Linearity Embedded in Nonconvex Programs

Leo Liberti

Centre for Process Systems Engineering
Imperial College of Science, Technology and Medicine

(l.liberti@ic.ac.uk)

20 Sept. 2002

Abstract

Nonconvex programs involving bilinear terms often appear more nonlinear than they really are. By a simple symbolic reformulation we can substitute some of the bilinear terms with linear constraints. This has a dramatically improving effect on the efficiency of local or global optimization software acting on the problem.

Keywords: bilinear, convex relaxation, global optimization, reformulation, MINLP, RLT, reduction constraint.

1 Introduction

This paper is concerned with programming problems of the form:

$$\left.\begin{array}{rrcl} \min_x & x^TQx+c^Tx & + & f(x) \\ & Ax & = & b \\ & g(x) & = & 0 \\ & h(x) & \le & 0 \\ & x^L \le & x & \le x^U \end{array}\right\} \quad (1)$$

where $Q=(q_{ij})$ is an $n \times n$ matrix, $x, c, x^L, x^U \in \mathbb{R}^n$, $A=(a_{ij})$ is an $m \times n$ matrix having rank m, $b \in \mathbb{R}^m$, $f:\mathbb{R}^n \to \mathbb{R}$, $g:\mathbb{R}^n \to \mathbb{R}^{m_1}$, $h:\mathbb{R}^n \to \mathbb{R}^{m_2}$. Notice f, g, h are completely arbitrary functions; notice also that we assume $m \le n$ otherwise the feasible region may be empty and the problem have no solution. Such a formulation is very general and encompasses many instances of problems arising from mathematical modelling of real life processes. Notice that although the main point of this paper is the reformulation of continuous NLPs, in section 8 we also explore an application of this research to mixed-integer nonlinear programming problems (MINLPs).

Because the theory developed herein will enable us to substitute some of the bilinear terms with linear constraints, we can restrict our attention to a more standard formulation of bilinear problems:

$$\left.\begin{array}{rrcl} \min_x & x^TQx & + & c^Tx \\ & Ax & = & b \\ & Hx & \le & d \\ & x^L \le & x & \le x^U, \end{array}\right\} \quad (2)$$

where H is a $m' \times n$ matrix and $d \in \mathbb{R}^{m'}$. The results which we shall obtain, however, can then be applied, with minor modifications, to the more general formulation (1). In formulation (2) we assume without loss of generality that A is upper triangular with nonzero entries along the main diagonal $a_{11}, \ldots, a_{mm}$ (we use the term "upper triangular" with a slight abuse of notation since A is not a square matrix; what we mean is that the leftmost $m \times m$ submatrix of A is upper triangular). Since this can be obtained from any $m \times n$ matrix or rank m by a simple application of Gaussian elimination and (possibly) column permutation, it does not restrict the generality of the results. The reason behind this assumption is that the notation in the proofs becomes greatly simplified.

Solving such problems to global optimality using a deterministic approach very often involves the use of Branch-and-Bound (BB) procedures [ADFN98, RS95, SP99, VEH96, KB00, EP97]. In BB-type algorithms, at each iteration upper and lower bound to the objective function are calculated relative to the current region. The overall efficiency of such algorithms depends very much on the quality of the lower bound, which is usually found by locally solving a convex relaxation of the problem relative to the current region: the tighter the relaxation, the better the lower bound is. The present work may be used as a way to improve the tightness of the convex relaxation. Notice that the methods we propose can be introduced as a pre-processing step in a BB framework, and therefore do not add significantly to the computational cost of the BB run.

In this paper we explain the theoretical reasons why some bilinear terms in certain linearly constrained problems can be replaced by newly formed linear constraints. In order to ease our theoretical explanations we shall assume that the bilinear problems (1) we tackle are "dense". In other words we require that most of the possible bilinear terms in the problem variables be present in the problem formulation: the matrix Q is assumed dense. We wish to emphasize here that this theory does not *require* Q to be dense: none of the theoretical results herein becomes false when Q is not dense. Rather, it becomes unlikely that a straightforward application of our methods will yield useful results. Because in a significant proportion of real-life cases requiring a dense Q is an unfair assumption, we address this issue in section 11, and we suggest how to circumvent the problems arising from the sparsity of Q.

