Ultimate Polynomial Time

Gregorio Malajovich*

January 21, 1999

Abstract

The class $\mathcal{UP}$ of 'ultimate polynomial time' problems over $\mathbb{C}$ is introduced; it contains the class $\mathcal{P}$ of polynomial time problems over $\mathbb{C}$.

The τ-Conjecture for polynomials implies that $\mathcal{UP}$ does not contain the class of non-deterministic polynomial time problems definable without constants over $\mathbb{C}$. This latest statement implies that $\mathcal{P} \neq \mathcal{NP}$ over $\mathbb{C}$.

A notion of 'ultimate complexity' of a problem is suggested. It provides lower bounds for the complexity of structured problems.

1 Introduction

A model of Computation and Complexity over a ring was developed in [2] and [1], generalizing the classical $\mathcal{NP}$-completeness theory [3]. Of particular interest is the model of Complexity over the ring $\mathbb{C}$ of complex numbers.

In the model of complexity over $\mathbb{C}$, a machine is allowed to input, to output and to store complex numbers, to compute polynomials and to branch on equality (See the textbook [1] for background). This model shares some of the features of the classical (Turing) model of computation (There is a discussion in [7]). It is known [5, 9] that the hypothesis $\mathcal{BPP} \not\supseteq \mathcal{NP}$ in the Turing setting implies $\mathcal{P} \neq \mathcal{NP}$ over $\mathbb{C}$. ($\mathcal{BPP}$ stands for Bounded

*Departamento de Matemática Aplicada, Universidade Federal do Rio de Janeiro. Caixa Postal 68530, CEP 21945, Rio de Janeiro, RJ, Brasil. e-mail: gregorio@labma.ufrj.br. On leave at MSRI, 1000 Centennial Drive, Berkeley CA 94720-5070. e-mail: gregorio@msri.org

Probability Polynomial Time. If $\mathcal{BPP}$ would happen to contain $\mathcal{NP}$, then there would be polynomial time randomized algorithms for such tasks as factorizing large integers or breaking most modern cryptographic systems).

In [8, 1, 10], the hypothesis $\mathcal{P} \neq \mathcal{NP}$ over the Complex numbers was related to a number-theoretical conjecture. Define a straight-line program as a list

$$s_0 = 1 \ , \ s_1 = x \ , \ s_2 \ , \ \cdots \ , \ s_\tau$$

where s_i is, for $i \geq 2$, either $s_j + s_k$, $s_j - s_k$ or $s_j s_k$, for some $j, k < i$. Each s_i is thus a polynomial in x. The straight-line program is said to *compute* the polynomial $s_\tau(x)$.

Given a polynomial $f \in \mathbb{Z}[x]$, the quantity $\tau(f)$ is defined as the smallest τ such that there exists a straight-line program $s_0, \cdots, s_\tau$ computing $f(x)$. For instance, $\tau(x^{2^n} - 1) = 2 + n$. Similarly, if $g \in \mathbb{Z}[x_1, \cdots, x_n]$, then $\tau(g)$ is the minimal length of a straight-line program $s_0 = 1, s_1 = x_1, \cdots, s_n = x_n, s_{n+1}, \cdots, s_\tau = g(x)$.

The τ Conjecture for Polynomials . *There is a constant $a > 0$ such that for any univariate polynomial $f \in \mathbb{Z}[x]$,*

$$n(f) < \tau(f)^a$$

where $n(f)$ is the number of integer zeros of f, without multiplicity.

It is known [1] that the τ-Conjecture for polynomials implies $\mathcal{P} \neq \mathcal{NP}$ over $\mathbb{C}$. A main step towards this result is the fact that, if the τ-Conjecture is true, then the polynomials

$$p_d(x) = (x-1)(x-2)\cdots(x-d)$$

are *ultimately hard to compute.* This means that there cannot be constants a and b such that, for any degree d, for some non-zero polynomial f (depending on d), we would have

$$\tau\ (\ p_d(x) f(x)\ ) < a\, (\log_2 d)^b$$

Therefore, all non-zero multiples of p_d are hard to compute, hence the wording ultimately hard.

