Monomial-wise Optimal Separable Underestimators for Mixed-Integer Polynomial Optimization

Christoph Buchheim · Claudia D'Ambrosio

Received: date / Accepted: date

Abstract In this paper we introduce a new method for solving box-constrained mixed-integer polynomial problems to global optimality. The approach, a specialized branch-and-bound algorithm, is based on the computation of lower bounds provided by the minimization of separable underestimators of the polynomial objective function. The underestimators are the novelty of the approach because the standard approaches in global optimization are based on convex relaxations. Thanks to the fact that only simple bound constraints are present, minimizing the separable underestimator can be done very quickly. The underestimators are computed monomial-wise after the original polynomial has been shifted. We show that such valid underestimators exist and their degree can be bounded when the degree of the polynomial objective function is bounded, too. For the quartic case, all optimal monomial underestimators are determined analytically.

We implemented and tested the branch-and-bound algorithm where these underestimators are hardcoded. The comparison against standard global optimization and polynomial optimization solvers clearly shows that our method outperforms the others, the only exception being the binary case where, though, it is still competitive. Moreover, our branch-and-bound approach suffers less in case of dense polynomial objective function, i.e., in case of polynomials having a large number of monomials.

This paper is an extended and revised version of the preliminary paper [4].

Keywords Polynomial Optimization · Mixed-Integer Nonlinear Programming · Separable Underestimators

C. Buchheim
Fakultät für Mathematik, TU Dortmund, Vogelpothsweg 87, 44227 Dortmund, Germany
E-mail: christoph.buchheim@tu-dortmund.de

C. D'Ambrosio
LIX CNRS (UMR7161), École Polytechnique, 91128 Palaiseau Cedex, France
E-mail: dambrosio@lix.polytechnique.fr

1 Introduction

In the last decade, the focus of the mathematical optimization community has increasingly moved towards mixed-integer non-linear programming (MINLP). The simultaneous presence of integrality constraints and non-linearities makes these problems harder than mixed integer linear programming (MILP) problems, especially when non-convexities come into play. In particular, MINLP problems are typically very challenging both from a theoretical and a practical viewpoint. As an example, minimizing a quadratic function subject to box-constraints is an NP-hard problem, both in the case of continuous and integer variables. Different types of convex relaxations and reformulations have been proposed in the literature for such problems. The resulting convex problems may be linear [3], semidefinite [22, 7] or other types of tractable problems [9, 8]. In fact, the standard way of dealing with general non-convex MINLPs is to relax them so as to obtain convex (often linear) relaxations. The relaxation results in a larger convex feasible set and a convex underestimator of the objective function. We refer to [12, 10, 2, 13] for an overview of standard methods for MINLPs.

Recently, Buchheim and Traversi [6] proposed to relax quadratic combinatorial optimization problems by replacing the objective function by separable non-convex underestimators instead of convex ones. In this paper, we propose a similar approach for a different class of problems, namely the minimization of polynomials subject to box and integrality constraints. More formally, we consider problems of the form

$$\begin{array}{rll} \min & f(x) & \\ \text{s.t.} & x \in [l,u] & \\ & x_i \in \mathbb{Z} & \forall i \in I\ , \end{array} \tag{MIPO}$$

where $l, u \in \mathbb{R}^n$ (with $l \leq u$ and $[l,u] := \prod_{i=1}^n [l_i, u_i]$), $f = \sum_{\alpha \in A} c_\alpha x^\alpha$ is an arbitrary polynomial of maximum degree $d \in \mathbb{N}$, and $I \subseteq \{1, \ldots, n\}$. We can assume without loss of generality that $l_i, u_i \in \mathbb{Z}$ for all $i \in I$.

Few specific methods exist for such kind of problem. In the case of binary variables, the standard approach is to add new variables replacing every monomial appearing in the problem and then linearize them. Another approach is to transform the polynomial problem into a quadratic one [18, 5]. When we have general integer variables, we can apply binary expansion to reduce the problem to the previous case. Clearly, the drawback is that a large number of variables is introduced and the approach usually is not practical.

In the mixed-integer case, the most widely used algorithms use a combination of convex underestimators and bound tightening techniques. Recently, Lasserre and Thanh [16] proposed to underestimate an arbitrary polynomial by a sequence of convex polynomials. The algorithms implemented in the commonly used MINLP solvers (see, for example, [21, 1, 19]) derive convex underestimators from the building blocks of the objective function.

In this paper, we propose a similar idea, i.e., we compute underestimators monomial-wise. However, our underestimators are non-convex but separable.

Even if we expect convex underestimator to be tighter than separables ones in general, a separable understimator can be minimized very quickly without needing to relax the integrality constraints. Thus, separable underestimators represent a good trade-off between approximation quality and tractability. These nice properties are confirmed by experimental results. Our approach performs very favorably against standard software for mixed-integer non-linear optimization such as BARON [21], Couenne [1], and SCIP [19]. We also compared our method against GloptiPoly [14], one of the best performing solvers for continuous polynomial optimization. It is based on hierarchies of semidefinite programming relaxations (see, e.g., [20, 15, 17]) and does not explicitly take into account integrality requirements. To circumvent this limitation, it is possible to add polynomial equations encoding integrality, but the resulting problems are intractable even for medium size instances. In the continuous instances, GloptiPoly performs better than in the mixed or integer ones but, in any case, the semidefinite programs used to underestimate the polynomial optimization problem take a long time to be solved.

In Section 2 the basic ideas of our approach are introduced. We then discuss the general branch-and-bound framework and its features (Section 3). In Section 4 we show extensive computational results that confirm the effectiveness of our approach. Finally, we conclude the paper and present some future directions in Section 5.

2 Separable Underestimators for Polynomials

In order to obtain a lower bound for Problem (MIPO), we aim at underestimating its objective function $f\colon \mathbb{R}^n \to \mathbb{R}$ of degree $d := \deg f$ by a separable polynomial $g(x) = \sum_{i=1}^n g_i(x_i)$, where all functions $g_i\colon \mathbb{R} \to \mathbb{R}$ are univariate polynomials of degree at most $\bar{d} := 2\left\lceil \frac{1}{2}d \right\rceil$. This is the smallest degree that ensures the existence of a global separable underestimator of f; see Section 2.2. Such underestimators are easy to minimize, as discussed in the next section.

2.1 Minimizing a Separable Polynomial

Concerning the minimization of a separable polynomial, we can immediately observe the following:

Lemma 1 *Let $g(x) = \sum_{i=1}^n g_i(x_i)$ be a separable polynomial. Then*

$$\begin{array}{rl} \min & g(x) \\ \text{s.t.} & x \in [l,u] \\ & x_i \in \mathbb{Z} \;\; \forall i \in I \end{array} \quad = \quad \sum_{i=1}^{n} \left\{ \begin{array}{rl} \min & g_i(x) \\ \text{s.t.} & x_i \in [l_i, u_i] \\ & x_i \in \mathbb{Z} \quad \text{if } i \in I\ . \end{array} \right.$$

The result above means that we can minimize the underestimator by solving n single univariate minimization problems involving the functions g_i separately. The problem is discrete if $i \in I$, continuous otherwise. It is important

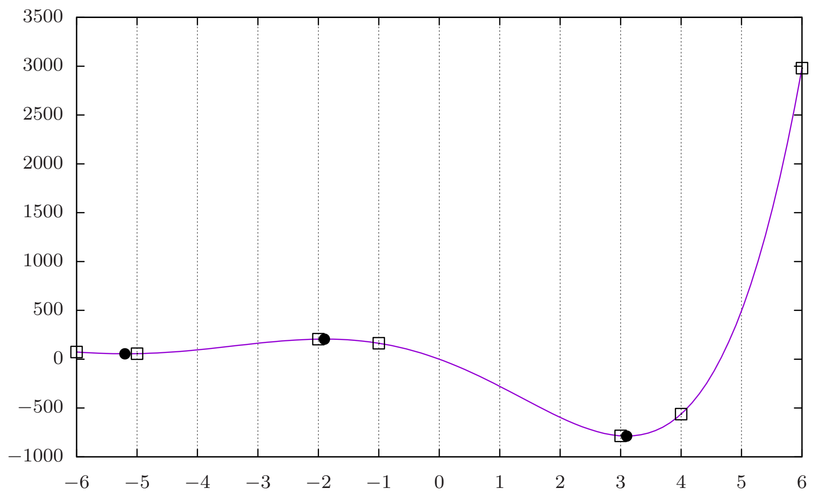

Fig. 1: A univariate polynomial of degree five, its stationary points (circles), and the candidates for being integer optimizers (squares).

to stress that we never relax the integrality requirements in our approach, thus we never change the feasible region. In particular, the union of the solutions of all univariate problems is a feasible solution of (MIPO). An upper bound can thus be obtained by re-evaluating this solution with respect to the original polynomial function f.

To guarantee feasibility, the two types of problems (integer or continuous) are solved in a slightly different way. Let us start with the solution of the continuous univariate problem ($i \notin I$). The minimizer of g_i over $[l_i, u_i]$ must either satisfy the first order condition $g_i'(x) = 0$ or be on the boundary. Recall that g_i' is a polynomial of degree at most $\bar{d} - 1$. Thus, it has at most $\bar{d} - 1$ real zeroes and they can be computed by closed formulae if $\bar{d} \leq 5$, numerically otherwise. This means that we have at most $\bar{d}+1$ candidates for the minimizer of g_i and that we can identify the minimizer by enumeration. When $i \in I$, i.e., the univariate problem has to satisfy integrality requirements, the integer minimizer must be a continuous minimizer of g_i rounded up or down. Thus, we start by identifying the continuous candidates as explained above, round them up and down, and end up with at most $2\bar{d}$ many candidates. See Figure 1 for an illustration.

