On the Complexity of the Bernstein Combinatorial Problem*

Dominique Michelucci
Le2i, UMR CNRS 5158, Université de Bourgogne,
BP 47870, 21078 Dijon, France.
dmichel@u-bourgogne.fr

Sebti Foufou
Computer Science and Engineering, CENG, Qatar University, PO Box. 2713, Doha, Qatar.
Le2i, UMR CNRS 5158, Université de Bourgogne,
BP 47870, 21078 Dijon, France.
sfoufou@qu.edu.qa,sfoufou@u-bourgogne.fr

Arnaud Kubicki
Le2i, UMR CNRS 5158, Université de Bourgogne,
BP 47870, 21078 Dijon, France.
arnaud.kubicki@u-bourgogne.fr

Abstract

Every multivariate polynomial $p(x)$, $x = (x_1, \ldots, x_n) \in [0,1]^n$, is enclosed in the interval given by the smallest and the greatest of its coefficients in the Tensorial Bernstein Basis (TBB). Knowing that the total number of these TBB coefficients is exponential with respect to the number of variables n, $\prod_{i=1}^{n}(1+d_i)$, even if all partial degrees d_i equal 1, a combinatorial problem arises: is it possible to compute in polynomial time the smallest and the greatest coefficients? This article proves that the 3-SAT problem, known to be NP-complete, polynomially reduces to the above defined combinatorial problem, which let us consequently conclude that this problem is NP-hard.

Keywords: Bernstein polynomials, tensorial Bernstein basis, interval arithmetics, combinatorial complexity

1 Introduction

Multivariate polynomials are frequently used in formalizing and solving various engineering and physical science problems.

Expressing a multivariate polynomial $p(x)$, $x = (x_1, \ldots x_n) \in \mathbb{R}^n$ in the Tensorial Bernstein Basis (TBB) gives a tight enclosure of $p(x \in [0,1]^n)$: the smallest coefficient

*Submitted: July 5, 2012; Revised: December 14, 2012; Accepted:December 16, 2012.

in the TBB is a lower bound of $p(x \in [0,1]^n)$, and the greatest coefficient in the TBB is an upper bound. The lower (upper) bound is exact, *i.e.* it is reached, when the smallest (greatest) coefficient is related to a vertex of the hypercube $[0,1]^n$, as explained in the next section. For multivariate polynomials with all partial degrees at most 1, the lower and the upper bounds are exact, see section 3.

However, a combinatorial problem arises, the number of TBB coefficients is exponential with respect to the number of variables: $\prod_{i=1}^{n}(1+d_i)$ where d_i is the partial degree of p in x_i. Even if all partial degrees d_i equal 1, the number of coefficients, *i.e.* the cardinality of the TBB, is exponential, it is equal to 2^n. We are only interested in the smallest and the greatest coefficients. We call this problem the Bernstein Combinatorial Problem (BCP). The acronym BCP will be used to refer to this problem throughout the remaining part of this paper.

The naive method to solve BCP computes all TBB coefficients, and then finds the smallest and the greatest ones. Each TBB coefficient is a linear combination of the m non-zero monomial coefficients in the canonical basis. It is polynomial time, assuming that m is polynomial in n, *i.e.* that the polynomial is sparse in the canonical basis. Weights in the linear combination are products of binomial coefficients and are given in standard formulas in section 2. The naive method is clearly exponential and can be used only for small polynomial systems, say $n \leq 6$ or 7, with low degree. It is sufficient for some classical problems in computer graphics. But, in many new domains of computer graphics or computer engineering like geometric constraint solving, some problems need to solve polynomial systems with hundreds or thousands of unknowns. Therefore it becomes essential to know the complexity of the BCP.

In this study, we consider the complexity of the BCP, and show that most likely it is not possible to solve it in polynomial time, by proving that the 3-SAT problem polynomially reduces to the BCP, thus the latter is NP-hard. 3-SAT, NP-completeness, NP-hardness and complexity theory are explained in textbooks such as [14, 10]. The paper is organized as follows: Section 2 presents tensorial Bernstein bases and the BCP. Section 3 presents the polynomial reduction of the 3-SAT problem to the BCP. To the best of our knowledge this result is new. Section 4 presents the principles of some workarounds. Section 5 concludes the paper.

2 Tensorial Bernstein Bases

Knowledgeable readers can skip this section, Farin's textbook presents the material included hereunder in more details [4].

For univariate degree d polynomials, Bernstein polynomials $B_0^{(d)}(x), \ldots B_d^{(d)}(x)$ form a basis, and are defined by:

$$B_i^{(d)}(x) = \binom{d}{i} x^i (1-x)^{d-i}, \quad i = 0, \ldots d$$

Clearly, for $x \in [0,1]$ they are always non-negative and smaller than or equal to 1. Moreover, their sum equals 1 for every x, because of Newton's expansion:

$$1^d = (x + (1-x))^d = \sum_{i=0}^{d} \binom{d}{i} x^i (1-x)^{d-i} = \sum_{i=0}^{d} B_i^{(d)}(x)$$

Thus for $x \in [0,1]$, $p(x) = \sum p_i B_i(x)$ is a linear convex combination of the p_i. So $p(x \in [0,1])$ belongs to $[\min p_i, \max p_i]$. Moreover, since $p(0) = p_0$, if p_0 is the smallest (or

greatest) coefficient, this lower (or upper) bound is exact, *i.e.* it is reached. Similarly, $p(1) = p_d$, so if p_d is the smallest (or greatest) coefficient, this lower (or upper) bound is exact.

