On the composition of convex envelopes for quadrilinear terms*

Pietro Belotti, Sonia Cafieri, Jon Lee, Leo Liberti, and Andrew Miller

Abstract Within the framework of the spatial Branch-and-Bound algorithm for solving Mixed-Integer Nonlinear Programs, different convex relaxations can be obtained for multilinear terms by applying associativity in different ways. The two groupings $((x_1x_2)x_3)x_4$ and $(x_1x_2x_3)x_4$ of a quadrilinear term, for example, give rise to two different convex relaxations. In [6] we prove that having fewer groupings of longer terms yields tighter convex relaxations. In this paper we give an alternative proof of the same fact and perform a computational study to assess the impact of the tightened convex relaxation in a spatial Branch-and-Bound setting.

P. Belotti
Dept. of Mathematical Sciences, Clemson University, Clemson SC 29634, USA
e-mail: pbelott@clemson.edu

S. Cafieri
Dept. de Mathématiques et Informatique, École Nationale de l'Aviation Civile, 31055 Toulouse, France
e-mail: sonia.cafieri@enac.fr

J. Lee
Dept. of Mathematical Sciences, IBM T.J. Watson Research Center, PO Box 218, Yorktown Heights, N.Y. 10598, USA
e-mail: jonlee@us.ibm.com

L. Liberti
LIX, École Polytechnique, 91128 Palaiseau, France
e-mail: liberti@lix.polytechnique.fr

A. Miller
Institut de Mathématiques de Bordeaux, 33405 Talence, RealOpt, INRIA Bordeaux Sud-Ouest, France
e-mail: andrew.miller@math.u-bordeaux1.fr

* The second and the fourth authors gratefully acknowledge financial support under ANR grant 07-JCJC-0151.

1 Introduction

One of the most crucial steps of the spatial Branch-and-Bound algorithm for solving Mixed-Integer Nonlinear Programming (MINLP) problems is the lower bound computation. When the MINLP is factorable, it is possible to construct a convex relaxation automatically by means of a particular type of lifting reformulation (called MINLP standard form [27, 10]) first proposed in [16] and then exploited in most existing sBB algorithms [22, 1, 27, 9, 31, 5]. If we consider polynomial problems, higher order monomials are recursively rewritten as products of monomials of sufficiently low order for which a tight convex relaxation (possibly the convex envelope) is known. Each lower order monomial is replaced by an added variable, and an equality constraint defining the added variable in terms of the monomial it replaces is adjoined to the MINLP. This operation is carried out recursively until the MINLP consists of a linear objective, some linear constraints, and several *defining constraints* of the form $w_j = h_j(x,w)$ for all j in some appropriate set J, where the functions h_j represent monomials. To obtain a convex relaxation, each defining constraint is replaced by a set of constraints defining the convex relaxation of its feasible set, thus yielding a convex relaxation for the whole problem.

Let $B = [x^L, x^U]$. The quadrilinear feasible set $S^4 = \{(w_1,x_1,x_2,x_3,x_4) \mid w_1 = x_1x_2x_3x_4\} \cap B$ over a box can be lifted in many different ways according to the way associativity is applied: the grouping $((x_1x_2)x_3)x_4$, for example, yields the set $S^{2,2,2} = \{(w_1,w_2,w_3,x_1,x_2,x_3,x_4) \mid w_2 = x_1x_2 \wedge w_3 = w_2x_3 \wedge w_1 = w_3x_4\} \cap B$, whereas the grouping $(x_1,x_2,x_3)x_4$ yields $S^{3,2} = \{(w_1,w_2,x_1,x_2,x_3,x_4) \mid w_2 = x_1x_2x_3 \wedge w_1 = w_2x_4\} \cap B$. Since convex/concave envelopes exist in explicit form for both bilinear [2, 16] and trilinear terms [18, 17], we can derive two different convex relaxations of S^4. The first, $\overline{S}^{2,2,2}$, consists in replacing the bilinear constraints $w_i = x_jx_k$ appearing in $S^{2,2,2}$ by the corresponding bilinear envelopes. The second, $\overline{S}^{3,2}$, consists in replacing the trilinear terms with the trilinear envelope and the bilinear term with the bilinear envelope. A question then arises naturally: which one is tighter?

In [6] we proved that $\overline{S}^{3,2} \subseteq \overline{S}^{2,2,2}$ and performed a computational study of the containment of the convex relaxations when different parameters were varied. In this paper we provide an alternative proof (based on formal grammars) of the same result, and then test the impact of the tightened convex relaxation $\overline{S}^{3,2}$ using sBB.

The rest of this paper is organized as follows. In Sect. 2 we present the main motivations of this work and a literature review on convex relaxations for multilinear monomials and their impact on a sBB algorithm. In Sect. 3 we propose a theoretical framework, based on concepts from the formal languages theory, to compare convex relaxations of multilinear monomials obtained as a composition of convex envelopes of lower-degree monomials. In Sect. 4 we discuss some computational experiments aimed at comparing different convex relaxations of quadrilinear terms in a spatial Branch-and-Bound setting. Concluding remarks are given in Sect. 5.

2 Motivation and literature

The above discussion implies that deriving convex relaxations that are as strong as possible (i.e., that approximate the convex hull as closely as possible) for multilinear monomials can be critically important for the performance of a spatial branch-and-bound algorithm designed to globally solve nonconvex polynomial optimization problems. Because of this, numerous efforts have studied the convex hulls of sets defined by lower order product terms and the use of these convex hulls in recursively factorized formulations (such as the MINLP standard form defined above).

Four valid inequalities for the three-dimensional set $S^2 = \{w, x_1, x_2 : w = x_1 x_2, x \in [x^L, x^U]\}$ were proposed by [16], and later [2] showed that these four inequalities suffice to describe the convex hull. At present most global MINLP solvers that use general sBB methods (among recent examples see [5], [14], [23]) use the convex hull for recursively defined instances of S^2 to define the polyhedral relaxations that are solved at each node of the branch-and-bound tree.

However, it may be thought that limiting solvers to the use of envelopes defined by simple bilinear terms may result in convex approximations for the original problem that are less strong (perhaps much less so) than those that exploit envelopes for more complex expressions. For problems involving multilinear multinomials defined by products of more than two variables, this consideration has motivated research into the envelopes of *trilinear* functions [18, 17]. Comparing the use of convex envelopes for bilinear and trilinear forms in building convex approximations for MINLPs motivated the study in [6], and comparisons involving more general functional forms motivate the present article.