In all cases, we thus obtain a minimizer x_i^* for g_i, such that x^* is a mixed-integer minimizer of g by Lemma 1. We thus obtain $g(x^*)$ as lower bound and $f(x^*)$ as upper bound on (MIPO). Obviously, the objective is to achieve a small distance $f(x^*) - g(x^*)$, i.e., to compute a tight underestimator g. In the next section, we first argue that separable global underestimators exist.

2.2 Global Underestimators for Arbitrary Degree

Since the feasible region $[l, u]$ of Problem (MIPO) is compact, it is clear that separable – even constant – underestimators for any polynomial f over $[l, u]$ exist. We now show that each polynomial f of degree d even admits a *global* separable underestimator of degree $\bar{d}$. For this, it suffices to underestimate each monomial of f separately.

Theorem 1 *Let $f = cx^\alpha$ be a monomial of even degree d. Then*

$$f(x) \geq -\frac{|c|}{d} \sum_{i=1}^{n} \alpha_i x_i^d$$

for all $x \in \mathbb{R}^n$.

Proof As d is even, we obtain

$$|x^\alpha| = \prod_{i=1}^{n} |x_i|^{\alpha_i} = \Big(\prod_{i=1}^{n} (x_i^d)^{\alpha_i} \Big)^{1/d} \leq \frac{1}{d} \sum_{i=1}^{n} \alpha_i x_i^d$$

by the inequality of arithmetic and geometric means. Hence

$$f(x) = cx^\alpha \geq -|c||x^\alpha| \geq -\frac{|c|}{d} \sum_{i=1}^{n} \alpha_i x_i^d$$

for all $x \in \mathbb{R}^n$. □

Theorem 2 *Let $f = cx^\alpha$ be a monomial of odd degree d. Then*

$$f(x) \geq -\frac{|c|}{2d} \sum_{i=1}^{n} \alpha_i (x_i^{d+1} + x_i^{d-1})$$

for all $x \in \mathbb{R}^n$.

Proof As $d-1$ and $d+1$ are even, we obtain

$$|x^\alpha| = \prod_{i=1}^{n} |x_i|^{\alpha_i} = \Big(\prod_{i=1}^{n} (x_i^{d-1} x_i^{d+1})^{\alpha_i} \Big)^{1/(2d)} \leq \frac{1}{2d} \sum_{i=1}^{n} \alpha_i (x_i^{d-1} + x_i^{d+1})$$

by the inequality of arithmetic and geometric means. Hence

$$f(x) = cx^\alpha \geq -|c||x^\alpha| \geq -\frac{|c|}{2d} \sum_{i=1}^{n} \alpha_i (x_i^{d-1} + x_i^{d+1})$$

for all $x \in \mathbb{R}^n$. □

2.3 Computing Tight Separable Underestimators

Now that we know that a global separable underestimator exists and how to minimize it, we discuss how to obtain a best possible one, i.e., one that yields the tightest lower bound. Ideally, we would like to solve the following problem:

$$\begin{array}{rl} \max & \displaystyle\min_{\substack{x\in[l,u]\\ x_i\in\mathbb{Z} \text{ for } i\in I}} \quad g(x) \\ \text{s.t.} & g(x) \le f(x) \quad \forall x \in [l,u] \\ & g \text{ separable polynomial of degree at most } \bar{d}. \end{array}$$

It is evident that solving the problem above is too difficult as even testing whether $g = 0$ is a valid underestimator for f would imply to decide non-negativity of f, which is well-know to be an intractable problem for general f.

We thus simplify the problem by considering each monomial separately, i.e., by computing the optimal separable underestimator for each monomial, and obtain a global separable underestimator for the polynomial by summing up each of them. The resulting underestimator will not be optimal for f, but we hope that the provided lower bounds are still reasonably tight (as confirmed by the experimental results presented in Section 4).

When summing up monomial-wise underestimators, it is not useful any more to consider the above objective function

$$\min_{\substack{x\in[l,u]\\ x_i\in\mathbb{Z} \text{ for } i\in I}} \quad g(x) \ ,$$

as each monomial underestimator may have a different minimizer. With this in mind, we now aim at finding an underestimator that is as close as possible to f on average on the feasible region $[l,u]$. Hence we replace the objective function above by

$$\int_{[l,u]} g(x)\,\mathrm{d}x \ .$$

In summary, we obtain the problem

$$\begin{array}{rl} \max & \int_{[l,u]} g(x)\,\mathrm{d}x \\ \text{s.t.} & g(x) \le f(x) \quad \forall x \in [l,u] \\ & g \text{ separable polynomial of degree at most } \bar{d}, \end{array} \tag{P}$$

where the variables of the problem are the coefficients of the univariate polynomials g_i. We can define a variable c_0 collecting all the constant terms of all g_i and rewrite the separable underestimator as

$$g(x) = \sum_{i=1}^{n} \sum_{j=1}^{\bar{d}} c_{ij} x_i^j + c_0.$$

Problem (P) has $n\bar{d}+1$ variables, i.e., the coefficients c_{ij} and c_0. We then have a linear objective function, as

$$\int_{[l,u]} g(x)\,\mathrm{d}x = \sum_{i=1}^{n}\sum_{j=1}^{\bar{d}} c_{ij} \int_{[l,u]} x_i^j\,\mathrm{d}x + c_0 \int_{[l,u]} 1\,\mathrm{d}x\ .$$

As f is fixed, Problem (P) can thus be modeled as a linear optimization problem over the cone of non-negative polynomials. The dual problem can be calculated as

$$\begin{array}{ll} \min & \int f\,\mathrm{d}\mu \\ \text{s.t.} & \int x_i^j\,\mathrm{d}\mu = \int_{[l,u]} x_i^j\,\mathrm{d}x \text{ for } i=1,\dots,n,\ j=1,\dots,\bar{d} \\ & \int 1\,\mathrm{d}\mu = \int_{[l,u]} 1\,\mathrm{d}x \\ & \mu \text{ Borel-measure with support on } [l,u]. \end{array} \tag{D}$$

The global underestimators introduced in Section 2.2 do not constitute optimal solutions of Problem (P) in general, even if f is a monomial. In order to determine optimal monomial underestimators, the first step is to translate the problem such that the origin becomes the center of the feasible region, i.e., we obtain a symmetric box. We can calculate the shifted polynomial $\bar{f}$ by $\bar{f}(x) := f(x+t)$, where $t_i := \frac{1}{2}(l_i+u_i)$ for all $i = 1,\dots,n$. The resulting shifted polynomial has the same degree as f but a different number of monomials. These monomials are considered separately and a Problem (P) is solved for each of them over the new box $[l-t, u-t]$. The separable underestimator of $\bar{f}$, say $\bar{g}$, is obtained by summing up all monomial understimators. We finally get the separable underestimator for f by shifting back g, using $g(x) := \bar{g}(x-t)$.

By shifting we thus get a box $[l-t, u-t]$ where $l-t = -(u-t)$ by definition of t. Consequently, we can assume that the feasible set in Problem (P) is symmetric with respect to the origin, i.e., that $l = -u$. As f is a monomial in our approach, we are allowed to scale the feasible box to normalize further:

Lemma 2 *Let $u > 0$ and let*

$$\sum_{i=1}^{n}\sum_{j=1}^{\bar{d}} c_{ij} x_i^j + c_0$$

be an optimal solution to (P) *over $[-1,1]^n$ for a monomial $f = cx^\alpha$ with $c \neq 0$. Then*

$$|\tfrac{1}{c} f(u)| \cdot \Big(\sum_{i=1}^{n}\sum_{j=1}^{\bar{d}} \frac{c_{ij}}{u_i^j} x_i^j + c_0 \Big)$$

is an optimal solution to (P) *over $[-u,u]$.*

The proof is straightforward. Note that we can assume $u > 0$ without loss of generality: if $u_i = 0$ for some i, the corresponding variable x_i can be eliminated. Thus the above lemma is always applicable.

It is a crucial observation for our approach that Problem (P), when applied to single monomials, only involves at most $\bar{d}$ many different variables. This means that when we fix a bound on the degree of f (and thus of g) we have to solve only a fixed number of different problems of type (P), namely those for $f(x) := x^\alpha$ and $f(x) := -x^\alpha$ over $[-1,1]^{\bar{d}}$ for each possible monomial x^α with $|\alpha| \leq \bar{d}$.

This implies that we do not need to compute the best underestimators of each monomial at each node of the branch-and-bound tree but it suffices to solve Problem (P) offline once for each relevant monomial and to hardcode the results. Then online we just need to replace each monomial in $\bar{f}$ by its previously computed optimal underestimator. The steps to perform to get the underestimator of f are just shift, scale, replace, sum up the single underestimators, and scale and shift back. If the total degree of f is bounded, the computation of the underestimator can thus be performed in time linear in the number of monomials. Note that the number of monomials after shifting is linear in the original number of monomials when the degree d is fixed, but the factor grows exponentially in d.

