The symmetric subset-sum problem over the complex numbers

Mihai Prunescu

Universität Freiburg, Abteilung Mathematische Logik, Eckerstr. 1, D-78103 Freiburg im Breisgau, Deutschland.

Abstract

A problem naturally arizing in the unit-cost complexity class NP over the field $\mathbb{C}$ of complex numbers consists in deciding if an input of length $2n$ belongs to a special absolutely irreducible hypersurface of the affine space $\mathbb{C}^{2n}$. Consequently, the decision problem is substituted by a computation problem.

1 Introduction

The P vs. NP problem put in the Blum - Shub - Smale (B. S. S.) computation model over algebraic structures deals with one of the oldest issues in algorithmic algebra and logic, the efficiency of quantifier elimination methods. This has been also a constant interest of Volker Weispfenning.

One of the most intriguing open questions in the area is the P vs. NP problem for the field $\mathbb{C}$ of the complex numbers. Our goal is to present a possible approach to this problem. This approach focusses on a family of irreducible polynomials, which will be called subset-sum polynomials. Their definition is related with problems from the classical Theory of Complexity (see [3] and [1]) like Knapsack and Subset-Sum. The author doesn't know if these polynomials have been intensively studied so far. This paper can be understood as a manifest for doing so.

In the B. S. S. computation model (see [1] and [7]) deterministic machines working over algebraic structures proceed signature operations and verify signature relations in units of time. Nondeterminism arises in two different forms. One of them is the boolean nondeterminism, produced by states of random branching in the computation path. This is equivalent with guessing in a set with two elements. The other is the existential nondeterminism, by guessing elements in the whole structure. A problem over a structure S is a set of strings of elements of S decided (non-deterministically recognized) by a machine. For a structure S we denote by P(S) the class of problems which can be deterministically decided in polynomial time in the number of elements of the string. This complexity measure is called unit-cost. NBP(S) is the class of problems which are recognized by branching non-deterministic machines in polynomial time according to the unit-cost. NP(S) is the class of problems which are recognized in polynomial time by existential non-deterministic machines. It always holds P(S) $\subseteq$ NBP(S) $\subseteq$ NP(S).

For results concerning P $\neq$ NP over some others algebraic structures see [2], [4], [5], [8], [9], [10], [11].

A deterministic machine working in polynomial time $p(n)$ can be seen as a recursive sequence of circuits (C_n), such that every C_n has at most $p(n)$ gates. P($\mathbb{C}$) = NP($\mathbb{C}$) if and only if there is an algorithm that can find in polynomial time for every existential formula with free variables an equivalent decision circuit. If such an algorithm exists then the number of gates of the corresponding circuit shall be bounded by a polynomial in the length of the given existential formula. In this sense, P($\mathbb{C}$) = NP ($\mathbb{C}$) means the existence of a procedure of polynomial-time quantifier elimination from formulas with an existential quantifier block to equivalent decision circuits. This question is open, but like for the classical P vs. NP problem the answer is suposed to be negative.

In the monograph [1] the unit-cost problem Knapsack is introduced as a possible candidate for a problem in NBP ($\mathbb{C}$) but not in P ($\mathbb{C}$). The problems discussed here are related with Knapsack, but are given as sequences of varieties defined by absolutely irreducible polynomials. The absolute irreducibility permits us to replace the decision circuits by computation circuits, without equality tests: if $\mathbb{C}$ had P = NP with unit-cost, then multiples of the subset-sum polynomials are computable by short straight-line programs. In the last section we complete this heuristic by considering the corresponding bit-problems.

2 Knapsack and subset-sum problems

The elementary symmetric polynomial $\sigma_k(x_1,\dots,x_n)$ is defined as:

$$\sigma_k(x_1,\dots,x_n) := \sum_{|J|=k} \prod_{j\in J} x_j.$$

The function $\vec{\sigma}(\vec{x}) := (\sigma_1(\vec{x}),\dots,\sigma_n(\vec{x}))$ is computable in quadratic time with respect to the unit-cost by iterating the rule:

$$\sigma_{k,n+1} = x_{n+1}\sigma_{k-1,n} + \sigma_{k,n}.$$

Definition 1 Knapsack Kn, Subset-sum SS, Subset k-Sum SS$_k$ (for k fixed), Symmetric Subset-sum SSS, and the special problems SS $(n,2n)$ and SSS $(n,2n)$ are defined as follows:

Kn := $\{(x_1,\dots,x_n,b) \,|\, n\in\mathbb{N},\ \exists\ \varepsilon_1,\dots\varepsilon_n \in \{0,1\},\ \#\{\varepsilon_i = 1\} > 0$ and $b = \sum \varepsilon_i x_i\}$.