A natural generalization of bi- and tri-linear functions are functions that are known to have *vertex polyhedral* convex envelopes. (The convex envelope of an n-dimensional function $f(x)$ is said to be vertex polyhedral if its domain X is a polyhedron, and if every extreme point of the convex hull of $\{(x, f(x) : x \in X\}$ is defined by an extreme point of X itself.) In [19] Meyer and Floudas generalized the approach developed for trilinear functions to functions with vertex polyhedral convex envelopes. Essentially, their approaches can be thought of as enumerative methods that consider all possible combinations of $n+1$ extreme points of X (equivalently, extreme points of $\text{conv}(\{(x, f(x)) : x \in X\})$), and then establish conditions under which the hyperplane defined by such a set of points defines a linear inequality satisfied by all the other extreme points of $\text{conv}(\{(x, f(x)) : x \in X\})$. Such an inequality is then valid for $\{(x, f(x)) : x \in X\}$ and facet-defining for the convex hull of this set.

General multilinear functions (i.e., any function composed of a sum of products of variables, in which the degree of each variable in each product is 0 or 1) were shown to have vertex polyhedral convex envelopes by [21]. An implication of this result is that many of the concepts mentioned in the preceding paragraph can be used for general mutilinear functions; their use is not limited to monomial products (for example). The extension of such results to define convex envelopes for multilinear functions (and generalizations of them) has been discussed in [26, 28, 29, 30], among other references.

Empirical testing of the approaches mentioned above (beyond the use of bilinear envelopes defined by [16]) has been limited, but recently authors have begun exploiting some of these concepts to solve quadratically constrained quadratic programs, in which sums of bilinear products often figure prominently. In particular, the authors of [4] discuss how to dynamically generate facets of the convex hull of the sum of bilinear products in order to define a stronger relaxation of the original MINLP, and they report that strengthening the formulation with such inequalities can significantly improve the performance of BARON [23], which by default uses only McCormick envelopes to exploit multilinear terms in defining convex relaxations. Even more recently, [15] provides rigorous bounds for how much the approach of [23] (and, implicitly, of [26]) can strengthen the relaxations defined by the use of McCormick envelopes, and also provides numerical results illustrating that these bounds are tight.

It is important to note that the bounds defined by [15] apply only to problems that have sums of bilinear products, but not quadratic terms (i.e., if the quadratic function in a given constraint is represented by $f_Q(x) = x^T Qx$, the bounds defined in [15] are valid for problems in which the diagonal elements of Q are all 0). Moreover, computational experience seems to confirm that the smaller the absolute values of elements on the diagonal of Q are in comparison to the off-diagonal elements, the more important the role played by strong convex relaxations for bilinear functions becomes in defining strong relaxations for the MINLP. (Defining effective relaxations for nonconvex quadratically constrained problems in which the diagonal elements of Q are large requires, in addition to the techniques described in this section, other methods that are fundamentally different. References that discuss solving nonconvex quadratically constrained problems with large diagonal absolute values include [3, 4, 8, 24, 25], and the references contained therein.)

An unresolved issue that is directly related to much of the research on multilinear functions described above is the question of whether or not it is possible to define a description of the convex envelope of multilinear functions that does not require the explicit a priori enumeration of all of the extreme points of the domain. More formally, given an n-dimensional function $f(x) = \prod_{i=1}^{n} x_i$ over a domain $B = [x^L, x^U]$, is it possible to define a set of criteria that 1) each facet of the convex envelope must satisfy, and 2) can be checked in time polynomial in n? Most of the approaches described above, as well as the motivation of this article, are based on the implicit assumption that the answer to this question is no. However, only a comparatively small number of research efforts (e.g., [26, 15]) have addressed this question directly. Moreover, their consideration of this question has been limited to establishing criteria for x^L and x^U that are sufficient to guarantee that the answer is yes.

Computational complexity theory, and in particular results of [7] suggests that a short (i.e., polynomial in n) description of the convex envelopes of multilinear functions can be defined if and only if the following optimization problem is polynomial solvable:

$$\min \ \prod_{i=1}^{n} x_i - \sum_{i=1}^{n} c_i x_i \tag{1}$$

$$\text{s.t.} \ \ x_i^L \leq x_i \leq x_i^U, i = 1, ..., n, \tag{2}$$

$$\tag{3}$$

where $c \in \mathbb{R}^n$ is some rational vector. It seems that this problem is likely to be *NP*-complete unless fairly restrictive assumptions on x^L and x^U are satisfied. For example, generalizing some of the results of [15], in [20] the authors show that it is possible to solve the above optimization problem in polynomial time if there exists a constant $a < 1$ such that $ax_i^L = x_i^U$ for $i = 1, ..., n$. It is also clear that slightly more general conditions can be established. However, the authors of [20] conjecture that the above optimization problem in *NP*-complete in general, and the complexity of this problem remains an important open question in the area of how best to approximate the convex envelopes of functions involving multilinear terms.

We will next turn to the general question of when, and how, one approach to defining convex relaxations of factorable functions can be shown to yield relaxations that are stronger than those generated by another approach. The primary contribution of this article is to establish a general result concerning this issue. We should perhaps first note, however, that this contribution does not tell us how much stronger the dominant formulation will be; this is necessarily an empirical question. Moreover, the comparative ease with which different relaxations can be solved is also a necessarily empirical criterion, and in general both of these considerations must be weighed in considering relaxation to use in a given situation.

3 The composition of convex envelopes

In this section we prove that a stronger relaxation is obtained when one replaces "large terms" with tight convex relaxations instead of breaking up such terms in sums/products of smaller terms before replacing each small term with its respective convex relaxation. Although we find that this is quite an intuitive result, because of the inherently recursive nature of factorable functions and of the fact that we deal with a recursive symbolic procedure for constructive the convex relaxation, we did not find it easy to prove this result formally. For this purpose, we use theoretical tools that are well known to the formal languages community but perhaps not so commonly found in the optimization literature: this is why we detail every step and attempt to be somewhat didactical in presentation, alternating formal statements to informal explanations and examples. To the well-versed in such matters, a brief glimpse to the section might suffice to understand our strategy: assign a special semantic value (the corresponding convex relaxation) to each operator node of an expression tree, define the semantics of the composition operator, and finally compare the resulting relaxation with the tight convex relaxation given for the composite operator at "atomic" level.

3.1 Alphabets, languages and grammars

An *alphabet* $\mathscr{A}$ is a set of symbols. We let $\mathscr{A}^*$ be the set of all finite sequences of elements of $\mathscr{A}$. A *formal language* $\mathscr{L}$ is a subset of $\mathscr{A}^*$. A language $\mathscr{L}$ is *decidable* if, given a string $s \in \mathscr{A}^*$, there exists a finite algorithmic procedure that decides whether $s \in \mathscr{L}$ or not.

