript "nb" indicates that there is no bound on the degree of the polynomial, in contrast with the original class VP of Valiant where a polynomial bound on the degree of the polynomial computed by the circuit is required. Note that these definitions are nonuniform. The class uniform $\mathrm{VP}_{\mathrm{nb}}$ is obtained by adding a condition of polynomial-time uniformity on the circuit family, as in Section 2.

The class VNP is the set of families of polynomials defined by an exponential sum of VP families. More precisely, $(f_n(\bar{x})) \in \mathrm{VNP}$ if there exists $(g_n(\bar{x}, \bar{y})) \in \mathrm{VP}$ and a polynomial p such that $|\bar{y}| = p(n)$ and $f_n(\bar{x}) = \sum_{\bar{\epsilon} \in \{0,1\}^{p(n)}} g_n(\bar{x}, \bar{\epsilon})$.

We can also forbid constants from our arithmetic circuits in unbounded-degree classes, and define constant-free classes. The only constant allowed is 1 (in order to allow the computation of constant polynomials). As for classes of decision problems, we will use the superscript 0 to indicate the absence of constant: for instance, we will write $\mathrm{VP}_{\mathrm{nb}}^0$ (for bounded-degree classes, we are to be more careful; see [17]).

In order to state our transfer theorems it is convenient to define a new class, called VPSPACE. We will only do this for fields of characteristic 0 since those are the only fields that we consider in the remainder of this note (the extension to fields of positive characteristic is straightforward). Roughly speaking, a family of polynomials is in VPSPACE if its coefficients can be computed in polynomial space. We first define formally the notion of coefficient function. If α is a tuple $(\alpha_1, \ldots, \alpha_{u(n)})$, we denote by $\bar{x}^\alpha$ the monomial $x_1^{\alpha_1} \cdots x_{u(n)}^{\alpha_{u(n)}}$.

Definition 1. *Let (f_n) be a family of multivariate polynomials with integer coefficients. The coefficient function of (f_n) is the function a whose value on input (n, α, i) is the i-th bit $a(n, \alpha, i)$ of the coefficient of the monomial $\bar{x}^\alpha$ in f_n. Furthermore, $a(n, \alpha, 0)$ is the sign of the coefficient of the monomial $\bar{x}^\alpha$. Thus f_n can be written as*

$$f_n(\bar{x}) = \sum_\alpha \Big((-1)^{a(n,\alpha,0)} \sum_{i \geq 1} a(n, \alpha, i) 2^{i-1} \bar{x}^\alpha \Big).$$

The coefficient function is a function $a : \{0,1\}^* \to \{0,1\}$ and can therefore be viewed as a language. This allows us to speak of the complexity of the coefficient function.

Definition 2. *The class* uniform $\mathrm{VPSPACE}^0$ *is the set of all families (f_n) of multivariate polynomials $f_n \in K[x_1, \ldots, x_{u(n)}]$ satisfying the following requirements:*

1. *the number $u(n)$ of variables is polynomially bounded;*
2. *the polynomials f_n have integer coefficients;*
3. *the size of the coefficients of f_n is bounded by $2^{p(n)}$ for some polynomial p;*
4. *the degree of f_n is bounded by $2^{p(n)}$ for some polynomial p;*
5. *the coefficient function of (f_n) is in* PSPACE.

We have chosen to define first uniform $\mathrm{VPSPACE}^0$, a uniform class without constants, because this is the main object of study in this paper. In keeping with the tradition set by Valiant, however, the class VPSPACE defined in Section 6, is nonuniform and allows for arbitrary constants. In [12] we have also defined a class VΠP, which sits in between VP_{nb} and VPSPACE.

4 The Transfer Theorems

One difficulty springs to mind immediately when one tries to establish transfer theorems between Valiant's model and the Blum-Shub-Smale model: the former model is highly non uniform (there is no uniformity requirement on circuit

families, and arbitrary constants are allowed), whereas the latter model is much more uniform (the availability of arbitrary constants may still be a source of non-uniformity, especially in the real case). I believe that these discrepancies are mostly historical accidents. That is, there is nothing special about evaluation problems that makes them more suitable for non-uniform models, and there is nothing special about decision problems that makes them more suitable for uniform models. One should therefore feel free to modify the Valiant and Blum-Shub-Smale conventions in order to facilitate comparisons. This is just what we do in the following theorem, where we work in the most uniform setting: our circuit families are uniform and constant-free. It is possible to state (and prove!) variations of this theorem for other uniformity conventions.