There have been a lot of papers in the literature devoted to bilinear problems (2), e.g. [McC76, AKF83, SA92]. In particular the RLT (Reformulation-Linearization Technique) for bilinear problems, proposed in [SA92], can potentially produce the same results as the method explained herein (as will be shown in section 6 below), because the basic ideas are very similar, and in fact the basic idea which forms the basis of this paper, that of multiplying a linear constraint by a variable, has been borrowed from the RLT. The method explained herein contains precise mathematical conditions of when and how to carry out the procedures. The RLT is also a theoretically precise method. However, most practical implementation have to take into account the fact that multiplying *all* linear constraints by *all* variables is overkill. To limit this computational explosion many limiting devices have been described, some precise and some heuristic.

Throughout this paper, we shall implicitly make use of the following elementary facts, which we state here without proofs.

1. The sum of all integers from 1 to n is $\frac{1}{2}n(n+1)$.
2. Given a system $Ax = b$, if we permute the columns of A and the r.h.s. vector b accordingly, we end up with a permuted solution vector x.
3. Given a system $Ax = b$, there may be more than one possible set of basic variables of the system.
4. Given a system $Ax = b$ where A is $m \times n$, upper-triangular with nonzero entries along the main diagonal, and has rank m, the set $\{x_1, \ldots, x_m\}$ is a set of basic variables and $\{x_{m+1}, \ldots, x_n\}$ is a set of nonbasic variables of the system.

2 Problem Reformulation

We reformulate the problem so that all bilinear terms in the objective function are replaced by newly defined variables w, and we add the definition of these variables in terms of the bilinear products in the list of problem constraints.

For each problem variable x_k, let

$$w^k = x_k x = (x_k x_1, x_k x_2, \ldots, x_k^2, \ldots, x_k x_n) \tag{3}$$

so that $w_i^k = x_k x_i$ for all $i \le n$. Let $W = \{w_j^i \mid i,j \le n\}$. Notice that $w_j^i = w_i^j$ for all $i,j \le n$, which implies that $|W| = \frac{1}{2}n(n+1)$. The equivalence relation $(i,j) \sim (j,i)$ on index pairs induces equivalence classes $[i,j] = \{(i,j),(j,i)\}$ on the set of all index pairs. Let w be the vector derived by ordering the elements of W in the natural way (i.e. $w_j^i < w_l^k \Leftrightarrow i < k \vee (i = k \wedge j < l)$). This ordering induces a bijection κ between the equivalence classes $[i,j]$ and a single index $h = \kappa([i,j])$ such that $1 \le h \le |W|$. We can then reformulate problem (2) as follows:

$$\left.\begin{array}{rrcl} \min_{x,w} & p^T w & + & c^T x \\ & Ax & = & b \\ & Hx & \le & d \\ & w_j^i & = & x_i x_j \ \forall i \le j \le n \\ & x^L \le & x & \le x^U \\ & w^L \le & w & \le w^U. \end{array}\right\} \tag{4}$$

where $p = (p_1, \ldots, p_{|W|})$ with $p_h = p_{\kappa([i,j])} = q_{ij} + q_{ji}$ for all $h \le |W|$.

2.1 Definition
*The equality constraints $w_j^i = x_i x_j$, for all $i,j \le n$, are called w-**defining constraints**.*

The vectors $w^L, w^U \in \mathbb{R}^{|W|}$ are the w variable bounds (which can be calculated by means of interval analysis on the w-defining constraints).

This reformulation effectively isolates the bilinear terms in the constraints. Notice that it is in fact more general than formulation (2): given any bilinear problem, where the bilinear terms can be either in the objective function (as in (2)) or in the constraints, a reformulation (4) of the problem is possible. We shall see in the next sections how some of these simple bilinear constraints can be replaced by linear constraints in both w and x variables.