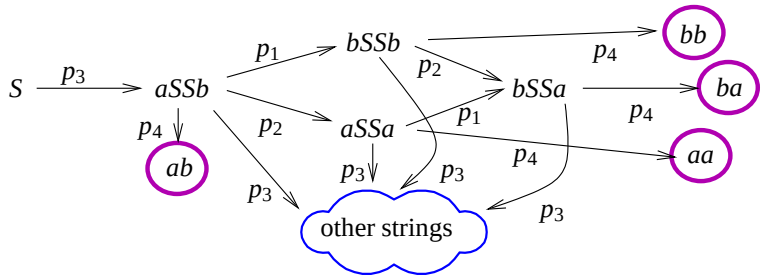
Informally, decidability of a language is concerned with its syntax: is a string a valid element of the language or not? Having decided what a language is, we have to decide what it says: to every string there corresponds a semantic value, which, in the theory and language of Zermelo-Fraenkel, is usually a set. In this setting, our formal language is the set of all valid functions $f(x)$ that can be written as finite strings of simbols in infix notation. The semantic values assigned to $f(x)$ are sets such as $\{(w,x) \in \mathbb{R}^{n+1} \mid w = f(x) \wedge x^L \leq x \leq x^U\}$ (exact semantics) and $\{(w,x) \in \mathbb{R}^{n+1} \mid w \in R(f,x^L,x^U) \wedge x^L \leq x \leq x^U\}$ (relaxed semantics) where $R(f,x^L,x^U)$ is a convex relaxation of the exact semantics. Since the cardinality of our language is countably infinite, we cannot explicitly assign exact/relaxed semantics to each function in the language. Instead, we recall that a decidable language has finite procedure for recognizing strings in the language: for each of the (finitely many) operations specified by this procedure we define a corresponding operation on the semantic values involved, thus obtaining a semantic definition for the whole language.

To this effect, we make use of possibly the best known device for specifying the syntax of a formal language $\mathscr{L}$, i.e. a *formal grammar*. This is a quadruplet $\Gamma = (\Sigma, N, P, S)$ such that:

- $\Sigma \subseteq \mathscr{A}$ is the set of *terminal symbols*
- N is a set of *nonterminal symbols* ($N \cap \Sigma = \emptyset$)
- P is a set of *rewriting*, or *production rules* ($P \subseteq (\Sigma \cup N)^* N (\Sigma \cup N)^* \rightarrow (\Sigma \cup N)^*$)
- $S \in N$ is the *start symbol*.

In practice, one recursively applies the production rules to the start symbol as many times as possible, generating strings in $(\Sigma \cup N)^*$. Those generated strings that are in Σ^* are strings of the language $\mathscr{L}$. If a string in $\mathscr{A}^*$ is not in the set of all strings in Σ^* that the grammar generates, then it is not in $\mathscr{L}$.

Example 1. Consider the alphabet $\{a,b\}$ and the grammar given by $N = \{S\}$ where S is the start symbol, $\Sigma = \{a,b\}$ and the production rules $\langle p_1 = aS \rightarrow bS, p_2 = Sb \rightarrow Sa, p_3 = S \rightarrow aSSb, p_4 = SS \rightarrow \emptyset \rangle$. We repeatedly apply $p_1, \ldots, p_4$ to the start symbol, obtaining the situation below:



From this, we conclude that aa, ab, ba, bb are in the language specified by the grammar. It must be remarked that formal grammars can also be given for languages which are not decidable (for example if the recursion does not terminate); this is one such grammar: the repeated application of p_3 yields longer and longer strings all involving the nonterminal symbol S. □

3.2 Mathematical expression language: syntax

We now formally define our function language through the use of a formal grammar. We use an alphabet $\mathscr{A} = \mathbb{X} \cup \mathbb{K} \cup \mathbb{B} \cup \mathbb{O}$ where $\mathbb{X} = \{x_1, \ldots, x_n\}$ is the set of symbols denoting original variables, $\mathbb{K}$ is the set of all computable numbers, $\mathbb{B} = \{\text{`('}, \text{`)'}\}$ and $\mathbb{O}$ is a finite set of operators $\{+, -, \times, \div, \uparrow, \sqrt{}, \log, \exp, \sin, \cos, \tan\}$, where $+, \times$ are binary operators, $-$ can be unary or binary and $\uparrow$ is the (binary) power operator. The grammar Γ is defined as follows. The start symbol is $\mathscr{F}$, $N = \{\mathscr{F}\}$, $\Sigma = \mathscr{A}$, and P is:

$$\mathscr{F} \longrightarrow x_i \in \mathbb{X} \quad (4) \qquad \mathscr{F} \longrightarrow \cos(\mathscr{F}) \quad (11)$$
$$\mathscr{F} \longrightarrow k \in \mathbb{K} \quad (5) \qquad \mathscr{F} \longrightarrow \tan(\mathscr{F}) \quad (12)$$
$$\mathscr{F} \longrightarrow (\mathscr{F}) \quad (6) \qquad \mathscr{F} \longrightarrow (\mathscr{F} - \mathscr{F}) \quad (13)$$
$$\mathscr{F} \longrightarrow (-\mathscr{F}) \quad (7) \qquad \mathscr{F} \longrightarrow (\mathscr{F} \div \mathscr{F}) \quad (14)$$
$$\mathscr{F} \longrightarrow \log(\mathscr{F}) \quad (8) \qquad \mathscr{F} \longrightarrow (\mathscr{F} \uparrow \mathscr{F}) \quad (15)$$
$$\mathscr{F} \longrightarrow \exp(\mathscr{F}) \quad (9) \qquad \mathscr{F} \longrightarrow (\mathscr{F} + \mathscr{F}) \quad (16)$$
$$\mathscr{F} \longrightarrow \sin(\mathscr{F}) \quad (10) \qquad \mathscr{F} \longrightarrow (\mathscr{F} \times \mathscr{F}) \quad (17)$$

Notice that rules (4)-(5) are given in schematic form: i.e. the string on the left of the arrow is not in $(\Sigma \cup N)^*$, but it is possible to define "sub-languages" that decide whether a string is in $\mathbb{X}$ or in $\mathbb{K}$.