The goal of this paper is to define a new complexity class $\mathcal{UP}$, of *ultimate polynomial time* problems. This class will contain $\mathcal{P} \cap \mathcal{K}$, where $\mathcal{P}$ is the class of problems decidable in polynomial time and $\mathcal{K}$ is the class of problems definable without constants (See [4] and Definition 1 below). Moreover:

Theorem 1. *The implications (a) $\Rightarrow$ (b) $\Rightarrow$ (c) $\Rightarrow$ (d) are true:*

(a) The τ-conjecture for polynomials.

(b) $\forall d$, p_d is ultimately hard to compute.

(c) $\mathcal{UP} \not\supseteq \mathcal{NP} \cap \mathcal{K}$ over $\mathbb{C}$.

(d) $\mathcal{P} \neq \mathcal{NP}$ over $\mathbb{C}$.

The implication (a) $\Rightarrow$ (b) $\Rightarrow$ (d) appears in [1], the hypothesis (c) in-between is new. It is at least as likely as the τ-conjecture, while still implying $\mathcal{P} \neq \mathcal{NP}$.

We will also show a $\mathcal{NP}$-hardness result for the class $\mathcal{UP}$: there is a structured problem $(HN, HN^{\text{yes}}) \in \mathcal{NP} \cap \mathcal{K}$, such that:

Theorem 2. $\mathcal{UP} \not\supseteq \mathcal{NP} \cap \mathcal{K}$ *over $\mathbb{C}$ if and only if $(HN, HN^{\text{yes}}) \notin \mathcal{UP}$ over $\mathbb{C}$.*

The problem (HN, HN^{yes}) is precisely the (structured) Hilbert Nullstellensatz, known to be $\mathcal{NP}$-complete over $\mathbb{C}$ ([1]).

This paper was written while the author was visiting Mathematical Sciences Research Institute in Berkeley. The author also wishes to thank Pascal Koiran and Steve Smale for their comments and suggestions.

2 Background and Notations

Recall from [1] that $\mathbb{C}^\infty$ is the disjoint union

$$\mathbb{C}^\infty = \bigsqcup_{i=0,1,\cdots} \mathbb{C}^i$$

This means that there is a well-defined *size* function,

$$\begin{array}{rccl} \text{Size}: & \mathbb{C}^\infty & \to & \mathbb{N} \\ & x & \mapsto & \text{Size}(x) = i \text{ such that } x \in \mathbb{C}^i \end{array}$$

A decision problem X is a subset of $\mathbb{C}^\infty$. It is in the class $\mathcal{P}$ if and only if there is a machine M over $\mathbb{C}$, that terminates for any input x in time bounded by a polynomial on $\mathrm{Size}(x)$, and such that

$$M(x) = 0 \;\Leftrightarrow\; x \in X$$

where $M(x)$ is the result of running M with input x. Without loss of generality we may assume that $M(x) \in \{0; 1\}$.

Under some circumstances, it is possible to assume that the machine M above has only coefficients 0 or 1 (This is called a *constant-free* machine). However, one may have to replace problem X over $\mathbb{C}$ by problem $X \cap \mathbb{Z}$ over $\mathbb{Z}$, with unit cost. (This is the contents of Propositions 3 and 9 of Chapter 7 of [1]). In order to avoid this technical complication and keep the same problem over $\mathbb{C}$, we will follow another approach to Elimination of Constants.

This approach was introduced by Koiran in [4]. The idea is to consider only machines for a subclass of problems. This subclass will contain most of the interesting examples, while precluding pathological cases such as $X = \{\pi\}$.

Definition 1 (Koiran). A problem L is said to be *definable without constants* if for each input size n there is a formula F_n in the first order theory of $\mathbb{C}$ such that 0 and 1 are the only constants occurring in F_n, and for any $x \in \mathbb{C}^n$, $x \in L$ if and only if $F_n(x)$ is true (there is no restriction on the size of F_n.

For future reference, we quote below Theorem 2 of [4]. The original statements of both Definition 1 and Theorem 3 are actually more general (for any algebraically closed field of characteristic 0).