Summarizing the discussion so far, for a given maximum degree d, it thus remains to solve Problem (P) for each monomial f of degree at most $\bar{d}$ over $[-1,1]$. An important tool for this is the following result:

Lemma 3 (Complementary Slackness) *Let g be feasible for* (P) *and let μ be feasible for* (D). *If $\int (f-g)\,\mathrm{d}\mu = 0$, then g is optimal for* (P).

We will use this observation in the next section in order to determine all optimal monomial underestimators for degree $d \leq 4$.

2.4 Optimal Underestimators for Quartic Monomials

In the following, we focus on monomials of degree at most four and determine their optimal underestimators according to (P). For the proofs we use Lemma 3, where a corresponding feasible solution for the dual problem (D) is constructed explicitly.

Up to symmetry, 12 monomials have to be considered but, because the cases of positive and negative coefficients have to be distinguished, we end up with 24 cases, namely the monomials $\pm x_1^k$ (for $k = 0, \ldots, 4$), $\pm x_1 x_2$, $\pm x_1^2 x_2$, $\pm x_1 x_2 x_3$, $\pm x_1^3 x_2$, $\pm x_1^2 x_2^2$, $\pm x_1^2 x_2 x_3$, and $\pm x_1 x_2 x_3 x_4$. The first ten are already separable and hence trivial. The following result covers a further nine cases:

Theorem 3 *Let $\alpha \in \mathbb{N}_0^n$ such that $d := \sum_{i=1}^n \alpha_i$ is even and $d \leq 4$. Assume that at least one α_i is odd, or that $c \leq 0$. Then an optimal solution of* (P) *for $f(x) := cx^\alpha$ over $[-1,1]^n$ is*

$$g(x) := -\frac{|c|}{d} \sum_{i=1}^n \alpha_i x_i^d \; .$$

Proof First note that validity of g follows from Theorem 1. To prove optimality, first assume that there exists some r with α_r odd. Define a Borel measure μ implicitly by requiring

$$\int h \,\mathrm{d}\mu = 2^{n-1} \int_{-1}^{1} h(t\tilde{e}) \,\mathrm{d}t \ ,$$

where $\tilde{e}_r = -\mathrm{sgn}\, c$ and $\tilde{e}_i = 1$ for $i \neq r$. Then

$$\int 1 \,\mathrm{d}\mu = 2^n = \int_{[-1,1]} 1 \,\mathrm{d}x \ .$$

Moreover, if j is even,

$$\int x_i^j \,\mathrm{d}\mu = 2^{n-1} \int_{-1}^{1} (t\tilde{e}_i)^j \,\mathrm{d}t = \frac{1}{j+1} 2^n = \int_{[-1,1]} x_i^j \,\mathrm{d}x \ ,$$

while for j odd

$$\int x_i^j \,\mathrm{d}\mu = 2^{n-1} \int_{-1}^{1} (t\tilde{e}_i)^j \,\mathrm{d}t = 0 = \int_{[-1,1]} x_i^j \,\mathrm{d}x \ .$$

Hence μ is feasible for the dual problem (D). Finally, we have

$$\begin{aligned} \int (f-g) \,\mathrm{d}\mu &= 2^{n-1} \int_{-1}^{1} \Big(ct^d \prod_k \tilde{e}_k^{\alpha_k} + \frac{|c|}{d} \sum_k \alpha_k t^d \tilde{e}_k^d \Big) \,\mathrm{d}t \\ &= 2^{n-1} \int_{-1}^{1} \Big(-|c|t^d + |c|t^d \frac{1}{d} \sum_k \alpha_k \Big) \,\mathrm{d}t \\ &= 0 \ , \end{aligned}$$

as d is even and $\sum_k \alpha_k = d$. Hence the underestimator g is optimal by Lemma 3. If all α_i are even but $c < 0$, the proof is analogous. □

From the above proof, it follows that the computed underestimators remain optimal for any $\bar{d} \geq d$ and that these optimal underestimators are independent of the domain $[-u, u]$, as $f - g$ turns out to be globally non-negative. Next, we consider the only remaining even degree case for $\bar{d} \leq 4$.

Theorem 4 *Let $c \geq 0$. Then an optimal solution of* (P) *for $f(x) := cx_1^2 x_2^2$ over $[-1,1]^n$ for $\bar{d} = 4$ is*

$$g(x) = -\tfrac{1}{2}cx_1^4 + \tfrac{2}{3}cx_1^2 - \tfrac{1}{2}cx_2^4 + \tfrac{2}{3}cx_2^2 - \tfrac{2}{9}c \ .$$

Proof By scaling, we may assume $c = 1$. The function g is a (global) underestimator of f as

$$f(x) - g(x) = \tfrac{1}{2}\Big(x_1^2 + x_2^2 - \tfrac{2}{3}\Big)^2 \geq 0$$

for all $x \in \mathbb{R}^2$. Consider the discrete Borel measure

$$\mu := \frac{1}{5}\Big(\delta_{\binom{\sqrt{1/3}}{\sqrt{1/3}}} + \delta_{\binom{\sqrt{1/3}}{-\sqrt{1/3}}} + \delta_{\binom{-\sqrt{1/3}}{\sqrt{1/3}}} + \delta_{\binom{-\sqrt{1/3}}{-\sqrt{1/3}}}\Big) + \frac{4}{5}\Big(\delta_{\binom{\sqrt{2/3}}{0}} + \delta_{\binom{-\sqrt{2/3}}{0}} + \delta_{\binom{0}{\sqrt{2/3}}} + \delta_{\binom{0}{-\sqrt{2/3}}}\Big)\,.$$

Then

$$\begin{aligned}\int 1 \,\mathrm{d}\mu &= \frac{4}{5} + \frac{16}{5} = 4\\ \int x_i^j \,\mathrm{d}\mu &= 0 \quad \forall j \text{ odd}\\ \int x_i^2 \,\mathrm{d}\mu &= \frac{1}{5}\big(\frac{4}{3}\big) + \frac{4}{5}\big(\frac{4}{3}\big) = \frac{4}{3}\\ \int x_i^4 \,\mathrm{d}\mu &= \frac{1}{5}\big(\frac{4}{9}\big) + \frac{4}{5}\big(\frac{8}{9}\big) = \frac{4}{5}\,,\end{aligned}$$

this implies that μ is feasible for (D). Since $\int (f-g)\,\mathrm{d}\mu = 0$, the result follows from Lemma 3. □

See Figure 2 for an illustration. In this case, the proof again shows optimality independently of the domain, but not of the degree $\bar{d}$. In fact, when allowing degree $\bar{d} = 6$, a better underestimator can be computed.

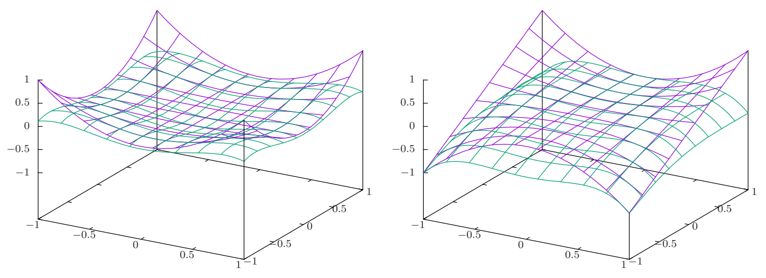

Fig. 2: Optimal separable underestimators of degree $\bar{d} \le 4$ for $f(x) = x_1^2 x_2^2$ (left) and $f(x) = x_1^2 x_2$ (right).

The following results complete the quartic case.

Theorem 5 *An optimal solution of* (P) *for* $f(x) := c x_1 x_2 x_3$ *over* $[-1,1]^n$ *is*

$$g(x) = -\tfrac{|c|}{6}\Big(\sqrt{\tfrac{5}{3}}(x_1^4 + x_2^4 + x_3^4) + \sqrt{\tfrac{3}{5}}(x_1^2 + x_2^2 + x_3^2)\Big)\,.$$

Proof By symmetry and scaling, we may assume $c = 1$. The function g is a (global) underestimator of f, as

$$\begin{aligned} f(x) - g(x) &= x_1x_2x_3 + \tfrac{1}{6}\Big(\sqrt{\tfrac{5}{3}}(x_1^4 + x_2^4 + x_3^4) + \sqrt{\tfrac{3}{5}}(x_1^2 + x_2^2 + x_3^2)\Big) \\ &= \tfrac{1}{18}\sqrt{15}\big((x_1x_2 + \tfrac{1}{5}\sqrt{15}x_3)^2 + (x_1x_3 + \tfrac{1}{5}\sqrt{15}x_2)^2 \\ &\quad +(x_2x_3 + \tfrac{1}{5}\sqrt{15}x_1)^2\big) + \tfrac{1}{72}\sqrt{15}(x_1^2 + x_2^2 - 2x_3^2)^2 \\ &\quad +\tfrac{1}{24}\sqrt{15}(x_1^2 - x_2^2)^2 \ . \end{aligned}$$

Consider the discrete Borel measure

$$\mu := \frac{32}{9}\delta_{\left(\begin{smallmatrix}0\\0\\0\end{smallmatrix}\right)} + \frac{10}{9}\Big(\delta_{\left(\begin{smallmatrix}-\sqrt{3/5}\\\sqrt{3/5}\\\sqrt{3/5}\end{smallmatrix}\right)} + \delta_{\left(\begin{smallmatrix}\sqrt{3/5}\\-\sqrt{3/5}\\\sqrt{3/5}\end{smallmatrix}\right)} + \delta_{\left(\begin{smallmatrix}\sqrt{3/5}\\\sqrt{3/5}\\-\sqrt{3/5}\end{smallmatrix}\right)} + \delta_{\left(\begin{smallmatrix}-\sqrt{3/5}\\-\sqrt{3/5}\\-\sqrt{3/5}\end{smallmatrix}\right)}\Big) \ .$$