Standard formulas [4, 23] shown below give the linear mapping (*i.e.* representable with a square invertible matrix) which converts an univariate polynomial from the canonical basis $(x^0, x^1, \dots x^d)$ to the Bernstein basis $(B_0^{(d)}(x), \dots B_d^{(d)}(x))$:

$$x^k = \frac{1}{\binom{d}{k}} \sum_{i=k}^{d} \binom{i}{k} B_i^{(d)}(x) \ , \quad k = 0, \dots d$$

$$x = \frac{1}{d} \sum_{i=0}^{d} i \, B_i^{(d)}(x)$$

$$x^0 = 1 = \sum_{i=0}^{d} B_i^{(d)}(x) \ : \text{ their sum equals 1.}$$

A consequence of these formulas is that the graph of a polynomial $p(x), x \in [0, 1]$, *i.e.* the curve $y = p(x)$ for $x \in [0, 1]$, lies inside the convex hull of the $d + 1$ points $(\frac{i}{d}, p_i), i = 0, \dots d$, called "control points" in Computer Aided Design [11] or Computer Graphics. Indeed

$$x = \sum_{i=0}^{d} \frac{i}{d} \, B_i^{(d)}(x)$$

If each p_i is a point in $\mathbb{R}^k$ space, then $p(x \in [0, 1]^k)$ describes an arc of curve in $\mathbb{R}^k$, each point of which is a convex combination of the p_i $i = 0, \dots d$; so this curve lies in the convex hull of the points p_i, $p_i \in \mathbb{R}^k$; this curve is called a Bézier curve, and is often used in Computer Aided Design [11]. Numerous interactive graphic software systems enable designers to edit such curves, by selecting and dragging its control points; the curve is refreshed and displayed in interactive time.

For two variables x_1 and x_2, the tensorial Bernstein basis, with degree d_1 in x_1 and degree d_2 in x_2 is:

$$B_{i_1,i_2}^{(d_1,d_2)}(x_1, x_2) = B_{i_1}^{(d_1)}(x_1) B_{i_2}^{(d_2)}(x_2)$$

and is the tensorial product:

$$(B_0^{(d_1)}(x_1), B_1^{(d_1)}(x_1), \dots B_{d_1}^{(d_1)}(x_1)) \ \otimes \ (B_0^{(d_2)}(x_2), B_1^{(d_2)}(x_2), \dots B_{d_2}^{(d_2)}(x_2))$$

For short, $B_{i_1}^{(d_1)}(x_1) B_{i_2}^{d_2}(x_2)$ is denoted $B_{i_1,i_2}^{(d_1,d_2)}(x)$ where x stands for (x_1, x_2). Degrees (d_1, d_2) can be omitted if no confusion is possible. Similarly the coefficient of $B_{i_1,i_2}^{(d_1,d_2)}(x)$ is denoted p_{i_1,i_2}, or p_i where i is a multi index $i = i_1, i_2$.

The convex hull property still holds:

$$\min p_{i_1,i_2} \leq p(x_1, x_2) \leq \max p_{i_1,i_2}$$

for $(x_1, x_2) \in [0, 1]^2$.

The square domain $[0, 1]^2$ has 2^2 vertices: $(0, 0)$, $(0, 1)$, $(1, 0)$, $(1, 1)$, and $p(0, 0) = p_{0,0}$, $p(0, 1) = p_{0,d_2}$, $p(1, 0) = p_{d_1,0}$, $p(1, 1) = p_{d_1,d_2}$. When the index of the smallest (greatest) TBB coefficient is a vertex in $\{p_{0,0}, p_{0,d_2}, p_{d_1,0}, p_{d_1,d_2}\}$ the lower (upper) bound of $p(x \in [0, 1]^2)$ is exact.

A trivial but key remark for the proof in the next section is: when all degrees $d_1, d_2, \dots d_n$ equal 1, all coefficients indices in the TBB are vertices. Thus the lower

(upper) bound given by the smallest (greatest) TBB coefficient is exact, and is reached at the corresponding vertex of the hypercube $[0,1]^n$.

Bézier patches used in Computer Aided Design [4, 11] are parametric algebraic surfaces:

$$\mathcal{S} = \{(x,y,z) \mid x = f(x_1,x_2), y = g(x_1,x_2), z = h(x_1,x_2), (x_1,x_2) \in [0,1]^2\}$$

where the polynomials f, g, h are given in the TBB. 3D coefficients $(f_{i_1,i_2}, g_{i_1,i_2}, h_{i_1,i_2})$ in the TBB are called control points. Again, users can interactively edit patches, by selecting and dragging control points.

Similarly, for n variables $x_1, \dots x_n$, the tensorial Bernstein basis with partial degrees d_i in x_i is the tensor product of the univariate Bernstein bases $B^{(d_i)}(x_i)$; its cardinality is the product of the partial degrees plus one: $\prod_{i=1}^{n}(1+d_i)$. It is clearly exponential with n, the number of variables. If all $d_i = 1$, it is 2^n. Thus even in the multilinear case, the TBB has an exponential number of basis functions and coefficients.