Example 2. In order to recognize that the string $F \equiv x_1 + ((x_2 \uparrow 2) + (x_3 \times (x_4 \times \log(x_1))))$ is in $\mathscr{L}$ we can apply the production rules as follows (there are other possible orders in which the rules can be applied yielding the same result):

$$
\begin{array}{rll}
\mathscr{F} & -[1] \rightarrow & (\mathscr{F}+\mathscr{F}) \quad \text{by (16)} \\
& -[2] \rightarrow & (\mathscr{F}+(\mathscr{F}+\mathscr{F})) \quad \text{by (16)} \\
& -[3] \rightarrow & (\mathscr{F}+((\mathscr{F}\uparrow\mathscr{F})+\mathscr{F})) \quad \text{by (15)} \\
& -[4] \rightarrow & (\mathscr{F}+((\mathscr{F}\uparrow\mathscr{F})+(\mathscr{F}\times\mathscr{F}))) \quad \text{by (17)} \\
& -[5] \rightarrow & (\mathscr{F}+((\mathscr{F}\uparrow\mathscr{F})+(\mathscr{F}\times(\mathscr{F}\times\mathscr{F})))) \quad \text{by (17)} \\
& -[6] \rightarrow & (\mathscr{F}+((\mathscr{F}\uparrow\mathscr{F})+(\mathscr{F}\times(\mathscr{F}\times\log(\mathscr{F}))))) \quad \text{by (8)} \\
& -[7] \rightarrow & (x_1+((x_2\uparrow\mathscr{F})+(\mathscr{F}\times(\mathscr{F}\times\log(\mathscr{F}))))) \quad \text{by (4)} \\
& -[8] \rightarrow & (x_1+((x_2\uparrow\mathscr{F})+(\mathscr{F}\times(\mathscr{F}\times\log(\mathscr{F}))))) \quad \text{by (4)} \\
& -[9] \rightarrow & (x_1+((x_2\uparrow\mathscr{F})+(x_3\times(\mathscr{F}\times\log(\mathscr{F}))))) \quad \text{by (4)} \\
& -[10] \rightarrow & (x_1+((x_2\uparrow\mathscr{F})+(x_3\times(x_4\times\log(\mathscr{F}))))) \quad \text{by (4)} \\
& -[11] \rightarrow & (x_1+((x_2\uparrow\mathscr{F})+(x_3\times(x_4\times\log(x_1))))) \quad \text{by (4)} \\
& -[12] \rightarrow & (x_1+((x_2\uparrow 2)+(x_3\times(x_4\times\log(x_1))))) \quad \text{by (5)} \\
& -[13] \rightarrow & x_1+((x_2\uparrow 2)+(x_3\times(x_4\times\log(x_1)))) \quad \text{by (6).}
\end{array}
$$

We need to apply 13 rewriting rules in order to recognize that $F \in \mathscr{L}$. □

3.3 Mathematical expression language: semantics

We are now going to use the formal grammar Γ to assign semantic values to strings. Informally, we assign different sets to the different occurrences of the symbol $\mathscr{F}$ in each production rule, in such a way that the set assigned to $\mathscr{F}$ appearing in the left hand side of each rule is defined in terms of the sets assigned to the symbols $\mathscr{F}$ appearing in the right hand side. More precisely, for a production rule ρ in (4)-(17) of the form $\mathscr{F} \rightarrow T$, where $T \in (\Sigma \cup N)^*$, let $\nu(\rho)$ be the number of occurrences of the symbol $\mathscr{F}$ in the string T. Let $X_0(\rho)$ be the set assigned to the symbol $\mathscr{F}$ appearing on the left hand side of ρ, and for all $i \in \{1,\ldots,\nu(\rho)\}$ let $X_i(\rho)$ be the set assigned to the i-th occurrence of the symbol $\mathscr{F}$ in T.

3.3.1 Exact semantics

The *exact semantics* of $\mathscr{L}$ is defined according to the following rules.

$$
\begin{aligned}
\mathscr{F} &\longrightarrow x_i \in \mathbb{X} && : X_0 = [x_i^L, x_i^U] \\
\mathscr{F} &\longrightarrow k \in \mathbb{K} && : X_0 = \{k\} \\
\mathscr{F} &\longrightarrow (\mathscr{F}) && : X_0 = X_1 \\
\mathscr{F} &\longrightarrow (-\mathscr{F}) && : X_0 = \{(w,x) \mid w = -x \wedge x \in X_1\} \\
\mathscr{F} &\longrightarrow \log(\mathscr{F}) && : X_0 = \{(w,x) \mid w = \log(x) \wedge x \in X_1\} \\
\mathscr{F} &\longrightarrow \exp(\mathscr{F}) && : X_0 = \{(w,x) \mid w = \exp(x) \wedge x \in X_1\} \\
\mathscr{F} &\longrightarrow \sin(\mathscr{F}) && : X_0 = \{(w,x) \mid w = \sin(x) \wedge x \in X_1\} \\
\mathscr{F} &\longrightarrow \cos(\mathscr{F}) && : X_0 = \{(w,x) \mid w = \cos(x) \wedge x \in X_1\} \\
\mathscr{F} &\longrightarrow \tan(\mathscr{F}) && : X_0 = \{(w,x) \mid w = \tan(x) \wedge x \in X_1\} \\
\mathscr{F} &\longrightarrow (\mathscr{F} - \mathscr{F}) && : X_0 = \{(w,x_1,x_2) \mid w = x_1 - x_2 \wedge \forall i \in \{1,2\}\ x_i \in X_i\} \\
\mathscr{F} &\longrightarrow (\mathscr{F} \div \mathscr{F}) && : X_0 = \{(w,x_1,x_2) \mid w = x_1/x_2 \wedge \forall i \in \{1,2\}\ x_i \in X_i\} \\
\mathscr{F} &\longrightarrow (\mathscr{F} \uparrow \mathscr{F}) && : X_0 = \{(w,x_1,x_2) \mid w = x_1^{x_2} \wedge \forall i \in \{1,2\}\ x_i \in X_i\} \\
\mathscr{F} &\longrightarrow (\mathscr{F} + \mathscr{F}) && : X_0 = \{(w,x_1,x_2) \mid w = x_1 + x_2 \wedge \forall i \in \{1,2\}\ x_i \in X_i\} \\
\mathscr{F} &\longrightarrow (\mathscr{F} \times \mathscr{F}) && : X_0 = \{(w,x_1,x_2) \mid w = x_1 x_2 \wedge \forall i \in \{1,2\}\ x_i \in X_i\}
\end{aligned}
$$

A meta-linguistic note: the naming of the semantic values X_0, X_1, X_2 must be local to each rule. Otherwise, if the same rule ρ is applied twice, we might get two different definitions assigned to the same name $X_0(\rho)$. In order to obtain a consistent naming, we observe that the recursive nature of string recognition in $\mathscr{L}$ is finite, so the different strings of $(\Sigma \cup N)^*$ generated during the recognition procedure can be listed in the order of rewriting, as in Example 2. For a string $f \in \mathscr{L}$ let $r(f)$ be the length of this list. For all $k \leq r(f)$, we can now let X_0^k be the semantic value assigned to $\mathscr{F}$ appearing in the left hand side of the production rule ρ being applied at the k-th rewriting step, and let $X_1^k, \ldots, X_{v(rho)}^k$ be the sets assigned to the various occurrences of $\mathscr{F}$ in the right hand side of ρ.