Then

$$\begin{aligned} \int 1 \,\mathrm{d}\mu &= \frac{32}{9} + 4 \cdot \frac{10}{9} = 8 \\ \int x_i^j \,\mathrm{d}\mu &= 0 \quad \forall j \text{ odd} \\ \int x_i^2 \,\mathrm{d}\mu &= \frac{10}{9} \cdot 4 \cdot \frac{3}{5} = \frac{8}{3} \\ \int x_i^4 \,\mathrm{d}\mu &= \frac{10}{9} \cdot 4 \cdot \frac{9}{25} = \frac{8}{5} \ , \end{aligned}$$

this implies that μ is feasible for (D). Since $\int (f - g)\,\mathrm{d}\mu = 0$, the result follows from Lemma 3. □

Theorem 6 *An optimal solution of* (P) *for $f(x) := cx_1^2x_2$ over $[-1, 1]^n$ has the form*

$$g(x) = |c|c_{14}x_1^4 + |c|c_{12}x_1^2 + |c|c_{24}x_2^4 + cc_{23}x_2^3 + |c|c_{22}x_2^2 + cc_{21}x_2 + |c|c_0$$

with

$$\begin{aligned} c_{14} &\approx -0.820950196623141 & c_{24} &\approx +0.052169786129554 \\ c_{12} &\approx +0.472574632153429 & c_{23} &\approx +0.057375067121109 \\ & & c_{22} &\approx -0.311590671084123 \\ c_0 &\approx -0.070759806839502 & c_{21} &\approx +0.272799782607049 \ . \end{aligned}$$

Proof By symmetry and scaling, we may assume $c = 1$ and hence $f(x) = x_1^2x_2$. We only sketch the proof, details can be checked with the help of computer algebra packages. Let δ be the third-smallest real zero of

$$\begin{aligned} &30625z^{10} - 98750z^9 - 366375z^8 - 290700z^7 + 43050z^6 + 145280z^5 \\ &+50334z^4 - 4148z^3 - 4259z^2 - 322z + 49 \ , \end{aligned}$$

moreover define $\gamma = \frac{1}{5\delta}$. Let φ be the smaller zero of the quadratic polynomial in z

$$(250\gamma^4 + 400\gamma^3 + 300\gamma^2 + 280\gamma + 50)z^2 + (-200\gamma - 40)z - 35\gamma^2 + 34\gamma + 7\ .$$

Finally, set $\alpha := \frac{2(5\gamma+1)(\gamma-1+2\varphi)}{5\gamma^2-2\gamma-1}$ and $\beta := \varphi(5\gamma^2+1)(\gamma+1)$. Numerically, we have

$$\delta \approx 0.63445,\ \gamma \approx 0.31523,\ \alpha \approx 0.89688,\ \beta \approx 0.47981\ .$$

Now define

$$\begin{aligned}
c_{14} &= \frac{1}{2} \cdot \frac{\gamma - 1}{\alpha - \beta} \\
c_{12} &= \frac{\beta - \alpha\gamma}{\alpha - \beta} \\
c_{24} &= -\frac{1}{4} \cdot \frac{\alpha^2}{(\delta+1)^2(\alpha-\beta)} + \frac{1}{2} \cdot \frac{\beta^2}{(\gamma+1)(\gamma+\delta)^2(\alpha-\beta)} \\
c_{23} &= -\frac{1}{4} \cdot \frac{\alpha^2(\gamma-1-2\delta)}{(\delta+1)^2(\alpha-\beta)} - \frac{\beta^2\delta}{(\gamma+1)(\gamma+\delta)^2(\alpha-\beta)} \\
c_{22} &= \frac{1}{4} \cdot \frac{\alpha^2(2\gamma\delta+\gamma-2\delta-\delta^2)}{(\delta+1)^2(\alpha-\beta)} + \frac{1}{2} \cdot \frac{\beta^2(\delta-1)(\delta+1)}{(\gamma+1)(\gamma+\delta)^2(\alpha-\beta)} \\
c_{21} &= -\frac{1}{4} \cdot \frac{\alpha^2\delta(2\gamma+\delta\gamma-\delta)}{(\delta+1)^2(\alpha-\beta)} + \frac{\beta^2\delta}{(\gamma+1)(\gamma+\delta)^2(\alpha-\beta)} \\
c_0 &= \frac{1}{4} \cdot \frac{\alpha^2\gamma\delta^2}{(\delta+1)^2(\alpha-\beta)} - \frac{1}{2} \cdot \frac{\beta^2\delta^2}{(\gamma+1)(\gamma+\delta)^2(\alpha-\beta)}\ .
\end{aligned}$$

It can be shown that, with these definitions, $(f-g)(x) = \sigma_1 + \sigma_2(1-x_2)$, where σ_1 is a sum-of-squares of degree 8 and σ_2 is a sum-of-squares of degree 6. Hence $f-g$ is non-negative on $[-1,1]^n$ and thus g is feasible for (P).

We next construct a dual solution as follows: the measure μ is defined as

$$\mu = \mu_1\delta_{p_1} + \mu_2\delta_{p_2} + \mu_3(\delta_{p_3^+} + \delta_{p_3^-}) + \mu_4(\delta_{p_4^+} + \delta_{p_4^-})$$

with

$$p_1 = (0,1), p_2 = (0,\delta), p_3^\pm = (\pm\sqrt{\alpha}, -1), p_4^\pm(\pm\sqrt{\beta}, -\gamma)$$

and

$$\begin{aligned}
\mu_1 &= \frac{1000}{3} \frac{\gamma^4}{(5\gamma-1)(5\gamma^2+1)(5\gamma+1)} \approx 1.48151 \\
\mu_2 &= \frac{2}{3} \frac{5\gamma^2-2\gamma-1}{(5\gamma+1)(\gamma-1)} \approx 0.42841 \\
\mu_3 &= -\frac{8}{3} \frac{1}{(\gamma-1)(\gamma+1)(5\gamma^2+1)} \approx 1.97808 \\
\mu_4 &= \frac{2}{3} \frac{5\gamma^2+2\gamma-1}{(\gamma+1)(5\gamma-1)} \approx 0.11201\ .
\end{aligned}$$

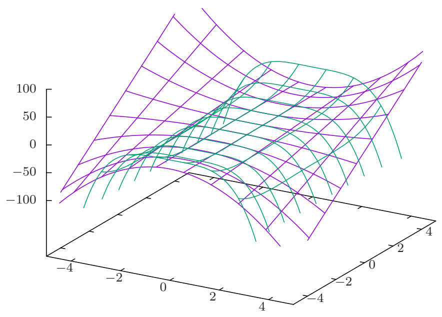

Fig. 3: The optimal separable underestimator for $f(x) = x_1^2 x_2$ of degree $\bar{d} \leq 4$ is not globally feasible.

This measure is dual feasible, as all points p_i belong to $[-1,1]^n$ and all dual constraints are satisfied. It thus suffices to show that f agrees with g on all points p_i, the result then follows from Lemma 3 again. □

Unlike in all other quartic cases, the optimal local underestimator in Theorem 6 is not valid globally, but requires $x_2 \in [-1,1]$; see Figure 3. In Table 1, we list all optimal underestimators of degree at most four for quartic monomials.

Note that the global underestimators devised in Section 2.2 are optimal in the case $d = 4$ if at least one α_i is odd or $c \leq 0$, as shown in Theorem 3. In fact, the proof of Theorem 3 clearly carries over to arbitrary even degrees with at least one α_i odd or $c \leq 0$. On the other hand, by Theorems 5 and 6, the underestimator of Section 2.2 is not optimal in the case $d = 3$.

2.5 Optimal Underestimators for Quartic Monomials over Integer Variables

When computing optimal underestimators according to (P), we do not take into account any given integrality constraints on the variables; these are considered only when minimizing the resulting underestimator. In theory, the underestimators could be improved by replacing (P) with the linear program

$$\begin{array}{rl} \max & \frac{1}{|F|} \sum_{x \in F} g(x) \\ \text{s.t.} & g(x) \leq f(x) \quad \forall x \in F \\ & g \text{ separable polynomial of degree at most } \bar{d}, \end{array} \tag{P'}$$

where F is the finite set of integer feasible solutions. This replacement affects both the constraints, which are weaker than in (P), and the objective function, which now only takes the integer solutions into account when measuring the quality of an underestimator.