3 Reduction Constraints

Consider one of the linear equation constraints $\sum_{j=1}^n a_{ij} x_j = b_i$ present in problem (4), for some $i \le m$. If we multiply this constraint by x_k (with $k \le n$) we get $\sum_{j=1}^n a_{ij} x_k x_j = b_i x_k$, and since $w_j^k = x_k x_j$, we obtain a linear relationship

$$\sum_{j=1}^n a_{ij} w_j^k - b_i x_k = 0 \tag{5}$$

between x_k and the w^k variables. Eqn. (5) is a valid problem constraint, in the sense that adding it to the problem formulation leaves the feasible region unchanged.

3.1 Definition
The linear equation (5) is called a **reduction constraint**. *Problem variable x_k is called the* **multiplier variable**.

Consider now the whole linear system $Ax = b$. For each variable x_k we can derive a *reduction constraint system* by multiplying the system by x_k:

$$Aw^k - x_k b = \begin{pmatrix} & -b_1 \\ A & \vdots \\ & -b_m \end{pmatrix} \begin{pmatrix} w_1^k \\ \vdots \\ w_n^k \\ x_k \end{pmatrix} = 0 \tag{6}$$

If we assume A to be upper triangular, then system (6) is also upper-triangular. Notice also that the rows of system (6) are $(a_{i1}, \ldots, a_{in}, -b_i)$ for all $i \le m$. Since the rows of A are linearly independent, the rows of system (6) are also linearly independent. Thus system (6) also has rank m.

We now restrict our interest to the following set,

$$C_k = \{(w^k, x) \mid Ax = b \wedge \forall j \le n \ (w_j^k = x_k x_j)\} \tag{7}$$

which forms a superset of the feasible region of problem (4), and we prove that it is equal to the set

$$R_k = \{(w^k, x) \mid Ax = b \wedge Aw^k - x_k b \wedge \forall j \in \{m+1, \ldots, n\} \ (w_j^k = x_k x_j)\}. \tag{8}$$

Notice that C_k is defined by m linear constraints and n bilinear constraints, whereas R_k is defined by $2m$ linear constraints and only $n - m$ bilinear constraints. In other words, a reduction constraint system of rank m can replace m of the n bilinear constraints. More precisely, as shown in lemma 3.3 below, it replaces all the bilinear terms $x_k x_j$ where x_j is a basic variable of the system $Ax = b$; since we are assuming that A is in upper triangular form with nonzero entries along the main diagonal, the basic variables are $x_1, \ldots, x_m$.

3.2 Definition
Given a reduction constraint system $Aw^k - x_k b = 0$ for some $k \le n$, we introduce the **reduction companion system** *$Az^k = 0$, where for all $j, k \le n$ we define $z_j^k = w_j^k - x_k x_j$ (and $z^k = (z_1^k, \ldots, z_n^k)$).*

Note that the definitions of reduction constraint systems and their corresponding reduction companion systems also apply to subsystems of $Ax = b$. We shall make use of this fact in the generalization to sparse systems in section 11.2.

3.3 Lemma
For all $k \le n$, we have $C_k = R_k$.

Proof. The fact that $C_k \subseteq R_k$ is easy to prove: the system $Ax = b$ implies the reduction constraint system, and deleting some of the w-defining constraints just makes the set R_k bigger than C_k. We now show that $R_k \subseteq C_k$. In the reduction constraint system $Aw^k - x_k b = 0$, replace b by Ax (since the relation $Ax = b$ holds) to get $Aw^k - x_k Ax = 0$, which implies $A(w^k - x_k x) = 0$, i.e. (after substitution with the z variables) the reduction companion system $Az^k = 0$. By definition of R_k, we have $w_j^k = x_k x_j$, and hence $z_j^k = 0$, for all j such that $m < j \le n$. Thus we can delete the last $n - m$ columns from the system $Az^k = 0$ to obtain an $m \times m$ system of full rank m having right hand side equal to zero. Such a system has a unique solution $z_j^k = 0$ for all $j \le m$, and hence $w_j^k = x_k x_j$ for all $j \le m$. The result follows. □

By applying the above lemma to problem (4), we see that we can readily substitute some bilinear w-defining constraints with reduction constraint systems.