Theorem 2. *If* uniform $\mathrm{VPSPACE}^0$ = uniform $\mathrm{VP}^0_{\mathrm{nb}}$ *then* $\mathrm{PAR}^0_{\mathbb{R}} = \mathrm{P}^0_{\mathbb{R}}$ *and* $\mathrm{PAR}^0_{\mathbb{C}} = \mathrm{P}^0_{\mathbb{C}}$.

The real case of this theorem is established in [14], and the complex case in [13]. In principle, the definitions of Valiant's classes are relative to a given field K. Note however that we work with constant-free circuits in this theorem, and our two fields $\mathbb{R}$ and $\mathbb{C}$ are both of characteristic 0. There is therefore actually only one class uniform $\mathrm{VP}^0_{\mathrm{nb}}$ to consider, and likewise there is only one class uniform $\mathrm{VPSPACE}^0$. It is therefore really the same hypothesis that implies the two collapses $\mathrm{PAR}^0_{\mathbb{R}} = \mathrm{P}^0_{\mathbb{R}}$ and $\mathrm{PAR}^0_{\mathbb{C}} = \mathrm{P}^0_{\mathbb{C}}$.

Note that for any field K, the collapse of the constant-free class PAR^0_K to P^0_K implies the collapse of PAR_K to P_K: just replace constants by new variables in order to transform a PAR_K problem into a PAR^0_K problem, and then replace these variables by their original values in order to transform a P^0_K problem into a P_K problem. Moreover, for $K = \mathbb{R}$ or $K = \mathbb{C}$ the collapse of PAR_K to P_K implies the collapse of NP_K to P_K since $\mathrm{P}_K \subseteq \mathrm{NP}_K \subseteq \mathrm{PAR}_K$ [2].

It would be interesting to find out whether a converse to Theorem 2 can be established. Results of this type have already been established by Bürgisser [5] and Lickteig [15,16]. For instance, Bürgisser ([5], Corollary 4.5) has shown that if $\mathrm{P}_{\mathbb{R}} = \mathrm{PAR}_{\mathbb{R}}$, VNP families are "easy to approximate".

5 A Proof Sketch

In this section we assume that $K = \mathbb{R}$ or $K = \mathbb{C}$. In both cases, the proof of Theorem 2 builds on the fact that PAR_K problems can be solved by families of branching trees of polynomial depth. This result is established in [10] in the real case, and in [11] in the complex case. At any internal node v of such a tree, one tests the sign that an *arbitrary* polynomial P_v takes on the input $x \in K^n$. As usual, testing the sign means testing whether $P_v(x) = 0$ when $K = \mathbb{C}$, or whether $P_v(x)$ is < 0, > 0 or equal to 0 when $K = \mathbb{R}$. The leaves of the tree are labeled by *Accept* or *Reject*.

The constructions of [10] and [11] exploit the geometric structure of PAR_K problems. For any problem $A \in \mathrm{PAR}_K$, the restriction $A \cap K^n$ of A to n-dimensional inputs is a union of cells of an arrangement of a finite set of hypersurfaces. These hypersurfaces are defined by the polynomials whose sign is tested

at the test gates of the circuit recognizing $A \in K^n$. Their degree is bounded by a singly exponential function of n (because our circuits have polynomial depth), and it turns out that their number is also bounded by a singly exponential function of n. By definition, two points $x, y \in K^n$ are in the same cell if for any such polynomial P occuring in the arrangement, $P(x)$ and $P(y)$ have the the same sign (where the sign is defined as above). Nonempty cells form a partition of the input space K^n. To decide whether an input $x \in K^n$ should be accepted, it is therefore sufficient to locate x in the arrangement, that is, to find to which cell it belongs. It is shown in [10] that point location in an arrangement of real hypersurfaces can be performed by branching trees of depth $O(\log N)$, where N is the number of nonempty cells. The bound established in [11] for the complex case (or in fact for any unordered field) is $O(n \log N)$. It follows from standard bounds in effective algebraic geometry that N is bounded by a singly exponential function of n. We have therefore obtained the desired polynomial bound on the branching complexity of PAR_K problems.