	$+$	$-$
± 1	1	-1
$\pm x_1$	x_1	$-x_1$
$\pm x_1 x_2$	$-\frac{1}{2}(x_1^2 + x_2^2)$	$-\frac{1}{2}(x_1^2 + x_2^2)$
$\pm x_1^2$	x_1^2	$-x_1^2$
$\pm x_1 x_2 x_3$	$-\frac{1}{6}\left(\sqrt{\frac{5}{3}}(x_1^4 + x_2^4 + x_3^4) + \sqrt{\frac{3}{5}}(x_1^2 + x_2^2 + x_3^2)\right)$	$-\frac{1}{6}\left(\sqrt{\frac{5}{3}}(x_1^4 + x_2^4 + x_3^4) + \sqrt{\frac{3}{5}}(x_1^2 + x_2^2 + x_3^2)\right)$
$\pm x_1^2 x_2$	$\approx -0.82x_1^4 + 0.47x_1^2 + 0.052x_2^4 + 0.057x_2^3 - 0.31x_2^2 + 0.27x_2 - 0.071$	$\approx -0.82x_1^4 + 0.47x_1^2 + 0.052x_2^4 - 0.057x_2^3 - 0.31x_2^2 - 0.27x_2 - 0.071$
$\pm x_1^3$	x_1^3	$-x_1^3$
$\pm x_1 x_2 x_3 x_4$	$-\frac{1}{4}(x_1^4 + x_2^4 + x_3^4 + x_4^4)$	$-\frac{1}{4}(x_1^4 + x_2^4 + x_3^4 + x_4^4)$
$\pm x_1^2 x_2 x_3$	$-\frac{1}{4}(2x_1^4 + x_2^4 + x_3^4)$	$-\frac{1}{4}(2x_1^4 + x_2^4 + x_3^4)$
$\pm x_1^2 x_2^2$	$-\frac{1}{2}x_1^4 + \frac{2}{3}x_1^2 - \frac{1}{2}x_2^4 + \frac{2}{3}x_2^2 - \frac{2}{9}$	$-\frac{1}{2}(x_1^4 + x_2^4)$
$\pm x_1^3 x_2$	$-\frac{1}{4}(3x_1^4 + x_2^4)$	$-\frac{1}{4}(3x_1^4 + x_2^4)$
$\pm x_1^4$	x_1^4	$-x_1^4$

Table 1: Optimal separable underestimators for monomials of degree $d \leq 4$ for $\bar{d} \leq 4$.

As F is exponential in general, even if the bounds on the integer variables are fixed, this problem cannot necessarily be solved in polynomial time. Nevertheless, we could stick to our approach proposed above and consider monomials independently, computing optimal underestimators offline. However, as a consequence of branching, we also need to consider all non-trivial contiguous subdomains for all appearing variables. By this, the number of underestimators that have to be computed and hardcoded grows quickly with the domain sizes of integer variables. At the same time, for large domains it is likely that the optimal solution of (P') does not differ significantly from the optimal solution of (P).

In the following, we thus restrict ourselves to the binary and ternary case. After translation and scaling, the two domains in question are thus $\{-1, 1\}$ and $\{-1, 0, 1\}$. In the respective cases, we may exploit that

$$x_i^2 = 1 \text{ if } x_i \text{ is binary, } x_i^3 = x_i \text{ if } x_i \text{ is ternary.} \qquad (*)$$

In particular, the optimal underestimators for binary or ternary problems can be assumed to be linear or quadratic, respectively. Again considering the quartic case, we thus need to take into account all 24 monomials as before, but for each of those we need to consider each combination of binary and ternary variables, which means up to 16 combinations. However, using symmetry and (*), only 44 cases have to be considered, as listed in Table 2. Note that Problem (P') has many optimal solutions in general; in the table we prefer symmetric solutions over sparse ones. We also state by how much the given optimal solution of (P') improves the optimal solution of (P) with respect to the objective function of (P'), both in absolute terms and in terms of the gap closed with respect to the average value of the given monomial on the feasible set F.

It turns out that the weaker constraints in (P') compared to (P) only have a minor effect on the optimal underestimator. One of the few cases where the optimal solution of (P') is infeasible for (P) is x_1x_2 for binary x_1 and ternary x_2: for $(x_1, x_2) = (-1, \frac{1}{2})$, we have $x_1x_2 \not\geq -x_2^2$; see Figure 4. The change in the objective function has a slightly stronger effect, but altogether the improvements are negligable; in the pure binary case, the maximal percentage of gap closed is about 3 %. We can thus conclude that using optimal solutions according to (P) is a reasonable approach also for integer variables with small domains.

3 Branch-and-Bound Algorithm

We propose to solve (MIPO) by a branch-and-bound algorithm in which the separable underestimators presented in Section 2 are used to compute lower bounds. We branch by splitting up the domain of a variable and generate two subproblems. The aim of branching is to make the feasible box smaller and to improve the bound given by the separable underestimator. This can be achieved either because the separable underestimator changes or just because the feasible set is smaller.

monomial	binary/ternary				+			−		
	x_1	x_2	x_3	x_4	optimal in (P')	improvement		optimal in (P')	improvement	
						absolute	gap		absolute	gap
± 1					1	—	—	-1	—	—
$\pm x_1$	B				x_1	—	—	$-x_1$	—	—
	T				x_1	—	—	$-x_1$	—	—
$\pm x_1 x_2$	B	B			-1	—	—	-1	—	—
	B	T			$-x_2^2$	$\frac{1}{6}$	$20\,\%$	$-x_2^2$	$\frac{1}{6}$	$20\,\%$
	T	T			$-\frac{1}{2}(x_1^2 + x_2^2)$	—	—	$-\frac{1}{2}(x_1^2 + x_2^2)$	—	—
$\pm x_1^2$	T				x_1^2	—	—	$-x_1^2$	—	—
$\pm x_1 x_2 x_3$	B	B	B		-1	$-1 + \frac{4}{\sqrt{15}}$	$\approx 3\,\%$	-1	$-1 + \frac{4}{\sqrt{15}}$	$\approx 3\,\%$
	B	B	T		$-x_3^2$	$-\frac{2}{3} + \frac{32}{9\sqrt{15}}$	$\approx 27\,\%$	$-x_3^2$	$-\frac{2}{3} + \frac{32}{9\sqrt{15}}$	$\approx 27\,\%$
	B	T	T		$-\frac{1}{2}(x_2^2 + x_3^2)$	$-\frac{2}{3} + \frac{28}{9\sqrt{15}}$	$\approx 17\,\%$	$-\frac{1}{2}(x_2^2 + x_3^2)$	$-\frac{2}{3} + \frac{28}{9\sqrt{15}}$	$\approx 17\,\%$
	T	T	T		$-\frac{1}{3}(x_1^2 + x_2^2 + x_3^2)$	$-\frac{2}{3} + \frac{8}{3\sqrt{15}}$	$\approx 3\,\%$	$-\frac{1}{3}(x_1^2 + x_2^2 + x_3^2)$	$-\frac{2}{3} + \frac{8}{3\sqrt{15}}$	$\approx 3\,\%$
$\pm x_1^2 x_2$	T	B			$x_1^2 + x_2 - 1$	≈ 0.23	$\approx 41\,\%$	$x_1^2 - x_2 - 1$	≈ 0.23	$\approx 41\,\%$
	T	T			$-\frac{1}{2}(x_2^2 - x_2)$	≈ 0.14	$\approx 30\,\%$	$-\frac{1}{2}(x_2^2 + x_2)$	≈ 0.14	$\approx 30\,\%$
$\pm x_1 x_2 x_3 x_4$	B	B	B	B	-1	—	—	-1	—	—
	B	B	B	T	$-x_4^2$	$\frac{1}{4}$	$\approx 27\,\%$	$-x_4^2$	$\frac{1}{4}$	$\approx 27\,\%$
	B	B	T	T	$-\frac{1}{2}(x_3^2 + x_4^2)$	$\frac{1}{6}$	$20\,\%$	$-\frac{1}{2}(x_3^2 + x_4^2)$	$\frac{1}{6}$	$20\,\%$
	B	T	T	T	$-\frac{1}{3}(x_2^2 + x_3^2 + x_4^2)$	$\frac{1}{12}$	$\approx 11\,\%$	$-\frac{1}{3}(x_2^2 + x_3^2 + x_4^2)$	$\frac{1}{12}$	$\approx 11\,\%$
	T	T	T	T	$-\frac{1}{4}(x_1^2 + x_2^2 + x_3^2 + x_4^2)$	—	—	$-\frac{1}{4}(x_1^2 + x_2^2 + x_3^2 + x_4^2)$	—	—
$\pm x_1^2 x_2 x_3$	T	B	B		$-x_1^2$	$\frac{1}{6}$	$20\,\%$	$-x_1^2$	$\frac{1}{6}$	$20\,\%$
	T	B	T		$-x_3^2$	$\frac{1}{12}$	$\approx 11\,\%$	$-x_3^2$	$\frac{1}{12}$	$\approx 11\,\%$
	T	T	T		$-\frac{1}{2}(x_2^2 + x_3^2)$	—	—	$-\frac{1}{2}(x_2^2 + x_3^2)$	—	—
$\pm x_1^2 x_2^2$	T	T			$x_1^2 + x_2^2 - 1$	$\frac{1}{3}$	$75\,\%$	$-\frac{1}{2}(x_1^2 + x_2^2)$	—	—

Table 2: Optimal separable underestimators for monomials of degree $d \leq 4$, binary and ternary variables

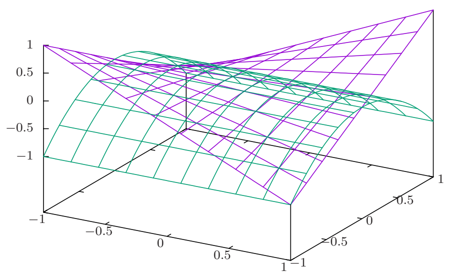

Fig. 4: The optimal separable underestimator for $f(x) = x_1x_2$ for binary x_1 and ternary x_2 is $-x_2^2$.

All further features of our algorithm are straightforward: the branching rule simply consists in choosing a variable x_i with largest domain, i.e., with maximal $u_i - l_i$. This domain is divided in the middle. More sophisticated branching strategies did not lead to any improvements. Our exploration strategy is depth first, again no improvements were obtained by more refined strategies.