Next, we will consider the system of all reduction constraint systems,

$$\forall k \le n \ (Aw^k - x_k b = 0), \tag{9}$$

from which we can also derive the companion system in the z variables

$$\forall k \le n \ (Az^k = 0), \tag{10}$$

the superset C of the feasible region of (4), given by

$$C = \{(w, x) \mid Ax = b \wedge \forall k, j \le n \ (w_j^k = x_k x_j)\}, \tag{11}$$

and the set defined by $Ax = b$ and system (9)

$$R = \{(w, x) \mid Ax = b \wedge \forall k \le n \ (Aw^k - x_k b)\}. \tag{12}$$

3.4 Proposition
Assume that the rank of system (10) is $\frac{1}{2}n(n+1)$. Then $C = R$.

Proof. As in the proof of lemma (3.3), the fact that $C \subseteq R$ is obvious. We show that $R \subseteq C$. From system (9), by substituting $b = Ax$ and $z_j^k = w_j^k - x_k x_j$, we get the companion system (10), which has mn equations in $\frac{1}{2}n(n+1)$ variables (because the relation $z_j^k = z_k^j$ holds for all $k, j \leq n$). This system has rank $\frac{1}{2}n(n+1)$ by hypothesis, so it has the unique solution $z_j^k = 0$, i.e. $w_j^k = x_k x_j$, for all $k, j \leq n$. □

By proposition (3.4), as long as the rank of system (10) is $\frac{1}{2}n(n+1)$, we can substitute all the bilinear terms in problem (2) with system (9) without changing the feasible region. In other words, such a bilinear problem can be reformulated precisely as a linear problem.

Unfortunately, however, numerical experiments in this sense seem to point out that the rank of system (10) may only be equal to $\frac{1}{2}n(n+1)$ when $m = n$, which implies that the feasible region of the original problem has at most one point, i.e. when the optimization problem is trivial. In most cases, however, a significant, if not total, reduction in the number of w-defining constraints is possible.

Conjecture
Let $r(m,n)$ be the maximum possible rank of system (10). Then $r(m,n) = \frac{1}{2}m(m+1) + (n-m)m$.

As a corollary to this conjecture, we have that $m \geq n$ is a necessary (but not sufficient) condition for $r(m,n) \geq \frac{1}{2}n(n+1)$, showing that only trivial optimization problems having n as the rank of A may be reformulated as completely linear.

Although we are not able to establish, at the present state of affairs, whether prop. (3.4) is ever actually useful in practice, it nonetheless unearths a crucial relationship between the number of bilinear w-defining constraints that can be disposed of via the introduction of reduction constraints, and the set of basic variables of the companion system (10).

3.5 Theorem
Given a bilinear program (4), if t is the rank of the companion system $\forall k \leq n(Az^k = 0)$, exactly t of the $\frac{1}{2}n(n+1)$ bilinear w-defining constraints are replaced by the reduction constraint systems $\forall k \leq n\ (Aw^k - x_k b = 0)$. More precisely, given a set of basic variables z for the companion system (10), the corresponding w-defining constraints are replaced by the reduction constraint systems (9).