Theorem 3 (Koiran). *Let $L \subseteq K^\infty$ be a problem which is definable without constants. If $L \in \mathcal{P}$, L can be recognized in polynomial time by a constant-free machine.*

The class of all the problems definable without constants will be denoted by $\mathcal{K}$.

We will need crucially in the sequel the notion of a *structured problem.* A structured problem is a pair (X, X^{yes}), $X^{\mathrm{yes}} \subseteq X \subseteq \mathbb{C}^\infty$. A non-structured problem X can always be written as the structured problem $(\mathbb{C}^\infty, X)$. The class $\mathcal{UP}$ will be meaningful only as a class of structured problems. But first of all, recall that

Definition 2. A structured problem (X, X^{yes}) belongs to the class $\mathcal{P}$ if and only if $X \in \mathcal{P}$ and $X^{\text{yes}} \in \mathcal{P}$.

Definition 3. A structured problem (X, X^{yes}) belongs to the class $\mathcal{K}$ if and only if $X \in \mathcal{K}$ and $X^{\text{yes}} \in \mathcal{K}$.

Definition 4. A structured problem (X, X^{yes}) belongs to the class $\mathcal{NP}$ if and only if:

(1) The problem X belongs to the class $\mathcal{P}$.

(2) There is a machine M with input x, g such that

$$x \in X \text{ and } \exists g \in \mathbb{C}^{\infty} \text{ s.t. } M(x,g) = 0 \Leftrightarrow x \in X^{\text{yes}}$$

(3) Furthermore, there is a polynomial p such that, for all $x \in X^{\text{yes}}$, there is $g \in \mathbb{C}^{\infty}$ such that $M(x,g) = 0$ and the running time of M with input x, g is no more than $p(\text{Size}(x))$.

Example 1. Let HN be the class of all lists $(m, n, f_1, \cdots, f_m)$ where $f_1, \cdots, f_m$ are polynomials in n variables. Each polynomial $f = \sum f_I x^I$ is represented sparsely by a list of monomials $(S, m_1, \cdots, m_S)$, where each monomial is a list $(f_I, I_1, \cdots, I_n)$.

An important convention to have in mind: integers appearing in the definition of a problem should be represented in bit representation. In this case, m, n, S, I_j are all lists of zeros and ones. Complex values are represented by one complex number. With this convention, HN is clearly in the class $\mathcal{P}$.

We also define HN^{yes} as the subset of polynomial systems in HN that have a common root over $\mathbb{C}$.

The definition above of the structured problem (HN, HN^{yes}) can be translated into first order constant-free formulae over $\mathbb{C}$. Therefore, $(HN, HN^{\text{yes}}) \in \mathcal{K}$. It is also $\mathcal{NP}$-complete over the complex numbers (Theorem 1 in Chapter 5 of [1]).

Example 2. Let

$$\begin{aligned} X &= \{(m, x) \in \mathbb{N} \times \mathbb{C}\} \\ X^{\text{yes}} &= \{(m, x) \in X \text{ such that } x \in \{1, 2, \cdots, m\}\} \end{aligned}$$

with the convention that m is in bit representation, while x is a complex number. Hence, $\text{Size}((m,x)) = O(1 + \lceil \log_2(m) \rceil)$. Then the problem (X, X^{yes}) is in $\mathcal{NP}$ over $\mathbb{C}$. The machine $M(x,g)$ can be constructing by guessing the bit decomposition g_i of x, and computing $x - \sum g_i 2^i$.

Again, (X, X^{yes}) is definable without constants.

3 Construction of the class $\mathcal{UP}$

In Chapter 7 of [1], it is proved that if the problem (X, X^{yes}) from Example 2 would happen to belong to the class $\mathcal{P}$, then condition (b) in Theorem 1 would be false. Therefore (b) implies $\mathcal{P} \neq \mathcal{NP}$ over $\mathbb{C}$.