The reader may have noticed a difference between branching trees and the computation trees mentioned in the introduction: in the branching tree model, the cost of polynomial evaluation is completely ignored. This model is therefore rather unrealistic, at least for the purpose of proving upper bounds (lower bounds can be found in [24,27]). To obtain a more realistic model, one should take the cost of evaluating the polynomials P_v into account. Not surprisingly, this is where Valiant's model comes into the picture. Namely, an analysis of the constructions of [10] and [11] shows that the polynomials P_v can be computed by (uniform, constant free) VPSPACE families. In fact, the analysis reveals that theses two constructions can be implemented by polynomial size arithmetic circuits augmented with special "help gates" that can test the sign of VPSPACE families. Theorem 2 follows immediately from this observation.

In the above proof sketch we have tried as much as possible to handle simultaneously the real and complex fields. We now fill in some of the details that are specific to each field, beginning with the branching tree construction for the point location problem.

5.1 The Real Case

Point location also plays an important role in the transfer theorems of [8,9]; these two papers deal with arrangements of hyperplanes rather than hypersurfaces since the underlying computation model is multiplication-free. In fact, these transfer theorems can be viewed as effective versions of the constructions of Meiser [20] and Meyer auf der Heide [21,22]. Meiser's point location algorithm especially looks at first sight like a good candidate for a generalization to hypersurfaces. It relies in particular on an efficient algorithm for dividing convex polyhedra into unions of simplexes. In order to generalize to arrangements of hypersurfaces, why not divide semi-algebraic sets into unions of "simple" semi-algebraic sets? Unfortunately, we do not know how to do that efficiently. It turns out that Grigoriev's solution [10] is very different, and has a more combinatorial character. The key combinatorial lemma, also established in [10], is as follows.

Lemma 1. *Let $u_1, \ldots, u_m$ be pairwise distinct vectors of $(\mathbb{Z}/2\mathbb{Z})^k$. If $m \geq 6$ there exists a vector $v \in (\mathbb{Z}/2\mathbb{Z})^k$ such that*

$$m/3 \leq |\{1 \leq i \leq m;\ \langle v, u_i \rangle = 0\}| \leq 2m/3.$$

In the application to point location, k should be viewed as the number of hypersurfaces in the arrangement, and m as the number of satisfiable sign conditions. By "sign condition", we mean here a system of polynomial inequalities of the form $f_i < 0$ or $f_i > 0$, where the f_i are the polynomials defining the hypersurfaces of the arrangement (conditions of the form $f_i = 0$ are handled in a different way). Lemma 1 is used to perform a kind of binary search among sign conditions (see [10] for details). In order to obtain a constructive version of Grigoriev's result, one needs a constructive version of this lemma. Unfortunately, the original proof is highly nonconstructive since it is a proof of contradiction. A very different proof, based on a probabilistic argument, was given in [7]. Instead of the $[m/3, 2m/3]$ range of Lemma 1, this proof yields the smaller and almost optimal range $[m/2 - \sqrt{m}/2, m/2 + \sqrt{m}/2]$. A probabilistic algorithm (just choose v at random) follows by relaxing slightly these bounds. More importantly for the application to Theorem 2, a deterministic algorithm running in logarithmic space can be obtained by derandomizing the probabilistic algorithm (or more precisely, the probabilistic proof). Note that in the application to Theorem 2, "logarithmic space" should be viewed as "polynomial space" since the the number of hypersurfaces (k) and of satisfiable sign conditions (m) can be exponential in the input dimension n. Our derandomization technique is reminiscent of the proof that bounded-error probabilistic algorithms can be efficiently simulated by boolean circuit families (i.e., BPP $\subseteq$ P/poly). Namely, for each k and m we build in logarithmic space a list of vectors which is guaranteed to contain a suitable vector v for every possible input. Then we find a suitable v in this list by an exhaustive search, which can also be implemented in logarithmic space. Of course, one major difference with the proof that BPP $\subseteq$ P/poly is that making that proof constructive remains a major open problem.

5.2 The Complex Case

The complex case is in a way simpler than the real case because we do not have to distinguish between positive and negative inputs. As a result, we do not need anything like Lemma 1. On the other hand, the complex case looks more difficult because we cannot use as in [10] the standard "sum of squares" trick to decide whether a point belong to an algebraic set.