Proof. By adding the reduction constraint systems (9) to the formulation of problem (4) we do not restrict its feasible region, as reduction constraints are obtained by multiplying an existing problem variable by an existing problem constraint (and the feasible region is geometrically closed under such an operation). As in the proofs of lemma (3.3) and prop. (3.4), system (10) can be derived from system (9); notice that the two systems are equivalent, in the sense that one implies the other and vice versa. Let I be a set of index pairs (i,j), where $i \leq j \leq n$, such that $\Gamma = \{z_j^i \mid (i,j) \in I\}$ is a set of basic variables for the companion system (10). If the w-defining constraints $w_j^i = x_i x_j$, for $(i,j) \notin I$, are present in problem (4), then $z_j^i = 0$ for all nonbasic variables z_j^i of the companion system (10). Thus, by elementary linear algebra, system (10) has a unique solution $z_j^i = 0$ for all $i, j \leq n$. This implies that formulation (4) need only contain the w-defining constraints corresponding to a set of nonbasic variables of system (10), whereas all the other bilinear constraints are implied in reduction constraint systems (9). Furthermore, because $|\Gamma| = t$ for any set of basic variables Γ (i.e. the number of basic variables of a system is equal to the rank of the system), we conclude that we can substitute t out of $\frac{1}{2}n(n+1)$ possible w-defining constraints. □

Thus, in order to reduce the number of bilinear constraints in a programming problem (4), we need to find a set of basic variables of system (10). For notational convenience, we refer to system (10) either as $\forall k \leq n(Az^k = 0)$ or in the more compact way $Bz = 0$ where B is an $mn \times \frac{1}{2}n(n+1)$ matrix and $z = (z_1^1, \ldots, z_n^1, z_2^2, \ldots, z_n^2, z_3^3, \ldots, z_n^3, \ldots, z_n^n)$. The problem is now to determine what B looks like, what is its rank and how to find a set of basic variables for the system $Bz = 0$. Prop. (3.4) supplies the answer in case B is equivalent to a square matrix. So what happens when the rank of B is less than $\frac{1}{2}n(n+1)$? From lemma (3.3), one might think that we should be able to eliminate mn bilinear terms; but in fact, this is an upper bound to the best case. The actual answer depends on the properties of the matrix A and its companion matrix B. It is always possible to reduce matrix B to row echelon form via Gaussian elimination and immediately calculate its rank and a set of basic and nonbasic variables: we then have to keep the bilinear relations corresponding to the nonbasic z variables, whereas we can discard those that stem from the basic z variables. However, if the size of A is big, the size of B may be prohibitively huge. Therefore, it would be desirable to be able to find out which bilinear constraints we can eliminate just by looking at the matrix A.

Before carrying on, we need some notation. For any matrix A, let A_- be the matrix A without the first row, and A^- be the matrix A without the first column. Similarly, for any vector v, let v^- be the vector v without the first element.

Consider system (10). As we have seen in the proofs of the theorems above, this system corresponds to (9). We start with $k = 1$, i.e. with the system $Az^1 = 0$. This will allow us to eliminate constraints $w^1 = x_1 x_j$ for all $j \leq m$. However, when $k = 2$, i.e. $Az^2 = 0$, we notice that in the first equation of the system, namely $a_{11}z_1^2 + \cdots + a_{1n}z_n^2 = 0$, the leading variable is z_1^2, which is by definition equal to z_2^1. But in the case $k = 1$ we had already derived the relation $z_2^1 = 0$, which implies $z_1^2 = 0$. Thus for $k = 2$ we can eliminate the first column of the system $Az^2 = 0$, which is equivalent to solving the system

$$A^-(z^2)^- = 0. \tag{13}$$

This has m equations but only $n-1$ variables. Furthermore, because the rank of A is m, the rank of A^- can only be m or $m-1$. If the rank of A^- is m, we need only keep $n-m-1$ bilinear equations (those corresponding to the nonbasic variables of system (13)) in the definition of the feasible region of problem (4) in order to reduce system (13) to a square $m \times m$ system of rank m (which then has a unique solution of $z^2 = 0$, thus implying the rest of the bilinear constraints). Conversely, if the rank of A^- is $m-1$, we need to keep $n-m$ bilinear constraints (those corresponding to the nonbasic variables of system (13) in order to make system (13) square and of full rank. Thus, according as to whether matrix A conserves its rank whilst discarding its first i columns, we need only keep a number of bilinear constraints varying between $n-m$ (in the worst case) and $n-m-i$ (in the best case). The process of discarding the leftmost column of the matrix (and keeping the appropriate set of bilinear constraints) goes on until its rank is equal to the number of its columns. From then on the system determines a unique solution $z^i = 0$ which allows us to replace all the remaining bilinear constraints.