The class $\mathcal{UP}$ will be constructed by abstracting the same reasoning. The construction relies on some geometric properties of structured problems in $\mathcal{P}$. The notation that follows will be used in the sequel:

Let (X, X^{yes}) be a structured problem with $X \in \mathcal{P}$. We denote by $X \cap \mathbb{C}^i$ the set $\{x \in X : \text{Size}(x) = i\}$ of size i instances of the problem. Then we write $\overline{X \cap \mathbb{C}^i}$ for its Zariski closure over $\mathbb{C}$. We can define a new object associated to X as:

$$\overline{X} = \bigsqcup_{i=0,1,\cdots} \overline{X \cap \mathbb{C}^i}$$

We can think of $\overline{X}$ as the *closure* of X, indeed it is the smallest 'closed' problem containing X. Remark that in Examples 1 and 2, we have respectively $X = \overline{X}$ and $HN = \overline{HN}$.

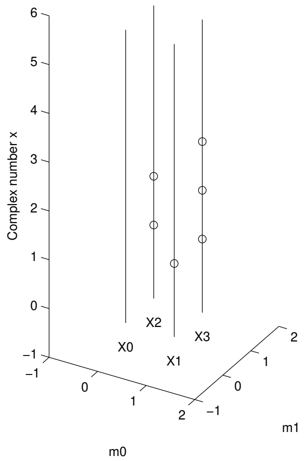
We can also decompose each Zariski-closed set $\overline{X \cap \mathbb{C}^i}$ into a finite union of irreducible components (affine varieties). Thus it makes sense to write $\overline{X}$ as the countable union:

$$\overline{X} = \bigcup X_j$$

where each X_j is an affine variety lying in some $\mathbb{C}^s$, where $s = \text{Size}(x), x \in X_j$. We can further define:

$$\begin{aligned} X_j^{\text{yes}} &= X_j \cap X^{\text{yes}} \\ X_j^{\text{no}} &= X_j \setminus X^{\text{yes}} \end{aligned}$$

(See Figure 1). Using this notation,



This is Problem (X, X^{yes}) from Example 2, restricted to the inputs (m_0, m_1, x) of size 3. X is represented by the four (complex !) lines and X^{yes} by the dots. Each of the complex lines is irreducible, and hence corresponds to a different X_i.

Figure 1: (X, X^{yes}) from Example 2

Definition 5. The class $\mathcal{UP}$ is the class of all structured problems (X, X^{yes}) such that $X \in \mathcal{P}$ and for all X_i, there is a non-zero polynomial $f_i \in \mathbb{Z}[x_1, \cdots, x_{s_i}]$, where $s_i = \text{Size}(x)$ for $x \in X_i$, with the following properties:

(1) $\tau(f_i)$ is polynomially bounded in S_i.

(2) $X_i^{\text{yes}} \subseteq Z(f)$ or $X_i^{\text{no}} \subseteq Z(f)$

Proposition 1. $\mathcal{P} \cap \mathcal{K} \subseteq \mathcal{UP}$

Proof of Proposition 1. Let (X, X^{yes}) be in $\mathcal{P} \cap \mathcal{K}$. Let $M = M(x)$ be the machine that recognizes $x \in X^{\text{yes}}$ in polynomial time, where the input x is assumed to be in $\overline{X}$. Although it is possible that an $x \in X_i$ is not in X, it is still possible to recognize $x \in X^{\text{yes}}$ in polynomial time. Indeed, X is also in $\mathcal{P}$. The machine $M(x)$ will check $x \in X$ and $x \in X^{\text{yes}}$.

Now we apply elimination of constants (Theorem 3), and choose M to be constant-free.

The nodes of the machine M are supposed to be numbered. Given an input x, the *path* followed by input x is the list of nodes traversed during the computation of $M(x)$.

When the input is restricted to one of the affine varieties X_i's, we can define the canonical path (associated to X_i as the path followed by the generic point of X_i. This corresponds to the following procedure:

At each decision node, at time T, branch depends upon an equality $F^T(x) = 0$, where x is the original input. The polynomial F can be computed within the machine running time. In case $F^T(x) = 0$ for all $x \in X_i$, we follow the Yes-path and say that this branching is trivial.

If not, we follow the no-path and say that this branching is non-trivial. The fact that X_i is a variety is essential here, since it guarantees that only a codimension ≥ 1 subset of inputs may eventually follow the Yes-path at this time.