The convex hull property of the univariate Bernstein basis extends to the TBB, and $p(x), x = (x_1, \dots x_n) \in [0,1]^n$ lies in $[\min p_i, \max p_i]$, here i is a multi index $i = i_1, i_2, \dots i_n$. Moreover, if the index of the smallest (greatest) TBB coefficient is a vertex of the hypercube $[0,1]^n$, then this lower (upper) bound is exact. It is the case when the multivariate polynomial is multilinear, *i.e.* all its partial degrees in all variables equal at most one.

Thus expressing a polynomial into the TBB gives a (usually tight) enclosure of $p([0,1]^n)$. Some variable change permits to compute enclosures of p over any box: see [4] for details and Casteljau's method. These convenient properties are intensively used in Computer Aided Design, for solving algebraic systems, computing intersection curves or points between 3D algebraic surfaces, ray tracing implicit algebraic surfaces (given by their polynomial equation $p(x,y,z) = 0$) or Bézier patches [9, 18, 3, 15]. However, n is low in these Computer Aided Design problems.

When one wants to use the TBB for larger n, one faces the Bernstein combinatorial problem: there is an exponential number of Bernstein polynomials (and coefficients) in the TBB, and we are interested only in the smallest and the greatest one. The multi index of the smallest and the greatest TBB coefficient are unknown. Is it possible to compute these two TBB coefficients in polynomial time? or: what is the complexity of the BCP?

The rest of this article considers, without loss of generality, only the computation of the smallest coefficient in the TBB: the greatest coefficient for p is just the opposite of the smallest coefficient of $-p$. Thus both problems are equivalent in complexity.

Polynomials are usually sparse in the tensorial canonical basis: their size is polynomial in n. For the reduction below, they are $O(n^3)$. They are no more sparse in the TBB. For instance, all TBB coefficients of the constant polynomial $p(x) = 1$ are equal to 1. The exponential size of this polynomial in the TBB is the main reason for the NP-hardness of the BCP.

Example. We consider now an example with $n = 2$ for simplicity, with $d_1 = d_2 = 2$, x_1 is renamed x, and x_2 is renamed y. Consider the polynomial

$$\begin{aligned} p(x,y) = 7 + 8x + 3y + 2xy + 4x^2 - 5y^2 = 7x^0y^0 \\ +8x^1y^0 + 3x^0y^1 + 2x^1y^1 + 4x^2y^0 - 5x^0y^2 \end{aligned}$$

The formulas for converting from the canonical basis to the Bernstein basis are:

$$\begin{array}{rcrcrcr}
1 & = & 1\,B_0(x) & + & 1\,B_1(x) & + & 1\,B_2(x) \\
x & = & 0\,B_0(x) & + & 1/2\,B_1(x) & + & 1\,B_2(x) \\
x^2 & = & 0\,B_0(x) & + & 0\,B_1(x) & + & 1\,B_2(x) \\
1 & = & 1\,B_0(y) & + & 1\,B_1(y) & + & 1\,B_2(y) \\
y & = & 0\,B_0(y) & + & 1/2\,B_1(y) & + & 1\,B_2(y) \\
y^2 & = & 0\,B_0(y) & + & 0\,B_1(y) & + & 1\,B_2(y)
\end{array}$$

For simplicity, we note $X_k = B_k(x)$ and $Y_k = B_k(y)$. The TBB is: $\{X_0Y_0,\ X_0Y_1,\ X_0Y2,\ X_1Y_0,\ X_1Y_1,\ X_1Y2,\ X_2Y_0,\ X_2Y_1,\ X_2Y_2\}$, and contains 9 elements. To compute the coefficient of, say X_1Y_2, we must multiply row by row in the next tableau the content of the column of coefficients with the content of the column X_1 and with the content of the column Y_2. Then we compute the sum of the obtained (rightmost) column, which gives 10 here.

$$\begin{array}{rcccl}
7x^0y^0 & = & 7 \times (1X_0 + 1X_1 + 1X_2) \times (1Y_0 + 1Y_1 + 1Y_2) & = & \ldots + 7\,X_1Y_2 + \ldots \\
4x^2y^0 & = & 4 \times (0X_0 + 0X_1 + 1X_2) \times (1Y_0 + 1Y_1 + 1Y_2) & = & \ldots + 0\,X_1Y_2 + \ldots \\
8x^1y^0 & = & 8 \times (0X_0 + 1/2X_1 + 1X_2) \times (1Y_0 + 1Y_1 + 1Y_2) & = & \ldots + 4\,X_1Y_2 + \ldots \\
3x^0y^1 & = & 3 \times (1X_0 + 1X_1 + 1X_2) \times (0Y_0 + 1/2Y_1 + 1Y_2) & = & \ldots + 3\,X_1Y_2 + \ldots \\
2x^1y^1 & = & 2 \times (0X_0 + 1/2X_1 + 1X_2) \times (0Y_0 + 1/2Y_1 + 1Y_2) & = & \ldots + 1\,X_1Y_2 + \ldots \\
4x^2y^0 & = & 4 \times (0X_0 + 0X_1 + 1X_2) \times (1Y_0 + 1Y_1 + 1Y_2) & = & \ldots + 0\,X_1Y_2 + \ldots \\
-5x^0y^2 & = & -5 \times (1X_0 + 1X_1 + 1X_2) \times (0Y_0 + 0Y_1 + 1Y_2) & = & \ldots - 5\,X_1Y_2 + \ldots
\end{array}$$

3 Reduction of 3-SAT to the Bernstein Combinatorial Problem

For a given multivariate polynomial $p(x)$ in n variables $x = (x_1, \ldots x_n)$, and with *polynomial size in n* in the canonical basis, the BCP is to compute the smallest TBB coefficient, *i.e.* the smallest coefficient of p in the tensorial Bernstein basis.