As will appear clear in Example 3, some of the semantic sets will be projections of other semantic sets on some of their coordinates. For every semantic set X we shall therefore let $\mathscr{V}(X)$ be the sequence of variable symbols in terms of which X is defined (so that $X \subseteq \mathbb{R}^{|\mathscr{V}(X)|}$), and for all $W \subseteq \mathscr{V}(X)$ let $\pi(X,W)$ be the projection of X on the w coordinate (if $W = \{w\}$, we write $\pi(X,w)$).

Example 3. The exact semantics of F, as defined in Example 2, is derived as follows.

$$
\begin{aligned}
X_0^1 &= \{(w_1,w_2,w_3) \mid w_1 = w_2 + w_3 \wedge w_2 \in X_1^1 \wedge w_3 \in X_2^1\} \\
X_0^2 &= \{(w_3,w_4,w_5) \mid w_3 = w_4 + w_5 \wedge w_4 \in X_1^2 \wedge w_5 \in X_2^2\} \text{ and } X_2^1 = \pi(X_0^2, w_3) \\
X_0^3 &= \{(w_4,w_6,w_7) \mid w_4 = w_6^{w_7} \wedge w_6 \in X_1^3 \wedge w_7 \in X_2^3\} \text{ and } X_1^2 = \pi(X_0^3, w_4) \\
X_0^4 &= \{(w_5,w_8,w_9) \mid w_5 = w_8 w_9 \wedge w_8 \in X_1^4 \wedge w_9 \in X_2^4\} \text{ and } X_2^2 = \pi(X_0^4, w_5) \\
X_0^5 &= \{(w_9,w_{10},w_{11}) \mid w_9 = w_{10} w_{11} \wedge w_{10} \in X_1^5 \wedge w_{11} \in X_2^5\} \text{ and } X_2^4 = \pi(X_0^5, w_9) \\
X_0^6 &= \{(w_{11},w_{12}) \mid w_{11} = \log(w_{12}) \wedge w_{12} \in X_1^6\} \text{ and } X_2^5 = \pi(X_0^6, w_{11}) \\
X_0^7 &= [x_1^L, x_1^U] \text{ and } X_1^1 = X_0^7 \\
X_0^8 &= [x_2^L, x_2^U] \text{ and } X_1^3 = X_0^8 \\
X_0^9 &= [x_3^L, x_3^U] \text{ and } X_1^4 = X_0^9 \\
X_0^{10} &= [x_4^L, x_4^U] \text{ and } X_1^5 = X_0^{10}
\end{aligned}
$$

$X_0^{11} = [x_1^L, x_1^U]$ and $X_1^6 = X_0^{11}$
$X_0^{12} = \{2\}$ and $X_2^3 = X_0^{12}$
$X_0^{13} = X_0^1$.

Replacing symbols where possible, we obtain a definition of the exact semantics of our string in function of only six sets and ten variables (4 original variables and 6 added variables):

$$
\begin{aligned}
X_0^1 &= \{(w_1, x_1, w_3) \mid w_1 = x_1 + w_3 \wedge x_1 \in [x_1^L, x_1^U] \wedge w_3 \in \pi(X_0^2, w_3)\} \\
X_0^2 &= \{(w_3, w_4, w_5) \mid w_3 = w_4 + w_5 \wedge w_4 \in \pi(X_0^3, w_4) \wedge w_5 \in \pi(X_0^4, w_5)\} \\
X_0^3 &= \{(w_4, x_2) \mid w_4 = x_2^2 \wedge x_2 \in [x_2^L, x_2^U]\} \\
X_0^4 &= \{(w_5, x_3, w_9) \mid w_5 = w_8 w_9 \wedge x_3 \in [x_3^L, x_3^U] \wedge w_9 \in \pi(X_0^5, w_9)\} \\
X_0^5 &= \{(w_9, x_4, w_{11}) \mid w_9 = w_{10} w_{11} \wedge x_4 \in [x_4^L, x_4^U] \wedge w_{11} \in \pi(X_0^6, w_{11})\} \\
X_0^6 &= \{(w_{11}, x_1) \mid w_{11} = \log(x_1) \wedge x_1 \in [x_1^L, x_1^U]\}.
\end{aligned}
$$

Suppose now we consider an enriched alphabet $\mathscr{A}'$ with one more 4-ary operator $\otimes$ such that $\otimes(x_1, \ldots, x_4) = x_1 + x_2^2 + x_3 x_4 \log(x_1)$, and an extended grammar with one more production rule $\rho' \equiv \mathscr{F} \longrightarrow \mathscr{F} + \mathscr{F} \uparrow 2 + \mathscr{F} \times \mathscr{F} \log(\mathscr{F})$. The generated language $\mathscr{L}$' is identical to $\mathscr{L}$ because we showed previously that $\mathscr{L}$ contains strings as that appearing in the right hand side of ρ' even without the production rule ρ'. However, using the extended grammar, the string F can be recognized in only one step. By replacement of the appropriate variable symbols w_ℓ, the exact semantics $\{(w,x) \mid w = \otimes(x_1, \ldots, x_4) \wedge \forall i \leq 4 x_i \in [x_i^L, x_i^U]\}$ of F computed with the extended grammar is precisely the projection of X_0^1 on the subspace of $\mathbb{R}^{10}$ spanned by $(w_1, x_1, \ldots, x_4)$. $\square$