The set of inputs that do NOT follow the canonical path can be described as the zero-set of

$$f_i = \prod F^T$$

where the product ranges over the non-trivial branches only. The polynomial f_i can be computed in at most twice the running time of the machine M restricted to X_i. By hypothesis, this is polynomial time in the size of $x \in X_i$.

Since we assumed that M returns only 0 or 1, the set of the inputs that follow the canonical path (i.e. $Z(f_i)$) is either all in X_i^{yes} or all in its complementary X_i^{no}.

There are now two possibilities. First possibility, X_i^{yes} has measure zero in X_i, and therefore it must be contained in $Z(f_i)$. Second possibility, X_i^{yes} has non-zero measure, hence it contains the complementary of $Z(f_i)$, and hence X_i^{no} is a subset of $Z(f_i)$. $\square$

4 Proof of the Theorems

Proof of Theorem 1.

(a) $\Rightarrow$ (b) is trivial, refer to [1] Chapter 7.

(b) $\Rightarrow$ (c): Let (X, X^{yes}) be the problem in Example 2. Since X_i^{no} is generic in X_i, all inputs in X_i^{yes} should escape the canonical path. Hence, if f_d is the polynomial that defines the canonical path, $f_d(i) = 0$ for $i = 1, 2, \cdots, d$. But then it cannot be evaluated in time polylog(d), by hypothesis (b). Hence, under the assumption (b), the problem (X, X^{yes}) is not in $\mathcal{UP}$. It does belong to $\mathcal{NP} \cap \mathcal{K}$, so $\mathcal{UP} \not\supseteq \mathcal{NP} \cap \mathcal{K}$.

(c) $\Rightarrow$ (d) : Using Theorem 2, Condition (c) implies that $(HN, HN^{\text{yes}}) \notin \mathcal{UP}$. However, since $(HN, HN^{\text{yes}}) \in \mathcal{K}$, Proposition 1 implies $(HN, HN^{\text{yes}}) \notin \mathcal{P}$. Hence $\mathcal{P} \neq \mathcal{NP}$ over $\mathbb{C}$. $\square$

Proof of Theorem 2. Let $(X, X^{\text{yes}}) \in \mathcal{NP} \cap \mathcal{K}$ and assume that $(HN, HN^{\text{yes}}) \in \mathcal{UP}$. We have to show that $(X, X^{\text{yes}}) \in \mathcal{UP}$.

For each X_i, one can embed (X_i, X_i^{yes}) into some $(HN_i, HN_i^{\text{yes}})$ as follows:

Let $M = M(x)$ be the deterministic polynomial time machine to recognize X, and let $N = N(x, g)$ be the non-deterministic polynomial time machine to recognize X^{yes}. We can assume without loss of generality that M and N are constant-free (Theorem 3).

Let T be the maximum running time of M and N when the input is restricted to X_i. Let $\phi(x)$ be the combined Register Equations of machines M and N for time T (Theorem 2 in Chapter 3 of [1]). Thus, $\phi(x)$ is a system of polynomial equations with integer coefficients and indeterminate coefficients $x_1, x_2, \cdots$. The polynomial system $\phi(x)$ can be constructed in polynomial time from x, and the size of $\phi(x)$ is polynomially bounded by the size of x.

We claim that $\phi(X_i)$ is contained in some HN_j, and that in that case $\phi(X_i^{\text{yes}}) \subseteq HN_j^{\text{yes}}$ and $\phi(X_i^{\text{no}}) \subseteq HN_j^{\text{no}}$.

Indeed, $X_i \subseteq \mathbb{C}^s$ for some s, and $\phi(\mathbb{C}^s) \subseteq HN_j$ for some j. Then $x \in X_i$ belongs to X^{yes} if and only if the corresponding $\phi(x)$ has a solution over $\mathbb{C}$.

We now distinguish two cases:

Case 1: HN_j^{yes} has measure zero in HN_j. Thus $HN_j^{\text{yes}} \subseteq Z(\hat{f}_j)$ for an easy-to-compute polynomial $\hat{f}_j$. In that case, since X_i^{yes} gets mapped into HN_j^{yes}, the composition $f_i = \hat{f}_j \circ \phi$ gives the polynomial associated to X_i.