SS := $\{(x_1,\dots,x_n) \,|\, n\in\mathbb{N},\ \exists\ \varepsilon_1,\dots\varepsilon_n \in \{0,1\},\ \#\{\varepsilon_i = 1\} > 0$ and $\sum \varepsilon_i x_i = 0\}$.

SS$_k$:= $\{(x_1,\dots,x_n) \,|\, n \geq k,\ \exists\varepsilon_1,\dots\varepsilon_n \in \{0,1\},\ \#\{\varepsilon_i = 1\} = k$ and $\sum \varepsilon_i x_i = 0\}$.

SS $(n,2n)$:= $\{(x_1,\dots,x_{2n}) \,|\, n\in\mathbb{N}$ and $\vec{x} \in$ SS$_n\}$.

SSS := $\{(\sigma_1,\dots,\sigma_n) \,|\, \exists\ \vec{x} \in$ SS with $\vec{\sigma} = \vec{\sigma}(\vec{x})\}$.

SSS $(n,2n)$:= $\{(\sigma_1,\dots,\sigma_{2n}) \,|\, \exists\ \vec{x} \in$ SS $(n,2n)$ with $\vec{\sigma} = \vec{\sigma}(\vec{x})\}$.

We see that SS_k is in P($\mathbb{C}$) and that Kn like SS and SS $(n,2n)$ are in NBP($\mathbb{C}$). The fundamebtal symmetric polynomials are computable in polynomial time, so SSS and SSS $(n,2n)$ are in NP($\mathbb{C}$).

There is the following connection between Kn and SS:

$$(\vec{x},b) \in \mathrm{Kn} \Leftrightarrow \exists\, k \in \{1,\dots,n\}\ \ (kx_1 - b,\dots,kx_n - b) \in \mathrm{SS}_k.$$

So there is a polynomial time decision procedure for Kn finding also the cardinality of all solutions if and only if there is a uniform decision algorithm for the problems SS_k $(k = 1,\dots,n)$ in a uniform polynomial time depending only of n.

3 Subset-sum polynomials

The subset-sum polynomials $X_{k,n}(x_1,\dots,x_n)$ are defined as:

$$X_{k,n}(\vec{x}) = \prod_{|J|=k} (\sum_{j\in J} x_j).$$

The subset-sum polynomials verify the following identity:

$$X_{k,n}(x_1,\dots,x_n) = X_{k-1,n-1}(x_1 + \frac{x_n}{k-1},\dots,x_{n-1} + \frac{x_n}{k-1}) \cdot X_{k,n-1}(x_1,\dots,x_{n-1}).$$

This leads to a parallel computation procedure of depth n in the language with division.

Lemma 2 Let $u_1(\vec{x}),\dots,u_s(\vec{x}) \in \mathbb{C}[\vec{x}]$ be symmetric polynomials and $U \in \mathbb{C}[\vec{u}]$ some polynomial such that the following identity holds:

$$\forall\, \vec{x}\ \ X_{k,n}(\vec{x}) = U(\vec{u}(\vec{x})).$$

Then the polynomial U is absolutely irreducible, seen as polynomial in the new variables u_i.

The subset-sum polynomials $X_{k,n}$ are symmetric homogenous polynomials of degree $\binom{n}{k}$ with coefficients in $\mathbb{Z}$, so they can be expressed as polynomials in any basis of the ring of symmetric polynomials in n variables. There exist and are uniquely determined polynomials $\Sigma_{k,n} \in \mathbb{Z}[\sigma_1,\dots,\sigma_n]$ such that for all $\vec{x}$ hold $X_{k,n}(\vec{x}) = \Sigma_{k,n}(\vec{\sigma}(\vec{x}))$.