One important ingredient in the $O(n \log N)$ branching complexity bound of [11] is the well-known fact that any algebraic subset of $\mathbb{C}^n$ (defined by some number k of polynomial equations) can be defined by only $n + 1$ equations (this is actually where the factor "n" in the $O(n \log N)$ bound comes from). The $n + 1$ equations can be obtained by taking "generic" linear combinations of the k original equations. The constructive version of this proof obtained in [13] replaces the generic coefficients in these linear combinations by sufficiently fast

growing sequences of integers. This is by no means a new method in algebraic complexity, but one must perhaps work harder than usual to keep the size of these integers under control (see [13] for details).

6 On the Hypothesis that VPSPACE families have small circuits

There are actually several version of this hypothesis, depending on uniformity conditions and the role of constants. As pointed in section 4 the hypothesis uniform $\mathrm{VPSPACE}^0$ = uniform $\mathrm{VP}^0_{\mathrm{nb}}$ in Theorem 2 is the most uniform, hence the strongest. We define below two increasingly nonuniform classes, $\mathrm{VPSPACE}^0$ and VPSPACE. The only difference between $\mathrm{VPSPACE}^0$ and uniform $\mathrm{VPSPACE}^0$ is the nonuniformity of the coefficient function; VPSPACE is even more nonuniform since arbitrary constants are allowed.

Definition 3. *The class* $\mathrm{VPSPACE}^0$ *is the set of all families* (f_n) *of multivariate polynomials* $f_n \in K[x_1, \ldots, x_{u(n)}]$ *satisfying the following requirements:*

1. *the number* $u(n)$ *of variables is polynomially bounded;*
2. *the polynomials* f_n *have integer coefficients;*
3. *the size of the coefficients of* f_n *is bounded by* $2^{p(n)}$ *for some polynomial* p*;*
4. *the degree of* f_n *is bounded by* $2^{p(n)}$ *for some polynomial* p*;*
5. *the coefficient function of* (f_n) *is in* PSPACE/poly.

Now, the class VPSPACE *is the set of all families* $(f_n(\bar{x}))$ *of multivariate polynomials* $f_n \in K[x_1, \ldots, x_{u(n)}]$ *such that there exist a family* $(g_n(\bar{x}, \bar{y}))$ *in* $\mathrm{VPSPACE}^0$ *together with a family of tuples of constants* $(\bar{a}^{(n)})$ *satisfying for all* n*:*

$$f_n(\bar{x}) = g_n(\bar{x}, \bar{a}^{(n)}).$$

One could consider up to 6 different versions of the class VPSPACE since there are 2 possible choices for the coefficient function (uniform or nonuniform) and 3 possible choices for constants (no constants, arbitrary constants as in Valiant's original definitions, or the intermediate Blum-Shub-Smale convention in which the same tuple of constants is used for every input size). Some of these classes are in fact equal since nonuniformity comes for free when arbitrary constants are available. We have shown in [14] that the weakest hypothesis, namely, $\mathrm{VP}_{\mathrm{nb}} = \mathrm{VPSPACE}$, already has strong consequences:

Proposition 1. *Under the generalized Riemann hypothesis (GRH),*

$$\mathrm{VP}_{\mathrm{nb}} = \mathrm{VPSPACE} \iff [\mathrm{P/poly} = \mathrm{PSPACE/poly} \textit{ and } \mathrm{VP} = \mathrm{VNP}].$$

Moreover, the implication from right to left holds even without GRH.

One advantage of working with these nonuniform classes is therefore that there is after all no need to introduce the new class VPSPACE: since we have an equivalence in Proposition 1, the hypothesis can be formulated using only the familiar classes P/poly, PSPACE/poly, VP and VNP. The hypothesis uniform $\mathrm{VPSPACE}^0$ = uniform $\mathrm{VP}^0_{\mathrm{nb}}$ has an even more dire consequence:

Proposition 2.

$$\text{uniform VPSPACE}^0 = \text{uniform VP}^0_{\text{nb}} \Longrightarrow \text{PSPACE} = \text{P-uniform NC}.$$

The separation "PSPACE $\neq$ P-uniform NC" is extremely plausible, but to the best of my knowledge remains a conjecture (by contrast, PSPACE can be separated from logspace-uniform NC thanks to the space hierarchy theorem).