We assume for simplicity that p is given in the tensorial *canonical* base, and that it is sparse enough, so that its size is polynomial in n; in the reduction below of 3-SAT to BCP, this size is $O(n^3)$. Actually, any representation of p with polynomial size in n is appropriate, for instance a straight line program representation, also called DAG (Directed Acyclic Graph) representation.

This section proves that the BCP is NP-hard by reducing, in polynomial time, any instance of the NP-complete 3-SAT problem to the BCP. The following three lemmas L1, L2 and L3 will be needed to reduce the 3-SAT problem to BCP.

Let $p(x)$ be a *multilinear* polynomial in n variables $x = (x_1, \ldots x_n)$. From now on, we will omit superscripts in the univariate case in $B_v^{(1)}(x_k), v \in \{0,1\}$, and in the multivariate case in $B_v^{(1_n)}(x), x = (x_1, \ldots x_n), v \in \{0,1\}^n$.

Lemma L1. In the univariate and in the multivariate cases, $B_v(v) = 1$ for all $v \in \{0,1\}^n$, and $B_v(w) = 0$ when $w \neq v$, with $w \in \{0,1\}^n$.

Proof. First, consider the case $n = 1$, we have $B_0(x) = 1 - x$ and $B_1(x) = x$, thus for any v in $\{0,1\}$, $B_v(v) = 1$ and $B_v(1-v) = 0$. Thus Lemma L1 holds in the univariate case.

Second, consider the case $n > 1$. Let $v \in \{0,1\}^n$ be a vertex or a multi index: $v_i \in \{0,1\}$. Then $B_v(v) = B_{v_1}(v_1) \ldots B_{v_n}(v_n) = 1$ because every $B_{v_k}(v_k)$ equals

1 after Lemma L1 in the univariate case. Consider now a multi index or vertex $w \in \{0,1\}^n$ not equal to v; thus $v_k = 1 - w_k$ for some integer k, $1 \leq k \leq n$; then $B_v(w) = B_{v_1}(w_1) \dots B_{v_k}(w_k) \dots B_{v_n}(w_n) = 0$ because $B_{v_k}(w_k) = B_{v_k}(1 - v_k) = 0$ after Lemma L1 in the univariate case. Thus $B_v(v) = 1$ and $B_w(v) = 0$ when $w \neq v$, in the multivariate case as well. **QED.**

Lemma L2. $p(x) = \sum_{v \in \{0,1\}^n} p(v) B_v(x)$

Proof. Express $p(x)$ in the TBB:

$$p(x) = \sum_{v \in \{0,1\}^n} p_v B_v(x)$$

We want to prove that $p_v = p(v)$.

$$\begin{aligned} p(v) &= \sum_{w \in \{0,1\}^n} p_w B_w(v) && (1) \\ &= p_v B_v(v) + \sum_{w \neq v} p_w B_w(v) \text{ but } w \neq v \Rightarrow B_w(v) = 0 \text{ by L1.} && (2) \\ &= p_v B_v(v) \text{ but } B_v(v) = 1 \text{ by L1.} && (3) \\ &= p_v && (4) \end{aligned}$$

Which terminates the proof of Lemma L2 showing that $p(v) = p_v$ for each vertex $v \in \{0,1\}^n$. **QED.**

Let p_s be the smallest TBB coefficient of p (or one of the smallest, if there are many), its multi index is the vertex s in $\{0,1\}^n$; similarly let p_g be the greatest TBB coefficient of p (or one of the greatest, if there are many), its multi index is the vertex g of $\{0,1\}^n$.

Lemma L3. $p([0,1]^n) = [p_s, p_g]$. In words, for a multilinear polynomial, the smallest and the greatest of its TBB coefficients give the exact range of $p([0,1]^n)$.

Proof. We already know that $p([0,1]^n) \subset [p_s, p_g]$: it is a classical property of TBB. Now, after Lemma L2, $p_s = p(s)$ and the vertex s belongs to the hypercube $[0,1]^n$. Similarly, $p_g = p(g)$ and the vertex g belongs to the hypercube $[0,1]^n$. Thus $p([0,1]^n) \subset [p(s), p(g)]$ and we conclude that $p([0,1]^n) = [p(s), p(g)]$. **QED.**

We can now reduce the 3-SAT problem to BCP as follows:

Theorem. BCP is NP-hard.