Case 2: HN_j^{no} has measure zero in HN_j. Thus $HN_j^{\text{no}} \subseteq Z(\hat{f}_j)$ for an easy-to-compute polynomial $\hat{f}_j$. In that case, since X_i^{no} gets mapped into HN_j^{no}, $f_i = \hat{f}_j \circ \phi$ is the polynomial associated to X_i. $\square$

5 Ultimate Complexity

Let (Y, Y^{yes}) be a problem over $\mathbb{C}$, definable without constants and with Y semi-decidable (i.e. Y is the halting set of some machine). The closure $\overline{Y}$ is well-defined and can be written as a countable union of irreducible varieties Y_i.

For any machine M to solve (Y, Y^{yes}), one can produce a family of polynomials f_i, vanishing on the set of inputs that follow the canonical-path of M restricted to Y_i. As in item (2) of Definition 5, we have

$$Y_i^{\text{yes}} \subseteq Z(f_i) \text{ or } Y_i^{\text{no}} \subseteq Z(f_i)$$

Also, for each input size s, one has a finite number of indices i corresponding to components i $Y_i \subseteq \overline{Y}$ of size-s input. We can thus maximize over those indices i:

$$u_M(s) = \max_{i:Y_i \subseteq \mathbb{C}^i} \tau(f_i)$$

This invariant may be called 'ultimate running time', and is a lower bound (up to a constant) for the worst-case running time of M. As with ordinary complexity theory, one can define the 'ultimate complexity' class of a problem as the class of functions $u : \mathbb{N} \to \mathbb{R}$ such that $\exists M, c > 0 : \forall x u_M(x) \leq cu(x)$ and M recognizes (Y, Y^{yes}). This provides notions such as 'ultimate logarithmic time' or 'ultimate exponential time'.

In [6], a similar construction is used to obtain lower bounds for some specific decision problems. Those problems, however, had a very simple geometric structure (for each 'input size', X^{yes} was a finite set in $\mathbb{C}$). The motivation of this paper was to extend some of the ideas therein and in Chapter 7 of [1] to non-codimension-1 problems.

References

[1] Lenore Blum, Felipe Cucker, Mike Shub, and Steve Smale. *Complexity and Real Computation.* Springer, 1998.

[2] Lenore Blum, Mike Shub, and Steve Smale. On a theory of computation and complexity over the real numbers: $\mathcal{NP}$-completeness, recursive functions and universal machines. *Bulletin of the AMS*, 21(1), 1989.

[3] Michael R. Garey and David S. Johnson. *Computers and intractability.* W. H. Freeman and Co., San Francisco, Calif., 1979. A guide to the theory of NP-completeness, A Series of Books in the Mathematical Sciences.

[4] Pascal Koiran. Elimination of constants from machines over algebraically closed fields. *J. Complexity*, 13(1):65–82, 1997.

[5] Pascal Koiran. A weak version of the Blum, Shub, and Smale model. *Journal of Computer and System Sciences*, 54(1, part 2):177–189, 1997. 1st Annual Dagstuhl Seminar on Neural Computing (1994).

[6] Gregorio Malajovich. Lower bounds for some decision problems over $\mathbb{C}$. Preprint, MSRI, 1999.

[7] Gregorio Malajovich and Klaus Meer. On the structure of $\mathcal{NP}_{\mathbb{C}}$. *SIAM Journal on Computing*, 28(1), 1999.

[8] Mike Shub and Steve Smale. On the intractability of Hilbert's Nullstellensatz and an algebraic version of $\mathcal{P} = \mathcal{NP}$. *Duke Mathematical J.*, 81:47–54, 1995.

[9] Steve Smale. Manifestations of computational complexity. Lecture given at the Introductory Workshop on Foundations of Computational Mathematics and Symbolic Computation in Geometry and Analysis, MSRI, August 1998. Lecture on video at www.msri.org.

[10] Steve Smale. Mathematical problems for the next century. *Math. Intelligencer*, 20(2):7–15, 1998.