Note that, as mentioned before, we do not relax the integrality requirement when minimizing the separable underestimator. At each node of the branch-and-bound we hence get not only a lower bound but also an upper bound, i.e., a feasible solution for our original problem (MIPO). All we need is to re-evaluate the objective function of (MIPO) at the minimizer of the separable underestimator. No further primal heuristic has been implemented.

4 Experimental Results

For testing our algorithm, we implemented the proposed branch-and-bound scheme, which we will call PolyOpt, in C++. All experiments were run on a cluster of 64-bit Intel(R) Xeon(R) CPU E5-2670 0 processors running at 2.60GHz with 20480 KB cache. As for the instances, we randomly generated continuous, mixed, and integer instances with polynomials f of degree $d = 4$. We used the following parameters:

- The number of variables is $n \in \{10, 15, 20, 25, 30, 35, 40\}$.
- For the bounds on the variables we consider
  - $[l_i, u_i] = [-10, 10]$ for continuous, mixed-integer, and integer instances,
  - $[l_i, u_i] = [-1, 1]$ for integer (i.e., ternary) instances, and
  - $[l_i, u_i] = [0, 1]$ for integer (i.e., binary) instances,

  for all $i = 1, \ldots, n$ in each case.

– The number m of monomials appearing in f is either a multiple of n or covers all possible monomials of degree ≤ 4 (complete instances).
– The coefficients of each monomial were chosen uniformly at random in the interval $[-1, 1]$.
– The variables composing each monomial and the corresponding degree are computed by randomly generating four values within the variables index set plus 0, i.e., $\{0, 1, \ldots, n\}$. Each time value 0 is generated, the degree of the monomial decreases by one. The number of repetitions of the same index thus determines the degree of the corresponding variable.

We generated 10 different instances for each possible combination of the values above so as to show an average behavior.

PolyOpt's performance was compared against three general non-convex mixed-integer programming solvers, namely BARON v. 12.7.7 [21], Couenne v. 0.4 [1, 11], and SCIP v. 3.0.1 [19], as well as a solver for continuous polynomial optimization, namely GloptiPoly v. 3.6.1 [14]. We selected these solvers because they are capable of solving problems of type (MIPO) with any degree d, domain size, and variable type. As GloptiPoly has been developed for continuous polynomial optimization problems, we modeled the variables' integrality by adding a polynomial constraint, as GloptiPoly developers suggested. The only option we set for all the methods was a time limit of one CPU hour.

Note that all these solvers have been developed for more general classes of problems, in particular all of them allow non-trivial constraints. However, to the best of our knowledge, no other exact methods capable of solving problem (MIPO) have been proposed in the literature.

Tables 3–5 show results on pure integer instances, while results on continuous and mixed-integer instances are presented in Table 6 and 7, respectively. For each table, the first column represent n, the number of variables, and the second m, the number of monomials. The first block of rows after the header lists the results for complete instances. Each of the columns three to eight correspond to one of the compared approaches, namely SCIP, Couenne, BARON, GloptiPoly and PolyOpt in two versions, namely the one considering the global underestimator of Section 2.2 (PolyOpt-su) and the one considering the optimal underestimators of Section 2.4 (PolyOpt), respectively. At each entry we find the average CPU time needed to solve one instance to global optimality while in parentheses we state how many instances were solved out of 10. Whenever a solver is not able to solve any of the 10 instances within the time limit, we report it as ***. If only a part of the instances was solved, the given average running time refers only to the solved instances. Note that GloptiPoly returned several times the message "cannot guarantee optimality". In this case, we did not consider the instance as solved.

In Table 3 results for instances with integer variables in the box $[-10, 10]$ are shown. We first note that GloptiPoly cannot solve any instance, not even for $n = 10$. It is clearly due to the high-degree polynomial constraints needed to model the integrality requirements. The method that performs best in terms of solved instances is PolyOpt with 281 solved instances out of 310. SCIP,

n	m	scip	couenne	baron	gloptipoly	polyopt-su	polyopt
10	1001	***	***	***	***	246.83 (10)	32.42 (10)
10	10	0.15 (10)	0.06 (10)	0.21 (10)	***	1.38 (10)	0.20 (10)
10	20	0.79 (10)	4.97 (10)	0.83 (10)	***	0.99 (10)	0.10 (10)
10	30	3.61 (10)	370.71 (8)	2.84 (10)	***	1.34 (10)	0.13 (10)
10	40	31.78 (10)	109.19 (4)	18.24 (10)	***	3.48 (10)	0.45 (10)
10	50	70.25 (9)	52.58 (1)	33.02 (10)	***	4.32 (10)	0.63 (10)
10	60	291.51 (7)	178.86 (2)	51.94 (10)	***	4.21 (10)	0.61 (10)
10	70	692.92 (6)	1165.88 (2)	74.14 (10)	***	4.77 (10)	0.76 (10)
10	80	736.56 (4)	1085.39 (3)	293.39 (10)	***	9.14 (10)	1.32 (10)
10	90	989.00 (4)	***	177.17 (10)	***	8.57 (10)	1.08 (10)
10	100	677.50 (2)	1526.85 (1)	405.48 (10)	***	11.48 (10)	1.39 (10)
15	15	0.24 (10)	0.22 (10)	0.44 (10)	***	31.46 (10)	0.28 (10)
15	30	11.05 (10)	82.25 (9)	6.91 (10)	***	93.47 (10)	1.48 (10)
15	45	234.25 (9)	1887.84 (2)	113.19 (10)	***	181.52 (10)	4.99 (10)
15	60	1203.34 (7)	***	523.03 (10)	***	551.52 (10)	17.36 (10)
15	75	1371.59 (4)	***	985.78 (9)	***	581.15 (10)	16.42 (10)
15	90	***	***	1285.80 (3)	***	1059.66 (10)	38.11 (10)
15	105	***	***	2958.22 (1)	***	1525.17 (10)	51.27 (10)
15	120	***	***	898.22 (1)	***	1778.03 (10)	54.01 (10)
15	135	***	***	***	***	2044.54 (6)	90.37 (10)
15	150	***	***	***	***	2605.63 (7)	115.98 (10)
20	20	0.69 (10)	0.37 (10)	0.99 (10)	***	1921.78 (10)	3.43 (10)
20	40	307.14 (10)	707.38 (3)	102.92 (10)	***	3172.81 (2)	45.59 (10)
20	60	3593.83 (1)	***	1582.92 (6)	***	***	258.83 (10)
20	80	***	***	***	***	***	820.18 (10)
20	100	***	***	***	***	***	1370.54 (10)
20	120	***	***	***	***	***	2057.88 (9)
20	140	***	***	***	***	***	2370.40 (6)
20	160	***	***	***	***	***	2902.04 (4)
20	180	***	***	***	***	***	2066.20 (1)
20	200	***	***	***	***	***	2496.25 (1)

Table 3: Results for bounds $[-10, 10]$, integer variables.

Couenne, BARON, and PolyOpt-su can solve 133, 75, 180, and 215 of them, respectively. Having a closer look we can observe that the denser the instances are, i.e., the larger the number of monomials is, the better is PolyOpt's performance with respect to the other methods. In fact, it is the only method that, in both versions, could solve complete instances on 10 variables. Complete instances for $n = 15$ could not be solved by any of the methods.

Table 4 shows the results for pure integer instances with bounds $[-1, 1]$. We generated these instances so as to test the influence of the size of the variable domains on the performance of the different methods. These instances turn out to be slightly easier than the previous ones: the complete instances with $n = 10$ are solved by PolyOpt, PolyOpt-su, BARON, and GloptiPoly, while Couenne solves 8 out of 10 instances. For $n = 15$, only PolyOpt solves all the complete instances; GloptiPoly can solve 8 out of 10, the other methods cannot solve any instance. However, PolyOpt is again the clear winner on number of solved instances with 213 out of 220. GloptiPoly performs better here with 72 instances while PolyOpt-su, SCIP, Couenne, and BARON can solve 73, 93, 164, and 117, respectively. The results confirm the performance of PolyOpt being the best also on smaller variables domains. Moreover, besides the sparser instances, PolyOpt outperforms the other methods also if we consider the average CPU time.