3.3.2 Relaxed semantics

We now define the relaxed semantics of $\mathscr{L}$. Whereas in the exact semantics we assigned to each string the set of values taken by the corresponding function as its arguments range in the appropriate (recursively defined) sets, the relaxed semantics assigns to strings convex relaxations of such sets. To this end, we shall describe an operator $\mathscr{R}_\Gamma$ that computes the convex relaxation of a set using the composition of production rules in Γ. For each operator $\oplus \in \mathbb{O}$, let $\alpha(\oplus)$ be its arity (the number of its arguments). Denote $\alpha(\oplus)$ by ℓ, $\mathbb{I}$ the class of all closed and bounded intervals in $\mathbb{R}$, and let $I_1, \ldots, I_\ell \in \mathbb{I}$; then we use the notation $\mathscr{R}_\Gamma(\oplus, I_1, \ldots, I_\ell)$ to indicate a convex relaxation in $\mathbb{R}^{n+1}$ of the exact semantic value of $\oplus$, i.e. the set $\{(w_0, w_1, \ldots, w_\ell) \mid w_0 = \oplus(w_1, \ldots, w_\ell) \wedge \forall i \leq n \, (w_i \in I_i)\}$. We impose a consistency (monotonicity) requirement:

$$
\begin{aligned}
\forall \oplus \in \mathbb{O}, I_1, \ldots, I_\ell, J \in \mathbb{I} \;\; \text{s.t.} \;\; & \exists i \leq \ell \; J \subseteq I_i \\
& \mathscr{R}_\Gamma(\oplus, I_1, \ldots, I_\ell) \;\; \supseteq \;\; \mathscr{R}_\Gamma(\oplus, I_1, \ldots, I_{i-1}, J, I_{i+1}, \ldots, I_\ell)), \qquad (18)
\end{aligned}
$$

which means that convex relaxations should get tighter when the definition intervals get smaller.

We remark that $\mathscr{R}$ is a symbol in the metalanguage, in the sense that it should be replaced by an actual description of the convex sets assigned to each operator (in other words, it stands for the sentence "for all possibly ways of defining convex relaxations of operators..."). A typical definition of $\mathscr{R}_\Gamma$ used by most sBB solver codes (e.g. ooOPS [13] and Couenne [5], both based on a grammar very similar to Γ) is as follows: for all linear operators, $\mathscr{R}_\Gamma$ applied to that operator is the same as the exact semantics (because, as an affine space defined over a cartesian product of intervals, it is convex). The $\log, \exp$ operators are concave/convex univariate, and hence $\mathscr{R}_\Gamma$ is defined as a convex subset of $\mathbb{R}^2$ delimited by the function itself and the secant at the interval endpoints [9]; for piecewise convex/concave functions we employ the convex envelope defined in [12]; for trigonometric functions it is easy to work out convex relaxations/envelopes using secants and convex/concave portions of the functions themselves. We remark that providing convex/concave relaxations/envelopes of convex/concave functions and piecewise convex/concave functions suffices to define $\mathscr{R}_\Gamma$ over all univariate monomials of the form x^k where $x \in I \in \mathbb{I}$. For bilinear products, we employ the well-known McCormick envelopes:

$$\begin{aligned}\mathscr{R}(\times, [w_1^L, w_1^U], [w_2^L, w_2^U]) = \{(w_0, w_1, w_2) \mid \\ w_0 \geq w_1^L w_2 + w_2^L w_1 - w_1^L w_2^L \wedge \\ w_0 \geq w_1^U w_2 + w_2^U w_1 - w_1^U w_2^U \wedge \\ w_0 \leq w_1^L w_2 + w_2^U w_1 - w_1^L w_2^U \wedge \\ w_0 \leq w_1^U w_2 + w_2^L w_1 - w_1^U w_2^L) \wedge \\ w_1 \in [w_1^L, w_1^U] \wedge w_2 \in [w_2^L, w_2^U]\}.\end{aligned}$$

It is easy to check that the above definition of $\mathscr{R}$ satisfies (18).

The *relaxed semantics* of $\mathscr{L}$ is defined according to the rules:

$$\mathscr{F} \longrightarrow \oplus(\mathscr{F}, \ldots, \mathscr{F}) : X_0 = \mathscr{R}_\Gamma(\oplus, I_1, \ldots, I_{\alpha(\oplus)}).$$

Relaxed semantics can be combined following grammatic production rule composition in much the same way as exact semantics can, by noticing that when X is a convex subset of $\mathbb{R}^n$, the projection of X on one coordinate axis is always an interval (because projection preserves convexity).

Now let F be a valid string of $\mathscr{L}$: then F is a mathematical expression with, say, $x = (x_1, \ldots, x_n)$ as variable symbol arguments corresponding to a certain mathematical function $f : \mathbb{R}^n \to \mathbb{R}$. Then we can certainly add the following rule to Γ:

$$\rho' \equiv \mathscr{F} \longrightarrow F(\underbrace{\mathscr{F}, \ldots, \mathscr{F}}_{n}), \tag{19}$$

yielding an extended grammar Γ', and still obtain $\mathscr{L}$ as generated language. The advantage is that Γ' allows recognition of the string F in one step, and assignment of a special relaxed semantics to F (instead of relying on the composition of relaxed

semantics of substrings of F through the production rules). This is useful for those operators which do not appear in the list of production rules, but for which we have a tight convex relaxation (or a convex envelope).

3.4 Comparison of relaxed semantics

Let $F \in \mathscr{L}$ represent an n-ary function such that ρ', defined as in (19), is not a production rule of Γ. Define $\mathscr{A}'$ as $\mathscr{A} \cup \{F\}$ and Γ' as Γ with ρ' as an added production rule. Assume that the given relaxed semantics for F in Γ' is included in the computed relaxed semantics for F in Γ (which is usually the case in practice, for otherwise we would not add the "useless" rule ρ' to Γ'), i.e. that, for all $I_1, \ldots, I_\ell \in \mathbb{I}$,

$$\mathscr{R}_{\Gamma'}(F, I_1, \ldots, I_\ell) \subseteq \mathscr{R}_{\Gamma}(F, I_1, \ldots, I_\ell). \tag{20}$$

Theorem 1. *For all strings $T \in \mathscr{L}$ that are functions of p variable symbol arguments and for all $I_1, \ldots, I_p \in \mathbb{I}$, we have $\mathscr{R}_{\Gamma'}(T, I_1, \ldots, I_p) \subseteq \mathscr{R}_{\Gamma}(T, I_1, \ldots, I_p)$.*

Proof. If recognition of T through Γ' never involves rule ρ', both grammars yield the same relaxed semantics. Otherwise, consider the *last* time that ρ' is used on T: then Γ' matches a string $\mathscr{F}$ which is an operator F of n arguments. Let $J_1, \ldots, J_\ell$ be the relaxed semantics assigned to each of the n arguments. Since this the last time ρ' is used, each of the J_i ($i \leq n$) are the same whether we use Γ or Γ', which means that, by (20), $J_{\Gamma'} = \mathscr{R}_{\Gamma'}(F, J_1, \ldots, J_\ell) \subseteq \mathscr{R}_{\Gamma}(F, J_1, \ldots, J_\ell) = J_\Gamma$. By (18), any relaxed semantics involving $J_{\Gamma'}$ will be contained in the same relaxed semantics with $J_{\Gamma'}$ replaced by J_Γ. Thus, if the statement holds from the $(k+1)$-st to the last time rule ρ' is used, the k-th time ρ' is used the argument intervals of the relaxed semantics in Γ' must be contained in the argument intervals of the corresponding relaxed semantics in Γ.