By following the reasoning above, we can replace from a minimum of $\frac{1}{2}m(m+1)$ to a maximum of mn bilinear terms with reduction constraints. The algorithm presented in section 4.2 is a precise description of the process explained above, and will identify the bilinear constraints in the problem that must be kept and some of those which the reduction constraint systems can replace. Although this procedure may find a set of bilinear constraints to keep which has the same size as the set of nonbasic variables of the companion system, there are cases when this is not true. So whereas finding the nonbasic variables of the companion system will identify the minimal set of bilinear constraints to keep, the procedure above might result in a slightly larger set.

To see why this limitation holds, let Ω be the set of constraints in the companion system: for each $\Lambda \subseteq \Omega$ let $\xi(\Lambda)$ be the set of nonbasic variables of the system of constraints in Λ, assuming this system transformed so that it is in upper-triangular form with nonzero entries along the main diagonal.

3.6 Lemma
Let $\lambda \in \mathbb{N}$ and $\{\Lambda_i \mid i \leq \lambda\}$ be any covering of Ω, such that $\bigcup_{i=1}^{\lambda} = \Omega$, having size λ (where the Λ_i may or may not be pairwise disjoint). Then $|\xi(\Omega)| \leq \sum_{i=1}^{\lambda} |\xi(\Lambda_i)|$.

In other words, the number of nonbasic variables of Ω is less then or equal to the sum of the numbers of nonbasic variables in each subsystem of the covering. The proof of the above lemma is an exercise in linear algebra; it basically rests on the fact that when one adds equations to a linear system the number of nonbasic variables of the system decreases.

Because of lemma 3.6, finding the nonbasic variables of each of the $Az^k = 0$ systems in the companion system (10), even whilst imposing the condition $z^i_j = z^j_i$ for all $i, j \leq n$ during the process, will gather, in general, a higher number of variables than if we just looked for the nonbasic variables of the companion system (10) straight away.

Although the theoretical considerations drawn in this section were inferred from problem formulation (2), they actually apply to the more general formulation (1): given any problem (1), in fact, it is easy to isolate the bilinear terms in w-defining constraints of the kind considered above. For example, given a constraint $\sin(x_1 x_2) \leq 0$ in problem (1) we can add the w-defining constraint $w^1_2 = x_1 x_2$ and reformulate the constraint as $\sin(w^1_2)$. The application of the theory is then straightforward.

4 Finding Nonbasic Variables of the Companion System

In this section we shall explain how to determine the set of nonbasic variables of the companion system (10), which in turn leads to the set of bilinear terms that is necessary to keep in the problem formulation. We will present two methods, one of which is optimal (i.e. it finds a set consisting precisely of all nonbasic variables) but makes use of an $mn \times \frac{1}{2}n(n+1)$ matrix, which can be prohibitively large if m, n are large. The second method finds a set of variables containing a complete set of nonbasics, but which may also include some basic variables. The second method, however, has the advantage of only requiring the $m \times n$ matrix A.

System (9) (which consists of all possible reduction constraints systems) has a special structure. It can be described as a system $R(w; x) = 0$ where $R = (B|R')$ is an $mn \times \frac{1}{2}n(n+3)$ matrix such that:

$$B = \left(\begin{array}{ccccccccccccccc}
a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & 0 & & & \cdots & & & & & & \\
\vdots & & & & \vdots & & & & & & & & & & \\
a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} & 0 & & & \cdots & & & & & & \\
 & a_{11} & 0 & \cdots & 0 & a_{12} & a_{13} & \cdots & a_{1n} & & 0 & \cdots & & & \\
 & \vdots & & & & \vdots & \vdots & & \vdots & & & & & & \\
 & a_{m1} & 0 & \cdots & 0 & a_{m2} & a_{m3} & \cdots & a_{mn} & & 0 & \cdots & & & \\
 & & a_{11} & 0 & \cdots & 0 & a_{12} & 0 & \cdots & 0 & a_{13} & \cdots & a_{1n} & 0 & \cdots \\
 & & \vdots & & & & \vdots & & & & \vdots & & \vdots & & \\
 & & a_{m1} & 0 & \cdots & 0 & a_{m2} & 0 & \cdots & 0 & a_{m3} & \cdots & a_{mn} & 0 & \cdots \\
 & & & \ddots & & & & \ddots & & & & \ddots & & & \\
 & & & & a_{11} & & & & a_{12} & & & & a_{13} & & a_{1n} \\
 & & & & \vdots & & & & \vdots & & & & \vdots & & \vdots \\
 & & & & a_{m1} & & & & a_{m2} & & & & a_{m3} & & a_{mn}
\end{array}\right)$$

$$
R' = \begin{pmatrix}
-b_1 & 0 & \dots & & & & 0 \\
\vdots & & & & & & \vdots \\
-b_m & 0 & \dots & & & & 0 \\
 & -b_1 & 0 & \dots & & & 0 \\
 & \vdots & & & & & \vdots \\
 & -b_m & 0 & \dots & & & 0 \\
 & & -b_1 & 0 & \dots & & 0 \\
 & & \vdots & & & & \vdots \\
 & & -b_m & 0 & \dots & & 0 \\
 & & & \ddots & & & \vdots \\
 & & & & & & -b_1 \\
 & & & & & & \vdots \\
 & & & & & & -b_m
\end{pmatrix}
$$

and $(w;x) = (w_1^1, \dots, w_n^1, w_2^2, \dots, w_n^2, \dots, w_n^n, x_1, \dots, x_n)$. Notice that $A = (a_{ij})$ does not need to be in upper-triangular form for this to hold.

The matrix B corresponds to system (10) and can be obtained from the matrix R above just by discarding the rightmost n columns involving the b_i's. Matrix B has a very special shape: in each of its n contiguous sets of m rows it contains a copy of the matrix A, in what amounts to an *almost block-diagonal* form. If we did not consider the relation $w_j^i = w_i^j$ the resulting matrix would be completely block diagonal, with A in each diagonal block. Instead, almost all columns of B, apart from those corresponding to the "pure quadratic" $w_i^i = x_i^2$ variables, contain two columns of A. More precisely, the column of B corresponding to w_j^i contains the two columns of A where a_{ij} and a_{ji} are located.

4.1 Lemma
For any $i, j \le n$ let $\mu(i,j) = (j-1)m + i$ and $\nu(j) = \frac{1}{2}j(2n - j + 1)$. The entry of B corresponding to row $\mu(i,j)$ and column $\nu(j)$ is a_{ij}. Furthermore, all the other entries of column $\nu(j)$ are zeroes.

Proof. It is easy to see that for each $j \le n$, rows $(j-1)m + 1$ to jm are a contiguous block of m rows containing a copy of A "laid out" on various columns. In particular, the entries of row $\mu(i,j)$ are $a_{i1}, \dots, a_{in}$ (in this order). Column $\nu(j)$ corresponds to to the variable w_j^j: ordering the w variables in their natural order

$$w = (w_1^1, \dots, w_n^1, w_2^2, \dots, w_n^2, \dots, w_{n-1}^{n-1}, w_n^{n-1}, w_n),$$

we see that w_j^j occurs at the $n + (n-1) + \ldots + (n - (j-1))$-th position. This means $\sum_{l=1}^{j}(n - l + 1) = \frac{1}{2}j(2n - j + 1) = \nu(j)$. Hence, because the $\nu(j)$ column corresponds to a "pure quadratic" variable w_j^j, it only contains one column of A: the column where a_{jj} is located. The result follows. □