Next we show the results for binary variables (Table 5). For these instances, we go up to $n = 25$ for the complete instance, to $n = 40$ for the others. In this case, SCIP and Couenne outperform the other methods both in terms of solved instances and of average CPU time, the only exception being the complete instances. For these, only GloptiPoly and PolyOpt could solve all instances up

n	m	scip	couenne	baron	gloptipoly	polyopt-su	polyopt
10	1001	***	125.68 (8)	639.95 (10)	2.53 (10)	9.24 (10)	1.04 (10)
15	3876	***	***	***	136.18 (8)	***	356.12 (10)
20	20	0.60 (10)	0.27 (10)	0.50 (10)	2290.54 (7)	115.01 (10)	0.47 (10)
20	40	12.90 (10)	4.91 (10)	11.98 (10)	2563.91 (10)	304.37 (10)	3.07 (10)
20	60	104.69 (10)	34.73 (10)	76.88 (10)	2805.90 (8)	800.97 (10)	10.06 (10)
20	80	411.59 (10)	137.07 (10)	270.80 (10)	2722.31 (3)	1530.60 (10)	26.65 (10)
20	100	1078.26 (10)	128.05 (10)	610.88 (10)	3062.31 (4)	2251.11 (9)	32.44 (10)
20	120	2042.38 (8)	259.23 (10)	1096.55 (10)	2842.35 (5)	2555.82 (7)	43.57 (10)
20	140	2557.72 (1)	330.48 (10)	1613.19 (8)	2635.04 (5)	3355.58 (4)	55.12 (10)
20	160	***	366.45 (10)	1943.67 (6)	2312.47 (6)	***	68.53 (10)
20	180	***	979.21 (10)	***	2464.21 (3)	***	81.57 (10)
20	200	***	1218.90 (10)	***	3038.32 (3)	***	98.58 (10)
25	25	1.09 (10)	0.67 (10)	1.07 (10)	***	2980.63 (3)	2.10 (10)
25	50	68.12 (10)	37.71 (10)	131.31 (10)	***	***	43.14 (10)
25	75	857.95 (10)	526.16 (10)	1182.56 (9)	***	***	210.34 (10)
25	100	2160.92 (4)	921.11 (10)	1558.19 (4)	***	***	535.91 (10)
25	125	***	1022.03 (9)	***	***	***	758.62 (10)
25	150	***	1959.41 (6)	***	***	***	1037.67 (10)
25	175	***	2647.16 (1)	***	***	***	1841.68 (10)
25	200	***	***	***	***	***	2443.04 (10)
25	225	***	***	***	***	***	2656.76 (8)
25	250	***	***	***	***	***	3082.51 (5)

Table 4: Results for bounds $[-1, 1]$, integer variables.

to $n = 20$ but the latter shows better average CPU times and could also solve some of the instances for $n = 25$. This behavior can be explained as follows: SCIP and Couenne both have specific heuristics and cuts to deal with binary variables while PolyOpt does not. The only special feature we implemented in PolyOpt was to exploit the fact that $x_i^2 = x_i$ in the binary case, so that we could replace each x_i^e with x_i for any $e \geq 2$, potentially lowering the degree of the monomials. Note that this feature was not enough to make PolyOpt-su competitive in the binary case: both in the previous tables and especially here it is clear that the performance of PolyOpt changes significantly when considering the optimal underestimators instead of the global ones.

We also tested the case of bounds in $[-10, 10]$ with continuous or mixed-integer instances, see Tables 6 and 7, respectively. As expected, the continuous instances happen to be easier to solve by GloptiPoly. BARON performs roughly similarly good while PolyOpt is slightly slower on larger instances ($n = 20$) and Couenne is slightly slower on smaller instances ($n = 10$). SCIP performs clearly worse on continuous instances. The performances on the mixed-integer instances are closer to the ones seen in the pure integer case, with the exception of Couenne that performs better. In both cases the performance of PolyOpt-su is always dominated by the one of PolyOpt. In summary, Tables 6 and 7 confirm that PolyOpt outperforms the other solvers on average, the only exception being the binary instances.

5 Conclusions and Future Directions

In this paper we presented a new method, called PolyOpt, for solving box-constrained polynomial minimization problems. It consists of a branch-and-bound scheme, where lower bounds are calculated by minimizing separable underestimators. These underestimators make the method effective as they are

n	m	scip	couenne	baron	gloptipoly	polyopt-su	polyopt
10	1001	8.75 (10)	16.98 (10)	30.08 (10)	2.38 (10)	0.25 (10)	0.05 (10)
15	3876	245.62 (10)	442.50 (10)	2377.41 (10)	105.40 (10)	35.61 (10)	1.81 (10)
20	10626	***	***	***	2520.86 (10)	2699.26 (1)	101.87 (10)
25	23751	***	***	***	***	***	1557.26 (7)
20	20	0.04 (10)	0.01 (10)	0.16 (10)	3426.62 (1)	0.92 (10)	0.02 (10)
20	40	0.23 (10)	0.12 (10)	0.72 (10)	2871.88 (9)	2.50 (10)	0.08 (10)
20	60	0.47 (10)	0.32 (10)	1.99 (10)	2592.85 (10)	5.39 (10)	0.17 (10)
20	80	0.90 (10)	0.70 (10)	5.08 (10)	2842.80 (10)	5.78 (10)	0.14 (10)
20	100	1.51 (10)	1.42 (10)	8.23 (10)	2751.33 (10)	13.70 (10)	0.36 (10)
20	120	2.09 (10)	1.95 (10)	13.13 (10)	2493.54 (10)	14.50 (10)	0.40 (10)
20	140	3.29 (10)	2.70 (10)	19.14 (10)	2490.07 (10)	21.86 (10)	0.44 (10)
20	160	4.05 (10)	3.84 (10)	29.20 (10)	2591.12 (10)	22.61 (10)	0.52 (10)
20	180	5.00 (10)	5.02 (10)	41.31 (10)	2350.67 (2)	37.77 (10)	0.75 (10)
20	200	5.90 (10)	5.59 (10)	51.77 (10)	***	31.95 (10)	0.61 (10)
30	30	0.04 (10)	0.03 (10)	0.39 (10)	***	370.58 (10)	1.29 (10)
30	60	0.47 (10)	0.60 (10)	4.99 (10)	***	1152.81 (10)	6.88 (10)
30	90	1.43 (10)	1.73 (10)	21.01 (10)	***	1521.74 (9)	8.35 (10)
30	120	3.39 (10)	4.49 (10)	65.12 (10)	***	2207.89 (3)	19.06 (10)
30	150	5.38 (10)	6.25 (10)	91.01 (10)	***	2784.61 (3)	13.77 (10)
30	180	7.16 (10)	11.11 (10)	189.31 (10)	***	3264.82 (2)	24.14 (10)
30	210	9.97 (10)	14.03 (10)	318.89 (10)	***	2258.73 (2)	22.23 (10)
30	240	14.10 (10)	19.86 (10)	671.06 (10)	***	***	32.17 (10)
30	270	19.07 (10)	28.43 (10)	971.77 (10)	***	***	29.10 (10)
30	300	28.15 (10)	42.75 (10)	1366.81 (9)	***	***	66.97 (10)
40	40	0.09 (10)	0.03 (10)	0.49 (10)	***	***	67.32 (9)
40	80	0.79 (10)	1.08 (10)	18.23 (10)	***	***	95.94 (10)
40	120	4.00 (10)	6.41 (10)	152.10 (10)	***	***	667.56 (10)
40	160	7.22 (10)	11.59 (10)	471.81 (10)	***	***	585.47 (9)
40	200	12.91 (10)	23.62 (10)	1337.27 (10)	***	***	814.77 (10)
40	240	21.88 (10)	41.51 (10)	2377.69 (8)	***	***	801.93 (8)
40	280	47.34 (10)	94.52 (10)	1862.72 (1)	***	***	1196.28 (7)
40	320	90.62 (10)	178.36 (10)	***	***	***	1219.47 (5)
40	360	120.58 (10)	217.36 (10)	***	***	***	1700.76 (9)
40	400	239.34 (10)	388.54 (10)	***	***	***	2163.80 (7)

Table 5: Results for bounds $[0, 1]$, integer variables.

n	m	scip	couenne	baron	gloptipoly	polyopt-su	polyopt
10	1001	***	***	***	3.20 (6)	548.52 (10)	86.44 (10)
10	10	0.14 (10)	0.13 (10)	0.17 (9)	2.94 (6)	291.64 (10)	17.67 (10)
10	20	3.96 (10)	18.37 (10)	0.94 (10)	2.46 (8)	31.12 (10)	19.22 (10)
10	30	491.81 (9)	8.96 (7)	3.08 (10)	2.55 (8)	2.79 (10)	0.53 (10)
10	40	552.12 (3)	172.32 (4)	31.22 (10)	2.84 (9)	11.63 (10)	0.94 (10)
10	50	1871.96 (1)	67.76 (1)	80.33 (10)	3.00 (9)	11.58 (10)	1.67 (10)
10	60	***	1958.36 (3)	51.22 (10)	2.51 (9)	10.57 (10)	1.32 (10)
10	70	***	***	93.70 (9)	2.72 (7)	12.27 (10)	1.84 (10)
10	80	***	2391.40 (2)	423.64 (9)	2.83 (8)	24.09 (10)	2.94 (10)
10	90	***	***	236.71 (10)	2.76 (8)	20.42 (10)	2.21 (10)
10	100	***	***	662.43 (10)	2.69 (6)	28.72 (10)	4.95 (10)
15	15	0.26 (10)	0.20 (10)	0.36 (10)	109.17 (8)	172.21 (9)	137.80 (10)
15	30	503.58 (10)	14.60 (9)	5.91 (10)	118.25 (9)	309.64 (10)	2.73 (10)
15	45	541.95 (1)	336.24 (1)	222.99 (9)	115.78 (9)	692.44 (10)	8.58 (10)
15	60	***	***	538.11 (9)	119.15 (7)	1859.56 (9)	37.27 (10)
15	75	***	***	1212.09 (8)	102.97 (8)	2045.09 (9)	28.27 (10)
15	90	***	***	1815.35 (3)	131.68 (8)	2346.23 (5)	72.15 (10)
15	105	***	***	3055.45 (2)	112.78 (8)	2621.26 (2)	99.17 (10)
15	120	***	***	1227.11 (1)	112.77 (8)	2823.28 (1)	110.90 (10)
15	135	***	***	***	105.31 (9)	***	172.77 (10)
15	150	***	***	***	119.62 (8)	2642.26 (1)	232.10 (10)
20	20	0.37 (10)	0.38 (10)	0.98 (10)	2403.32 (7)	2858.28 (1)	503.30 (7)
20	40	874.37 (3)	205.05 (3)	117.05 (10)	2600.67 (8)	***	224.48 (10)
20	60	***	***	1418.35 (5)	2994.37 (6)	***	454.73 (10)
20	80	***	***	***	2375.94 (8)	***	1263.02 (9)
20	100	***	***	***	2510.71 (5)	***	2036.37 (6)
20	120	***	***	***	2451.53 (9)	***	1772.69 (3)
20	140	***	***	***	2545.19 (5)	***	2838.28 (2)
20	160	***	***	***	2232.10 (7)	***	***
20	180	***	***	***	2658.68 (6)	***	***
20	200	***	***	***	2191.13 (5)	***	***
25	25	7.73 (10)	0.90 (10)	2.92 (10)	***	***	822.27 (6)
25	50	***	***	544.41 (8)	***	***	1272.61 (9)
25	75	***	***	***	***	***	1958.19 (1)
25	100	***	***	***	***	***	***
25	125	***	***	***	***	***	***
25	150	***	***	***	***	***	***
25	175	***	***	***	***	***	***
25	200	***	***	***	***	***	***
25	225	***	***	***	***	***	***
25	250	***	***	***	***	***	***