Hence polynomials $\Sigma_{k,n}(\vec{\sigma})$ are in particular absolutely irreducible. For other bases of symmetric polynomials, see [6] and [12]. It is not clear if other bases would be better to use if one tries to prove P ($\mathbb{C}$) $\neq$ NP ($\mathbb{C}$) using symmetric subset-sum problems like SSS $(n,2n)$.

At this point I remark that this polynomials $\Sigma_{k,n}$ can be symbolically computed by Maple using the function `simplify` but this works only for small values of n.

4 Eliminating equality tests

We focus on the problem SSS $(n,2n)$ over $\mathbb{C}$. Suppose that there is a machine over $\mathbb{C}$ solving this decision problem in polynomial time. An input $\vec{\sigma}$ consisting in algebraically independent elements σ_i defines a computation path where all tests $p(\vec{\sigma}) = 0$ are answered negatively and the final result is a rejection. This is the so called generic path. Give now an input $\vec{\sigma}(\vec{x})$ produced by $2n-1$ algebraically independent elements x_i and the only one relation $x_1 + \cdots + x_n = 0$. The first positive test occurs on the generic path. The corresponding polynomial must be already divisible with $\Sigma_{n,2n}$, because $\Sigma_{n,2n}$ is irreducible. By exploring the computation path starting here it follows that:

Theorem 3 *If* SSS $(n,2n)$ *belongs to* P$(\mathbb{C})$ *then there is a polynomial* $p(n)$ *and a sequence of test-free circuits* (C_n) *with* $\leq p(n)$ *many gates in* $+,-,\cdot$ *such that for all* n *there are polynomials* $A_1,\dots,A_d \in \mathbb{C}[\sigma_1,\dots,\sigma_{2n}]$ *and positive natural numbers* e_i $(i = 1,\dots,d)$ *such that:*

- C_n *gets an input* $(\sigma_1,\dots,\sigma_{2n})$ *and outputs the values of the following* d *polynomials:*

$$\begin{array}{ccc} p_1 & = & A_1\Sigma_{n,2n}^{e_1} \\ \cdots & \cdots & \cdots\cdots\cdots \\ p_d & = & A_d\Sigma_{n,2n}^{e_d} \end{array}$$

- *The algebraic set* $V(A_1,\dots,A_d) = \emptyset$.

Indeed, $\Sigma_{n,2n}(\vec{\sigma}) = 0$ if and only if $p_1(\vec{\sigma}) = \cdots = p_d(\vec{\sigma}) = 0$. In order to be tested, these values have to be computed.

This situation is delicate in the following sense. If we only look at the equations, one can ask: why did he write down this system? Isn't it, that it was enough to compute and test $\Sigma_{n,2n}(\vec{\sigma}) = 0$? Yes, it is, but our machine doesn't make symbolic computation. The machine just works with complex numbers and tests equalities, but cannot perform any elimination. On the other hand is possible that the given polynomial multiples of $\Sigma_{n,2n}$ are easy to compute, but $\Sigma_{n,2n}$ itself not. It is also not known if the polynomials A_i or some divisors of them are easy to compute. So all that we can say about a computation path is that it should look like this, but this is really more as one could say if the polynomials $\Sigma_{n,2n}$ wouldn't have been irreducible.

In particular, for all n there must be a sequence of polynomial multiples of $\Sigma_{n,2n}$ which is evaluated by a uniform family of straight-line programs with $\leq p(n)$ many lines. Using a notion introduced in [1] for the problem Twenty Questions, if SSS $(n,2n)$ belongs to P$(\mathbb{C})$ then the polynomials $\Sigma_{n,2n}$ are ultimately easy to compute by straight-line programs.

5 The corresponding bit-cost problems

Consider the following bit-cost problems SSS $(n,2n)(\mathbb{C})$ and SSS $(n,2n)(\mathbb{Z})$:

Input: A character-string of length m over the alphabet $\{0,1,\dots,9,\natural,-\}$. We interpret $\natural$ as a separator and $-$ as minus. Inputs making sense are the sequences of decimal representations of some $2n < m$ many integers $z_1,\dots,z_{2n}$.