Proof. Let $P(x)$ be an instance of the 3-SAT problem. We first encode the 3-SAT problem into a multilinear polynomial $p = \varphi(P)$: φ maps boolean values to $\{0,1\}$, boolean unknowns x_i to variables x_i (for convenience and simplicity, we use the same names for boolean and numeric unknowns), clauses to clausal polynomials, and 3-SAT problems to 3-SAT polynomials as follows: define T = true and F = false. We use $\varphi(T) = 0$ and $\varphi(F) = 1$. Though it may seem counter-intuitive, this convention is convenient: for $v \in \{0,1\}^n$, the polynomial $p(v)$ will count the number of clauses violated by the assignment $\varphi^{-1}(v)$. The polynomial $p = \varphi(P)$ is the sum of the clausal polynomials of all clauses in P. The clausal polynomial is defined as follows:

- $p(x,y,z) = \varphi(x \ \vee \ y \ \vee \ z) = xyz$. Thus

$$p(\varphi(F), \varphi(F), \varphi(F)) = p(1,1,1) = 1$$

for the unique assignment which violates the clause, and p vanishes for the other 7 assignments.

- $p(x,y,z) = \varphi(x \ \vee \ y \ \vee \ \neg z) = xy(1-z) = xy - xyz$. Thus

$$p(\varphi(F), \varphi(F), \varphi(T)) = p(1,1,0) = 1$$

for the unique assignment which violates the clause, and p vanishes for the other 7 assignments.

- $p(x,y,z) = \varphi(x \ \vee \ \neg y \ \vee \ \neg z) = x(1-y)(1-z) = x - xy - xz + xyz$. Thus

$$p(\varphi(F), \varphi(T), \varphi(T)) = p(1,0,0) = 1$$

for the unique assignment which violates the clause, and p vanishes for the other 7 assignments.

- $p(x,y,z) = \varphi(\neg x \ \vee \ \neg y \ \vee \ \neg z) = (1-x)(1-y)(1-z) = 1 + xy + xz + yz - xyz$. Thus

$$p(\varphi(T), \varphi(T), \varphi(T)) = p(1,1,1) = 1$$

for the unique assignment which violates the clause, and p vanishes for the other 7 assignments.

Thus, the clausal polynomial equals 1 for the assignment which violates the clause, and vanishes for assignments which satisfy the clause. Moreover, each clausal polynomial is size $O(1)$ in the tensorial *canonical* basis. The 3-SAT polynomial $p(x) = (\varphi(P))(x)$ of a 3-SAT instance $P(x)$ is the sum of all clausal polynomials of every clause of P; thus $p(v)$ counts the number of clauses violated by the assignment $\varphi^{-1}(v)$.

The 3-SAT polynomial p has the following properties:

- p is multilinear: all partial degrees with respect to each variable x_i are 1.
- The total degree of $p(x)$ is 3: each monomial in the tensorial *canonical* basis involves zero, one, two, or three variables.
- The total degree of the TBB is n. Indeed, the TBB can represent polynomials $x_1 x_2 \dots x_n, \dots (1-x_1)(1-x_2)\dots(1-x_n)$, the total degree of which is n.
- If the 3-SAT instance P has K clauses, the size of the 3-SAT polynomial p is $O(K)$ in the tensorial *canonical* basis, which is the size of the 3-SAT instance P in the 3-CNF (Conjunctive Normal Form).
- $K = O(n^3)$: there are $n(n-1)(n-2)/6$ triples of distinct variables amongst n unknowns x_i, and there are $2^3 = 8$ possible clauses for each triple, because each variable x_i appears either negatively ($\neg x_i$) or positively (x_i). $O(n^3)$ is also the size of the 3-SAT polynomial p in the tensorial *canonical* basis. This is polynomial in n.
- In the TBB, $p(x)$ is defined by 2^n coefficients, one coefficient for each vertex, or multi index, in $\{0,1\}^n$. Thus the representation of p in the TBB has exponential size. The TBB coefficients of p are integers, in $[0,K]$: remember that for an assignment V and the corresponding vertex $v = \varphi(V) \in \{0,1\}^n$, the coefficient $p_v = p(v)$ is the number of clauses of P violated by the assignment V. Thus $p_v = p(v)$ is an integer in $[0,K]$. As usual, we assume for simplicity that the size of K is constant.
- $p(v)$ counts the number of clauses violated by the assignment $\varphi^{-1}(v)$.

We can now proceed to the reduction of the 3-SAT problem to the BCP. Let P be the 3-SAT instance in n variables, and $p = \varphi(P)$ be the corresponding 3-SAT polynomial, represented in the tensorial *canonical* basis, so that P and p have both size $O(K) = O(n^3)$ which is polynomial in n. Assume that we can compute, in polynomial time, the value p_s of the smallest TBB coefficient of p (or one of the smallest TBB coefficients, if there are many). Moreover, by Lemma L2, p_s is the value $p(s)$ of p at some vertex s of the hypercube $[0,1]^n$. If p_s is zero, the 3-SAT instance P is satisfiable: the assignment $\varphi^{-1}(s)$ satisfies P. If p_s is non zero, it is the smallest number of violated clauses, thus $K - p_s$ is the maximum number of satisfiable clauses: we have solved in polynomial time the 3-SAT problem, but 3-SAT is NP-complete, thus the BCP is NP-hard. **QED.**

Example. Which boolean values of x, y, z, t satisfy the following problem P?