Table 6: Results for bounds [-10,10], continuous variables.

n	m	scip	couenne	baron	gloptipoly	polyopt-su	polyopt
10	1001	***	***	***	***	321.29 (10)	47.30 (10)
10	10	0.33 (10)	0.08 (10)	1.34 (10)	***	118.71 (10)	0.55 (10)
10	20	1.43 (10)	0.44 (10)	0.89 (10)	***	13.37 (10)	0.24 (10)
10	30	22.05 (10)	0.81 (10)	3.08 (10)	***	2.01 (10)	0.16 (10)
10	40	182.85 (9)	4.56 (10)	26.89 (10)	***	5.87 (10)	0.60 (10)
10	50	774.88 (9)	16.26 (10)	54.59 (10)	***	6.39 (10)	0.87 (10)
10	60	447.96 (5)	22.29 (10)	53.64 (10)	***	5.87 (10)	0.75 (10)
10	70	679.35 (4)	11.25 (10)	126.53 (10)	***	7.49 (10)	1.06 (10)
10	80	1574.58 (1)	74.86 (10)	577.44 (10)	***	13.08 (10)	1.72 (10)
10	90	3474.68 (1)	72.41 (10)	263.55 (10)	***	12.64 (10)	1.41 (10)
10	100	***	51.78 (9)	567.50 (10)	***	16.15 (10)	1.86 (10)
15	15	0.19 (10)	0.15 (10)	0.44 (10)	***	157.22 (9)	21.99 (10)
15	30	112.57 (10)	1.98 (10)	6.67 (10)	***	177.79 (10)	1.86 (10)
15	45	318.58 (7)	10.49 (9)	125.61 (10)	***	328.15 (10)	6.48 (10)
15	60	879.22 (2)	117.01 (10)	556.31 (10)	***	1043.49 (10)	23.74 (10)
15	75	***	318.52 (10)	967.85 (8)	***	1203.55 (10)	21.89 (10)
15	90	***	301.10 (6)	1440.39 (3)	***	1850.06 (9)	50.03 (10)
15	105	***	495.67 (6)	***	***	2302.83 (5)	69.93 (10)
15	120	***	586.83 (6)	952.72 (1)	***	2590.06 (5)	75.39 (10)
15	135	***	1673.22 (4)	***	***	2862.28 (1)	123.87 (10)
15	150	***	1614.30 (2)	***	***	1410.14 (1)	158.25 (10)
20	20	0.51 (10)	0.33 (10)	0.86 (10)	***	2741.94 (3)	457.89 (10)
20	40	707.09 (8)	12.50 (10)	113.98 (10)	***	***	55.10 (10)
20	60	***	383.20 (10)	1842.67 (6)	***	***	336.41 (10)
20	80	***	1141.59 (7)	***	***	***	1114.29 (10)
20	100	***	1110.76 (2)	***	***	***	1718.24 (9)
20	120	***	1984.69 (1)	***	***	***	2055.60 (6)
20	140	***	***	***	***	***	1857.18 (2)
20	160	***	1223.26 (1)	***	***	***	3321.33 (1)
20	180	***	***	***	***	***	3115.31 (1)
20	200	***	***	***	***	***	***
25	25	2.15 (10)	0.80 (10)	2.92 (10)	***	***	67.00 (9)
25	50	1233.17 (1)	51.20 (10)	606.13 (9)	***	***	1067.16 (10)
25	75	***	1237.38 (6)	3378.23 (1)	***	***	1544.15 (1)
25	100	***	1167.83 (1)	***	***	***	***
25	125	***	***	***	***	***	***
25	150	***	***	***	***	***	***
25	175	***	***	***	***	***	***
25	200	***	***	***	***	***	***
25	225	***	***	***	***	***	***
25	250	***	***	***	***	***	***

Table 7: Results for bounds [-10,10], mixed-integer variables.

quickly computed and provide a decent approximation of the original problem. Extensive computational results confirm the effectiveness of PolyOpt.

An interesting future direction could be to extend our approach to problems featuring non-trivial constraints. A straightforward idea would be to apply Lagrangian relaxation. If the problem contains only linear or, more generally, polynomial constraints, the additional Lagrangian term in the objective function would not compromise our method to solve the problem. Anyhow, the update of Lagrangian multipliers is not a trivial task to accomplish and a sophisticated stabilization technique would be needed.

Acknowledgements The work is partially supported by the EU grant FP7-PEOPLE-2012 – ITN no. 316647 "Mixed-Integer Nonlinear Optimization". The second author gratefully acknowledges the partial financial support under grant ANR 12-JS02-009-01 "ATOMIC".

References

1. Belotti P, Lee J, Liberti L, Margot F, Wächter A (2009) Branching and bounds tightening techniques for non-convex MINLP. Optimization Methods and Software 24:597–634

2. Belotti P, Kirches C, Leyffer S, Linderoth J, Luedtke J, Mahajan A (2013) Mixed-integer nonlinear optimization. Acta Numerica 22:1–131
3. Billionnet A, Elloumi S, Lambert A (2012) Extending the QCR method to general mixed integer programs. Mathematical Programming 131(1):381–401
4. Buchheim C, D'Ambrosio C (2014) Box-constrained mixed-integer polynomial optimization using separable underestimators. In: Integer Programming and Combinatorial Optimization – 17th International Conference, IPCO 2014, LNCS, vol 8494, pp 198–209
5. Buchheim C, Rinaldi G (2007) Efficient reduction of polynomial zero-one optimization to the quadratic case. SIAM Journal on Optimization 18(4):1398–1413, DOI 10.1137/050646500
6. Buchheim C, Traversi E (2013) Separable non-convex underestimators for binary quadratic programming. In: 12th International Symposium on Experimental Algorithms – SEA 2013, LNCS, vol 7933, pp 236–247
7. Buchheim C, Wiegele A (2013) Semidefinite relaxations for non-convex quadratic mixed-integer programming. Mathematical Programming 141(1–2):435–452, DOI 10.1007/s10107-012-0534-y
8. Buchheim C, De Santis M, Palagi L, Piacentini M (2013) An exact algorithm for nonconvex quadratic integer minimization using ellipsoidal relaxations. SIAM Journal on Optimization 23(3):1867–1889
9. Burer S (2010) Optimizing a polyhedral-semidefinite relaxation of completely positive programs. Mathematical Programming Computation 2(1):1–19
10. Burer S, Letchford AN (2012) Non-convex mixed-integer nonlinear programming: a survey. Surveys in Operations Research and Management Science 17:97–106
11. `COUENNE` (v. 0.4) projects.coin-or.org/Couenne
12. D'Ambrosio C, Lodi A (2011) Mixed integer nonlinear programming tools: a practical overview. 4OR 9(4):329–349
13. D'Ambrosio C, Lodi A (2013) Mixed integer nonlinear programming tools: an updated practical overview. Annals OR 204(1):301–320
14. Henrion D, Lasserre JB, Loefberg J (2009) Gloptipoly 3: moments, optimization and semidefinite programming. Optimization Methods and Software 24:761–779
15. Lasserre JB (2006) Convergent SDP-relaxations in polynomial optimization with sparsity. SIAM Journal on Optimization 17:822–843
16. Lasserre JB, Thanh TP (2013) Convex underestimators of polynomials. Journal of Global Optimization 56:1–25
17. Parrilo PA, Sturmfels B (2001) Minimizing polynomial functions. Algorithmic and quantitative real algebraic geometry, DIMACS Series in Discrete Mathematics and Theoretical Computer Science 60:83–99
18. Rosenberg IG (1975) Reduction of bivalent maximization to the quadratic case. Cahiers du Centre d'Etudes de Recherche Opérationelle 17:71–74
19. `SCIP` (v. 3.0.1) http://scip.zib.de/scip.shtml

20. Shor NZ (1987) Class of global minimum bounds of polynomial functions. Cybernetics 23:731–734
21. Tawarmalani M, Sahinidis NV (2005) A polyhedral branch-and-cut approach to global optimization. Mathematical Programming 103:225–249
22. Vandenbussche D, Nemhauser GL (2005) A branch-and-cut algorithm for nonconvex quadratic programs with box constraints. Mathematical Programming 102(3):559–575