In particular, we have the following.

Corollary 1. *If $F(x_1, x_2, x_3) = x_1 x_2 x_3$ and we assign to F the relaxed semantics given by the trilinear envelopes given in [18, 17], the convex relaxation obtained through Γ' is at least as tight as that obtained through Γ for any mathematical function in $\mathscr{L}$.*

Proof. Assumption (20) holds by definition of convex envelope.

4 Computational results

In this section, we computationally evaluate the tightness of convex relaxations for quadrilinear monomials obtained combining bilinear and trilinear convex envelopes in different ways. Specifically, we consider relaxations of the following four sets:

$$S^{222} = \{(x,w) \in \mathbb{R}^4 \times \mathbb{R}^3 \,|\, x_i \in [x_i^L, x_i^U] \wedge w_1 = x_1x_2, w_2 = w_1x_3, w_3 = w_2x_4\},$$
$$\tilde{S}^{222} = \{(x,w) \in \mathbb{R}^4 \times \mathbb{R}^3 \,|\, x_i \in [x_i^L, x_i^U] \wedge w_1 = x_1x_2, w_2 = x_3x_4, w_3 = w_1w_2\},$$
$$S^{32} = \{(x,w) \in \mathbb{R}^4 \times \mathbb{R}^2 \,|\, x_i \in [x_i^L, x_i^U] \wedge w_1 = x_1x_2x_3, w_2 = w_1x_4\},$$
$$S^{23} = \{(x,w) \in \mathbb{R}^4 \times \mathbb{R}^2 \,|\, x_i \in [x_i^L, x_i^U] \wedge w_1 = x_1x_2, w_2 = w_1x_3x_4\}.$$

In [6] numerical experiments were carried out in order to evaluate the relative tighteness of the four considered relaxations. The comparison was mainly made in terms of volume of the corresponding enveloping polytopes (projected onto $\mathbb{R}^5$ to have comparable results) on a set of randomly generated instances. It showed that the smallest values of volumes correspond to relaxations involving the composition of trilinear and bilinar envelopes, and in particular the best results for more than 80% of the considered instances were obtained using relaxation S^{23}. Numerical experiments on some real-life problems were carried out using a bound evaluation algorithm, whose purpose is to assess the quality of the proposed alternative bounds for quadrilinear terms. This "partial sBB" algorithm at each branching step only records the most promising node and discards the other, thus exploring a single branch up to a leaf. The best bounds were obtained using a relaxation involving a trilinear envelope.

In the present paper, we further investigate the strenght of the proposed relaxations in a sBB algorithm. To that effect, we implemented the computation of the four relaxations for quadrilinear monomials in `COUENNE` [5]. Computational experiments were carried out running `COUENNE` on 7 instances of the Molecular Distance Geometry Problem (MDGP) [11], the problem of finding an embedding $x : V \to \mathbb{R}^3$ of the vertices V of a weighted graph $G = (V,E)$ such that all the edge weights d_{uv} (for $\{u,v\} \in E$) are equal to the Euclidean distances $\|x_u - x_v\|$. The MDGP mathematical programming formulation is:

$$\min_x \sum_{\{u,v\} \in E} \left(\|x_u - x_v\|^2 - d_{uv}^2 \right)^2, \tag{21}$$

a nonconvex NLP involving polynomials of fourth degree. In our experiments we impose a time limit equal to 4 hours. Results were obtained on a 2.4 GHz Intel Xeon CPU of a computer with 8GB RAM shared by three other similar CPU running Linux. For the smallest MDGP instance, the optimal solution is computed within the time limit using all the considered relaxations. A comparison of CPU time is reported in Table 1 and shows that the time needed to solve the problem when relaxation S^{23} is used is 81% smaller than the time needed using S^{222}, that is the second best time to solve the problem. For the other instances, for which the optimal solution is not reached within the time limit, we compare the (lower) bounds obtained with the four relaxations. Results are shown in Table 2. These results confirm the results obtained in [6]. It appears that the best bounds are always obtained using a relaxation involving a trilinear envelope and, in 5 cases out of 6, correspond to relaxation S^{23}. The sBB based on this relaxation gives bounds which are significantly better than the ones obtained using a relaxation based on the composition of bilinear envelopes, in particular on the largest instances. For the first instance in Table 2 the

optimal solution is found with relaxations S^{222} and S^{23} within the time limit. It took 8311.97 seconds in the first case and 7063.73 seconds in the second one.

Instance	S^{222}	$\tilde{S}^{222}$	S^{32}	S^{23}
`lavor3`	311.934	372.306	475.835	**58.0872**

Table 1 Comparison of CPU time (seconds) obtained by running `couenne` with relaxations S^{222}, $\tilde{S}^{222}$, S^{32}, S^{23} on the smallest MDGP instance. The best value is reported in bold face. Solutions were obtained on a 2.4 GHz Intel Xeon CPU of a computer with 8GB RAM shared by three other similar CPU running Linux.

Instance	S^{222}	$\tilde{S}^{222}$	S^{32}	S^{23}
`lavor5`	**228.574**$^{(*)}$	199.864	200.45	**228.574**$^{(*)}$
`lavor6`	93.4905	135.899	84.9467	**144.399**
`lavor7`	2.75184	90.3962	70.9786	**207.255**
`lavor8`	24.5401	95.0223	36.421	**334.968**
`lavor10`	-266.843	-105.584	-91.4539	**93.6579**
`lavor20`	-1571.58	-1215.7	**-589.636**	-1146.5

Table 2 Comparison of lower bounds obtained by running `couenne` with relaxations S^{222}, $\tilde{S}^{222}$, S^{32}, S^{23} on MDGP instances. Bounds were obtained within a 4h time limit. The best values are reported in bold face. The symbol (*) denotes optimal solutions found. Solutions were obtained on a 2.4 GHz Intel Xeon CPU of a computer with 8GB RAM shared by three other similar CPU running Linux.