$$(C_1 : x \vee y \vee z) \wedge (C_2 : x \vee y \vee \bar{z}) \wedge (C_3 : x \vee \bar{y} \vee z) \wedge$$
$$(C_4 : \bar{x} \vee y \vee t) \wedge (C_5 : \bar{x} \vee \bar{y} \vee \bar{t}) \wedge (C_6 : x \vee \bar{y} \vee \bar{z}) \wedge (C_7 : \bar{x} \vee y \vee z)$$

Clauses are numbered $C_1, \ldots C_7$ for convenience. P is satisfied with these 3 assignments:

$$(x, y, z, t) \in \{(T, T, T, F), (T, T, F, F), (T, F, T, T)\}$$

where T is true and F is false. The 7 clausal polynomials are given below, in the tensorial *canonical* base, and in the TBB $B^{(1_4)}(x, y, z, t)$:

$$\begin{array}{lcll}
\varphi(C_1 : x \vee y \vee z) & = & xyz & = & B_{1110} + B_{1111} \\
\varphi(C_2 : x \vee y \vee \bar{z}) & = & xy(1-z) & = & B_{1100} + B_{1101} \\
\varphi(C_3 : x \vee \bar{y} \vee z) & = & x(1-y)z & = & B_{1010} + B_{1011} \\
\varphi(C_4 : \bar{x} \vee y \vee t) & = & (1-x)yt & = & B_{0101} + B_{0111} \\
\varphi(C_5 : \bar{x} \vee \bar{y} \vee \bar{t}) & = & (1-x)(1-y)(1-t) & = & B_{0000} + B_{0010} \\
\varphi(C_6 : x \vee \bar{y} \vee \bar{z}) & = & x(1-y)(1-z) & = & B_{1000} + B_{1001} \\
\varphi(C_7 : \bar{x} \vee y \vee z) & = & (1-x)yz & = & B_{0110} + B_{0111}
\end{array}$$

Remark: in this example, $n = 4$, which is too small to make visible the exponential growth of the size of the TBB, relatively to the size of the tensorial *canonical* basis. This growth is due to the fact that the constant polynomial 1 does not belong to the TBB, and must be converted. For instance, for the first line, clause $C1$: $xyz = B_1(x)B_1(y)B_1(z)(B_0(t) + B_1(t)) = B_{1110} + B_{1111}$. In the general case, expressing a clausal polynomial in the TBB requires 2^{n-3} Bernstein polynomials.

The 3-SAT polynomial $\varphi(P)$ of the problem P, *i.e.* the sum of all $\varphi(C_k)$, is:

$$B_{0000} + B_{0010} + B_{0101} + B_{0110} + 2B_{0111} + B_{1000} + B_{1001}$$
$$+B_{1010} + B_{1011} + B_{1100} + B_{1101} + B_{1110} + B_{1111}$$

where B_u are sorted in lexicographic order for convenience. The TBB coefficients are zero for:

- B_{0001}, thus $\varphi^{-1}(0, 0, 0, 1) = (T, T, T, F)$ satisfies P.
- B_{0011}, thus $\varphi^{-1}(0, 0, 1, 1) = (T, T, F, F)$ satisfies P.
- B_{0100}, thus $\varphi^{-1}(0, 1, 0, 0) = (T, F, T, T)$ satisfies P.

The greatest TBB coefficient in the 3-SAT polynomial is 2, for B_{0111}, thus the corresponding assignment: $\varphi^{-1}(0, 1, 1, 1) = (T, F, F, F)$ maximizes the number of unsatisfied clauses (these 2 non-satisfied clauses are $C_4 : \bar{x} \vee y \vee t \Rightarrow \varphi(C_4) = (1-x)yt =$

$B_{0101} + B_{0111}$ and $C_7 : \bar{x} \vee y \vee z \Rightarrow \varphi(C_7) = (1-x)yz = B_{0110} + B_{0111}$).

Before concluding this section, we note the following remarks:

Remark 1. Satisfying a 3-SAT problem $P(x)$, $x = (x_1, \dots x_n)$ is equivalent to solving the algebraic system: $p(x) = 0, x_i^2 - x_i = 0 \; \forall i = 1, \dots n$ where $p = \varphi(P)$. Clearly if x^* is a root of this system, then the assignment $\varphi^{-1}(x^*)$ satisfies the problem $P(x)$. This reduction permits to measure the complexity of solving polynomial systems, and of devoted algorithms like the Buchberger algorithm which computes Gröbner bases (also called standard bases). Our result is compatible with previous ones, like the complexity of the ideal membership problem.

Remark 2. Our result of the NP-hardness of BCP is also consistent with Gaganov theorem [8] which proves that computing the range of a multivariate polynomial over a box is NP-hard.

Remark 3. An anonymous reviewer noticed that, actually, the previous proof of the NP-hardness of BCP polynomially reduces MAX-3-SAT to BCP: the MAX-3-SAT is to compute the maximum number of satisfiable clauses, and 3-SAT is its decision version. Thus the theory of hardness of approximation, which considers the complexity of approximating problems like MAX-3-SAT, may shed some light on the complexity of approximating BCP, for instance with some non trivial bounds. However, the NP-hardness of BCP is sufficient for our purpose, which is to show that tensorial Bernstein bases do not scale for large n.