References

1. E. Allender. Arithmetic circuits and counting complexity classes. In J. Krajicek, editor, *Complexity of Computations and Proofs. Quaderni di Matematica Vol. 13, Seconda Universita di Napoli*, pages 2–15, 2004.
2. L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation*. Springer-Verlag, 1998.
3. 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.
4. P. Bürgisser. *Completeness and Reduction in Algebraic Complexity Theory*. Number 7 in Algorithms and Computation in Mathematics. Springer, 2000.
5. P. Bürgisser. The complexity of factors of multivariate polynomials. *Foundations of Computational Mathematics*, 4(4):369–396, 2004.
6. O. Chapuis and P. Koiran. Saturation and stability in the theory of computation over the reals. *Annals of Pure and Applied Logic*, 99:1–49, 1999.
7. P. Charbit, E. Jeandel, P. Koiran, S. Périfel, and S. Thomassé. Finding a vector orthogonal to roughly half a collection of vectors. To appear in *Journal of Complexity*, 2007.
8. H. Fournier and P. Koiran. Are lower bounds easier over the reals ? In *Proc. 30th ACM Symposium on Theory of Computing*, pages 507–513, 1998.
9. H. Fournier and P. Koiran. Lower bounds are not easier over the reals: Inside PH. In *Proc. 27th International Colloquium on Automata, Languages and Programming*, volume 1853 of *Lecture Notes in Computer Science*, pages 832–843. Springer, 2000.
10. D. Grigoriev. Topological complexity of the range searching. *Journal of Complexity*, 16:50–53, 2000.
11. P. Koiran. Circuits versus trees in algebraic complexity. In *Proc. STACS 2000*, volume 1770 of *Lecture Notes in Computer Science*, pages 35–52. Springer-Verlag, 2000.
12. P. Koiran and S. Périfel. Valiant's model: from exponential sums to exponential products. In *Mathematical Foundations of Computer Science*, volume 4162 of *Lecture Notes in Computer Science*, pages 596–607. Springer-Verlag, 2006.
13. P. Koiran and S. Périfel. VPSACE and a transfer theorem over the complex field. available from http://perso.ens-lyon.fr/pascal.koiran, 2007.
14. P. Koiran and S. Périfel. VPSACE and a transfer theorem over the reals. In *Proc. STACS 2007*, volume 4393 of *Lecture Notes in Computer Science*, pages 417–428. Springer-Verlag, 2007. long version: http://prunel.ccsd.cnrs.fr/ensl-00103018.
15. T. Lickteig. Testing polynomials for zero. Internal report, Universität Tübingen, 1988.
16. T. Lickteig. On semialgebraic decision complexity. Technical Report TR-90-52, International Computer Science Institute, Berkeley, 1990. Habilitationsschrift, Universität Tübingen.

17. G. Malod. *Polynômes et coefficients*. PhD thesis, Université Claude Bernard - Lyon 1, 2003.
18. G. Malod. The complexity of polynomials and their coefficient functions. In *Proc. 22nd IEEE Conference on Computational Cmplexity*, 2007.
19. K. Meer. A note on a $P \neq NP$ result for a restricted class of real machines. *Journal of Complexity*, 8:451–453, 1992.
20. S. Meiser. Point location in arrangements of hyperplanes. *Information and Computation*, 106(2):286–303, 1993.
21. 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.
22. F. Meyer auf der Heide. Fast algorithms for n-dimensional restrictions of hard problems. *Journal of the ACM*, 35(3):740–747, 1988.
23. B. Poizat. *Les Petits Cailloux*. Nur Al-Mantiq Wal-Ma'rifah **3**. Aléas, Lyon, 1995.
24. S. Smale. On the topology of algorithms. I. *Journal of Complexity*, 3:81–89, 1987.
25. L. G. Valiant. Completeness classes in algebra. In *Proc. 11th ACM Symposium on Theory of Computing*, pages 249–261, 1979.
26. L. G. Valiant. Reducibility by algebraic projections. In *Logic and Algorithmic (an International Symposium held in honour of Ernst Specker)*, pages 365–380. Monographie n^o 30 de L'Enseignement Mathématique, 1982.
27. V. A. Vassiliev. On decision trees for orthants. *Information Processing Letters*, 62(5):265–268, 1997.
28. J. von zur Gathen. Parallel linear algebra. In J. Reif, editor, *Synthesis of Parallel Algorithms*, pages 573–617. Morgan Kaufmann, 1993.