5 Conclusion

We analyzed four different convex relaxations for quadrilinear monomials, obtained by the composition of the known convex envelopes for bilinear and trilinear monomials. Starting from theoretical as well as computational results given in [6], we further investigated these relaxations. We provided an alternative proof of the fact that a relaxation of k-linear terms that employs a successive use of relaxing bilinear terms (via the bilinear convex envelope) can be improved by employing instead a relaxation of a trilinear term (via the trilinear convex envelope). We computationally evaluated the impact of the tightened convex relaxations in a spatial Branch-and-Bound algorithm on a set of instances of a real-life problem.

References

1. C.S. Adjiman, S. Dallwig, C.A. Floudas, and A. Neumaier. A global optimization method, αBB, for general twice-differentiable constrained NLPs: I. Theoretical advances. *Computers & Chemical Engineering*, 22(9):1137–1158, 1998.
2. F.A. Al-Khayyal and J.E. Falk. Jointly constrained biconvex programming. *Mathematics of Operations Research*, 8(2):273–286, 1983.
3. K.M. Anstreicher. Semidefinite programming versus the reformulation-linearization technique for nonconvex quadratically constrained quadratic programming. Pre-print, Optimization Online, May 2007.
4. X. Bao, N.V. Sahinidis, and M. Tawarmalani. Multiterm polyhedral relaxations for nonconvex, quadratically constrained quadratic programs. *Optimization Methods and Software*, 24:485–504, 2009.
5. P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter. Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods and Software*, 24(4):597–634, 2009.
6. S. Cafieri, J. Lee, and L. Liberti. On convex relaxations of quadrilinear terms. *Journal of Global Optimization*, 47:661–685, 2010.
7. R.M. Karp and C.H. Papadimitrou. On linear characterizations of combinatorial optimization problems. *SIAM Journal on Computing*, 11:620–632, 1982.
8. S. Kim and M. Kojima. Second order cone programming relaxation of nonconvex quadratic optimization problems. *Optimization Methods and Software*, 15:201–204, 2001.
9. L. Liberti. Writing global optimization software. In L. Liberti and N. Maculan, editors, *Global Optimization: from Theory to Implementation*, pages 211–262. Springer, Berlin, 2006.
10. L. Liberti, S. Cafieri, and F. Tarissan. Reformulations in mathematical programming: a computational approach. In A. Abraham, A.-E. Hassanien, P. Siarry, and A. Engelbrecht, editors, *Foundations on Computational Intelligence vol.3*, volume 203 of *Studies in Computational Intelligence*, pages 153–234. Springer, Berlin, 2009.
11. L. Liberti, C. Lavor, A. Mucherino, and N. Maculan. Molecular distance geometry methods: from continuous to discrete. *International Transactions in Operational Research*, 18:33–51, 2010.
12. L. Liberti and C.C. Pantelides. Convex envelopes of monomials of odd degree. *Journal of Global Optimization*, 25:157–168, 2003.
13. L. Liberti, P. Tsiakis, B. Keeping, and C.C. Pantelides. *ooOPS*. Centre for Process Systems Engineering, Chemical Engineering Department, Imperial College, London, UK, 2001.
14. Y. Lin and L. Schrage. The global solver in the LINDO API. *Optimization Methods and Software*, 24:657–668, 2009.
15. J. Luedtke, M. Namazifar, and J. Linderoth. Some results on the strength of relaxations of multilinear functions. University of Wisconsin-Madison. Submitted, 2010.
16. G.P. McCormick. Computability of global solutions to factorable nonconvex programs: Part I — Convex underestimating problems. *Mathematical Programming*, 10:146–175, 1976.
17. C.A. Meyer and C.A. Floudas. Trilinear monomials with positive or negative domains: Facets of the convex and concave envelopes. In C.A. Floudas and P.M. Pardalos, editors, *Frontiers in Global Optimization*, pages 327–352. Kluwer Academic Publishers, Amsterdam, 2003.
18. C.A. Meyer and C.A. Floudas. Trilinear monomials with mixed sign domains: Facets of the convex and concave envelopes. *Journal of Global Optimization*, 29(2):125–155, 2004.
19. C.A. Meyer and C.A. Floudas. Convex envelopes for edge-concave functions. *Mathematical Programming*, 103:207–224, 2005.
20. M. Namazifar, P. Belotti, and A.J. Miller. Valid inequalities, separation, and convex hulls for bounded multilinear functions. In preparation. Presented at MIP 2010, Atlanta, USA, 2010.
21. A. Rikun. A convex envelope formula for multilinear functions. *Journal of Global Optimization*, 10(4):425–437, 1997.
22. H.S. Ryoo and N.V. Sahinidis. A branch-and-reduce approach to global optimization. *Journal of Global Optimization*, 8(2):107–138, March 1996.

23. N.V. Sahinidis and M. Tawarmalani. Baron 8.1.1: Global optimization of mixed-integer nonlinear programs. Users Manual. Available at http://www.gams.com/dd/docs/solvers/baron.pdf, 2008.
24. A. Saxena, P. Bonami, and J. Lee. Convex relaxations of non-convex mixed integer quadratically constrained programs: Extended formulations. *Mathematical Programming B*, 124:383–411, 2010.
25. A. Saxena, P. Bonami, and J. Lee. Convex relaxations of non-convex mixed integer quadratically constrained programs: Projected formulations. *Mathematical Programming*, 2010. Accepted for publication.
26. H.D. Sherali. Convex envelopes of multilinear functions over a unit hypercube and over special discrete sets. *Acta Mathematica Vietnamica*, 22:245–270, 1997.
27. E.M.B. Smith and C.C. Pantelides. A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs. *Computers & Chemical Engineering*, 23:457–478, 1999.
28. F. Tardella. Existence and sum decomposition of vertex polyhedral convex envelopes. *Optimization Letters*, 2:363–375, 2008.
29. F. Tardella. On the existence of polyhedral convex envelopes. In C.A. Floudas and P.M. Pardalos, editors, *Frontiers in Global Optimization*, pages 149–188. Kluwer Academic Amsterdam Publishers, Amsterdam, 2008.
30. M. Tawarmalani and N.V. Sahinidis. Convex extensions and convex envelopes of l.s.c. functions. *Mathematical Programming*, 93:247–263, 2002.
31. M. Tawarmalani and N.V. Sahinidis. Global optimization of mixed integer nonlinear programs: A theoretical and computational study. *Mathematical Programming*, 99:563–591, 2004.