In conclusion, this section reduced in polynomial time the 3-SAT problem to the BCP. But 3-SAT is a well known NP-complete problem [10], thus the BCP is NP-hard. If the conjecture P$\neq$ NP (as usually assumed) holds, then there is no polynomial time algorithm for BCP, and for tight approximations of BCP. Conversely, if a polynomial time method is discovered for the BCP, it will solve in polynomial time all NP-complete problems, *e.g.* 3-SAT. It will also solve the MAX-3-SAT problem in polynomial time, and make the complexity hierarchy collapse.

4 Workarounds

This section presents the principles of some workarounds, for bypassing the BCP. When the number n of variables is small, in practice less than 7, it is possible to compute all coefficients in the TBB [9, 18, 17].

For greater but moderate n, Andrew P. Smith [21] proposes a lazy scheme: three tests (monotonicity, uniqueness, and dominance) dramatically reduce the number of computed Bernstein coefficients for "the types of polynomials encountered in global optimization problems", to quote Smith. However the method is still exponential in the worst cases, since the BCP is NP-hard. All that is consistent with the well known fact that not all instances of the SAT problem are hard.

Patrikalakis *e*t al. [19] use the simplicial (or barycentric) Bernstein basis, instead of the tensorial Bernstein basis. The barycentric Bernstein basis has a polynomial number of elements. However its domain is a simplex, *i.e.* the smallest and the greatest coefficients bound the possible values of a polynomial not inside the hypercube $[0, 1]^n$, but inside the simplex $0 \leq x_i, \sum x_i = 1$. It is not as convenient as an hypercube or box [16, 5].

In [6, 7] Michelucci *e*t al. proposed another, polynomial time, workaround, which relies on the definition of a Bernstein polytope. For example, for degree 2 multivariate

polynomials, the manifold $\mathcal{Q} \subset \mathbb{R}^N$, $N = n + n(n+1)/2$:

$$\mathcal{Q} = \{(x_1, \dots x_n, x_{11} = x_1^2, \dots x_{nn} = x_n^2, x_{12} = x_1 x_2, \dots x_{n-1,n} = x_{n-1} x_n)\}$$

where $(x_1, \dots x_n) \in [0,1]^n$, is first enclosed in a polytope (convex and bounded polyhedron) bounded with a polynomial number of hyperplanes: each hyperplane is given by the non-negativity of a Bernstein polynomial, namely:

$$\begin{aligned}
&0 \le B_0^{(2)}(x_k) = (1-x_k)^2 = x_k^2 - 2x_k + 1 \\
&\quad \Rightarrow 0 \le x_{kk} - 2x_k + 1 \; \forall k \in [1,n] \\
&0 \le B_1^{(2)}(x_k) = 2x_k(1-x_k) \Rightarrow 0 \le 2x_k - 2x_{kk} \\
&0 \le B_2^{(2)}(x_k) = x_k^2 \Rightarrow 0 \le x_{kk} \\
&0 \le B_0^{(1)}(x_i) B_0^{(1)}(x_j) = (1-x_i)(1-x_j) \\
&\quad \Rightarrow 0 \le 1 - x_i - x_j + x_{ij} \; \forall i \in [1, n-1] \forall j \in [i+1, n] \\
&0 \le B_0^{(1)}(x_i) B_1^{(1)}(x_j) = (1-x_i)x_j \Rightarrow 0 \le x_j - x_{ij} \\
&0 \le B_1^{(1)}(x_i) B_0^{(1)}(x_j) = x_i(1-x_j) \Rightarrow 0 \le x_i - x_{ij} \\
&0 \le B_1^{(1)}(x_i) B_1^{(1)} 1(x_j) = x_i x_j \Rightarrow 0 \le x_{ij}
\end{aligned}$$

It is possible to add n inequalities: $B_1^2(x_k) \le 1/2$. The Bernstein polytope can be defined for higher degrees, and for sparse polynomials its complexity (its number of hyperplanes) remains polynomial. Indeed, not all Bernstein polynomials are used, for instance in the previous example, the inequalities $0 \le B_{i_1}^{(2)}(x_1) \dots B_{i_n}^{(2)}(x_n)$, $i_k \in \{0,1,2\}$, $k \in [1,n]$ are unused. Computing polynomial ranges for $p(x \in [0,1]^n)$, or reducing the domain $x \in [0,1]^n$ (*i.e.* computing $\min x_i, \max x_i, \forall i \in [1,n]$) while preserving included roots of a given polynomial system $F(x) = 0$, becomes Linear Programming problems in the N variables x_i, x_{ii}, x_{ij}. Linear Programming is solvable in polynomial time (not strongly polynomial, though) with the ellipsoid method, as realized by Khachiyan [12, 13, 2] or with a bounding simplex method as realized by Levin and Yamnitsky [24]. In practice, these Linear Programming problems are solved with the Dantzig simplex method or better the revised simplex method [2], or with interior point methods. Note that, since this workaround is polynomial time, the intervals it provides are usually, but not always, less tight than the intervals provided by the smallest and greatest coefficients in the TBB [7, 1, 20, 22].

Though polynomial time, Linear Programming is costly. Maybe linear algebra is sufficient? Our first experiments suggest that resorting to Linear Programming cannot be avoided for problems in high dimension. Thus we are currently implementing Linear Programming methods on a Graphics Processing Unit (GPU).