Question SSS $(n,2n)(\mathbb{C})$: Does the vector $\vec{z}$ belong to the irreducible set $\Sigma_{n,2n}(\vec{z}) = 0$?

Question SSS $(n,2n)(\mathbb{Z})$: Are there $x_1,\dots,x_{2n} \in \mathbb{Z}$ with $\vec{x} \in$ SS $(n,2n)$ and $z_i = \sigma_i(\vec{x})$ for all $i = 1,\dots,n$?

Are there algorithms able to solve these problems in a polynomial time $p(m)$?

Theorem 4 *SSS $(n,2n)(\mathbb{C})$ is* NP-*hard. SSS $(n,2n)(\mathbb{Z})$ is* NP-*complete.*

NP-hardness: We interpret the problem 3SAT in Kn and get an instance of Kn where the input elements are natural numbers and have decimal representations of the same length. Now we observe that:

$$(x_1,\dots,x_n,b) \in \text{Kn} \Leftrightarrow (nx_1 - b,\dots,nx_n - b,-b,\dots,-b) \in \text{SS }(n,2n).$$

Finally we observe that the number of digits of $\vec{\sigma}(\vec{y})$ depends polynomially in the number of digits of $\vec{y}$ and we get a polynomial computation time for the bit representation of $\vec{\sigma}(\vec{y})$.

6 Conclusions

- P = NP with unit-cost is equivalent to the existence of a polynomial time procedure of quantifier elimination from existential formulas to deterministic decision circuits.
- The unit-cost problem Symmetric Subset-Sum is defined by polynomials having short implicit definitions but which are probably hard to compute (evaluate).
- These polynomials are absolutely irreducible. Consequently, a decision problem is reduced to a computation problem.
- The corresponding bit-cost problems are NP-hard.

Acknowledgments: I would like to thank the referee for helpful suggestions.

References

[1] Lenore Blum, Felipe Cucker, Mike Shub, Steve Smale. *Complexity and real computation*. Springer Verlag, New York, 1997.

[2] Hervé Fournier, Pascal Koiran. *Lower bounds are not easier over the reals: inside PH*. Montanari, Ugo (ed.) et al., Automata, languages and programming. 27th international colloquium, ICALP 2000, Geneva, Switzerland, July 9-15, 2000. Proceedings. Berlin: Springer. Lecture Notes Comput. Sci. 1853, 832 - 843, 2000.

[3] Michael R. Garey, David S. Johnson. *Computers and intractability.* W. H. Freeman and Company, New York, 1979.

[4] Chrisine Gaßner. *The P-DNP Problem for Infinite Abelian Groups.* Journal of Complexity 17, 574 - 583, 2001.

[5] Armin Hemmerling. *On P versus NP for parameter-free programs over algebraic structures.* Mathematical Logic Quarterly 47, 1, 67-92, 2001.

[6] Ian G. Macdonald. *Symmetric functions and Hall polynomials.* Clarendon Press, Oxford, 1995.

[7] Bruno Poizat. *Les petits cailloux*. Aleas, Lyon, 1994.

[8] Mihai Prunescu. *A model-theoretical proof for $P \neq NP$ over all infinite Abelian groups.* Journal of Symbolic Logic 67, 1, 235 - 238, 2002.

[9] Mihai Prunescu. *$P \neq NP$ for all infinite Boolean algebras.* Mathematical Logic Quarterly 49, 2, 210 - 213, 2003.

[10] Mihai Prunescu. *Two situations with unit-cost: ordered abelian semigroups and some commutative rings.* Journal of Complexity, to appear in 2005.

[11] A. Rybalov. *On the P-NP problem over real matrix rings.* Theoretical Computer Science 314, 281 - 285, 2004.

[12] Bernd Sturmfels. *Algorithms in invariant theory.* Springer Verlag, Wien, 1993.

Mihai Prunescu made 1992 his licence in mathematics at the University Bucharest, Romania, and 1998 his Ph. D. at the University Konstanz, Germany. Since 1992 he is member of the Institute of Mathematics "Simion Stoilow" of the Romanian Academy. After working at the University Greifswald, he gets in the present a post-doctoral grant in the `Graduiertenkolleg Mathematische Logik und Anwendungen`, University Freiburg.

`Mihai.Prunescu@math.uni-freiburg.de`