5 Conclusion

This paper proves that computing the smallest or the greatest TBB coefficient of a sparse multivariate polynomial (*i.e.* with polynomial size) is NP-hard. It means, in practice, that the tensorial Bernstein bases do not scale well when n, the number of variables, increases. Several workarounds are also presented.

Acknowledgements

This research work has been funded by NPRP grant number NPRP 09-906-1-137 from the Qatar National Research Fund (a member of the Qatar Foundation).

References

[1] Willem F. Bronsvoort, Daniel Gonsor, William C. Regli, Thomas A. Grandine, Jan H. Vandenbrande, Jens Gravesen, and John Keyser, editors. *Proceedings of the 2009 ACM Symposium on Solid and Physical Modeling, San Francisco, California, USA, October 5-8, 2009*. ACM, 2009.

[2] V. Chvàtal. *Linear Programming*. W.H. Freeman, 1983.

[3] I. Z. Emiris, B. Mourrain, and E. Tsigaridas. Real algebraic numbers: Complexity analysis and experimentations. *In Reliable Implementation of Real number Algorithms: Theory and Practice, LNCS*, 5045:57–82, 2008.

[4] G. Farin. *Curves and Surfaces for CAGD: A Pratical Guide*. Morgan-Kaufmann, San Diego, CA., 1988.

[5] Christoph Fünfzig, Michelucci Dominique, and Sebti Foufou. Optimizations for Tensorial Bernstein-Based Solvers by Using Polyhedral Bounds. *International Journal of Shape Modeling*, 16(1-2):109–128, 2010.

[6] Christoph Fünfzig, Dominique Michelucci, and Sebti Foufou. Nonlinear systems solver in floating-point arithmetic using LP reduction. In Bronsvoort et al. [1], pages 123–134.

[7] Christoph Fünfzig, Dominique Michelucci, and Sebti Foufou. Polytope-based computation of polynomial ranges. In Shin et al. [20], pages 1247–1252.

[8] A. A. Gaganov. Computational complexity of the range of the polynomial in several variables. *Cybernetics*, pages 418–425, 1985.

[9] J. Garloff and A. P. Smith. Solution of systems of polynomial equation by using Bernstein expansion. In Goetz Alefeld, Jiri Rohn, Siegfried M. Rump, and Tetsuro Yamamoto, editors, *Symbolic Algebraic Methods and Verification Methods*, pages 87–97. Springer, 2001.

[10] Oded Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008.

[11] J. Hoscheck and D. Lasser. *Fundamentals of Computer Aided Geometric Design*. Wellesley, AKPeters, 1993.

[12] L. G. Khachian. A polynomial algorithm in linear programming. *Dokl. Akad. Nauk SSSR*, 244:1093–1096, 1979. English translation in Soviet Math. Dokl. 20, 191-194, 1979.

[13] B.H. Korte and J. Vygen. *Combinatorial optimization: theory and algorithms*, volume 21. Springer Verlag, 2008.

[14] Vladik Kreinovich, Anatoly Lakeyev, Jiri Rohn, and Patrick Kahl. *Computational complexity and feasibility of data processing and interval computations*. Kluwer, Dordrecht, 1997.

[15] Angelos Mantzaflaris, Bernard Mourrain, and Elias Tsigaridas. On continued fraction expansion of real roots of polynomial systems, complexity and condition numbers. *Theoretical Computer Science*, 412(22):2312–2330, 2011.

[16] D. Michelucci, S. Foufou, L. Lamarque, and P. Schreck. Geometric constraints solving: some tracks. In *ACM Symp. on Solid and Physical Modelling*, pages 185–196, 2006.

[17] Dominique Michelucci, Sebti Foufou, Loïc Lamarque, and David Ménegaux. Bernstein based arithmetic featuring de casteljau. In *Proc. of the Canadian Conference on Computational Geometry*, pages 215–218, 2005.

[18] Bernard Mourrain and Jean Pascal Pavone. Subdivision methods for solving polynomial equations. *J. Symb. Comput.*, 44(3):292–306, 2009.

[19] Martin Reuter, Tarjei S. Mikkelsen, Evan C. Sherbrooke, Takashi Maekawa, and Nicholas M. Patrikalakis. Solving nonlinear polynomial systems in the barycentric Bernstein basis. *Vis. Comput.*, 24(3):187–200, 2008.

[20] Sung Y. Shin, Sascha Ossowski, Michael Schumacher, Mathew J. Palakal, and Chih-Cheng Hung, editors. *Proceedings of the 2010 ACM Symposium on Applied Computing (SAC), Sierre, Switzerland, March 22-26, 2010*. ACM, 2010.

[21] Andrew Smith. Fast construction of constant bound functions for sparse polynomials. *Journal of Global Optimization*, 43:445–458, 2009.

[22] Thomas Sturm and Christoph Zengler, editors. *Automated Deduction in Geometry - 7th International Workshop, ADG 2008, Shanghai, China, September 22-24, 2008. Revised Papers*, volume 6301 of *Lecture Notes in Computer Science*. Springer, 2011.

[23] A. Tahari, Sebti Foufou, and Samy Ait-Aoudia. Bases tensorielles de Bernstein et solveurs. *Technique et Science Informatiques*, 29(6):629–663, 2010.

[24] B. Yamnitsky and L. A. Levin. An old linear programming algorithm runs in polynomial time. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pages 327–328. IEEE, 1982.