4.1 Explicit Construction of the Companion System

The following algorithm will construct the matrix $B = (b_{ij})$ from the matrix A (which does not need to be in upper-triangular form in the following algorithm).

```
initialize B = 0;
for i = 1 to n {
    for j = 1 to m {
        l = 1;
        k = i;
        s = (i − 1)m + j;
        b_sk ← a_jl;
        l ← l + 1;
        for t = 1 to i − 1 {
            k ← k + n − t;
            b_sk ← a_jl;
            l ← l + 1;
        }
        for t = l to n {
            k ← k + 1;
            b_sk ← a_jt;
        }
    }
}
```

Constructing matrix B from A is useful because by theorem (3.5), finding the nonbasic variables of matrix B will immediately identify the minimal set of w-defining constraints to keep in the formulation of the problem.

4.2 Implicit Search of Nonbasics of the Companion System

The algorithm presented below identifies a set of variables in the companion system (10) which contains a complete set of nonbasic variables of (10). This algorithm follows the theoretical discussion in section 3 (p. 6) and has the limitation explained in lemma 3.6.

For any matrix A let $g(A)$ be the application of Gaussian elimination to A (i.e. $g(A)$ is A reduced to row echelon form, with a possible column permutation to ensure the main diagonal entries are nonzero), let $\mathrm{rk}(A)$ be the rank of A and $\mathrm{cols}(A)$ the set of column vectors of A. For a system $Ax = 0$ let $\mathrm{nonbasic}(A, x)$ be the set of nonbasic variables of the system. Notice that this algorithm does not apply to the particular case where A is such that $m = n$.

```
keep relations from nonbasic(A, z^1);
r = 0;
l = rk(A);
for k = 2 to n {
    h = rk(A^-);
    if h = |cols(A)| − 1 {
        stop;
    }
    if h = l {
        A ← g(A^-);
        r ← r + 1;
        keep relations from nonbasic(A, z^k);        (i)
    } else if h = l − 1 {
        A ← A^-_-;
        keep relations from nonbasic(A, z^k);        (ii)
        l ← l − 1;
    }
}
```

The instruction "keep relations from nonbasic(A, z^k)" means that the bilinear constraints $w_j^k = x_k x_j$ deriving from setting the nonbasic variables z_j^k to zero should be kept in the formulation of the problem. Notice that the counter r is not used directly in the algorithm above; however, it makes it possible to keep track of the number of bilinear constraints that have to remain in the formulation. In step (i) $n-k-r$ bilinear constraints have to be kept; in step (ii) $n-k-r+1$ have to be kept.

At each iteration, this algorithm takes away the leftmost column of A (i.e. the column corresponding to variables z_j^k where $k > j$, which had appeared in earlier steps as z_k^j). If the matrix preserves its rank, Gaussian elimination (with a possible column permutation) is performed on it so that the matrix becomes upper-triangular again, and the nonbasic variables of the system can be identified (and the w-defining constraints deriving from setting the nonbasic variables to zero kept in the problem formulation). If the rank of the matrix decreases, it means that the first row is linearly dependent on all the other rows, so by removing the first row we obtain a linearly independent system of rank $m-1$ which is already in upper-triangular form (because the matrix was in upper-triangular form before removing the first column and the first row). Hence we can immediately identify the nonbasic variables of the system and keep the deriving w-defining constraints in the problem formulation. The algorithm terminates when the system is square, as we need to keep no more w-defining constraints after that.

Let O_r be the complexity of the algorithm for calculating the rank of a matrix, and let O_g be the complexity of the algorithm for performing Gaussian elimination. The complexity of the above algorithm is then $O(nO_rO_g)$. Typically, O_r and O_g are polynomial in the sizes of the matrices they operate on, so the whole algorithm is a polynomial time algorithm.

5 Convex Relaxation

Having replaced as many as possible of the bilinear constraints in problem (4) with linear reduction constraints, we obtain a problem in the following form:

$$\left.\begin{array}{rrcl}
\min_{x,w} & p^T w & + & c^T x \\
 & Ax & = & b \\
 & Hx & \le & d \\
\exists I \times J \subset \mathbb{N}^2 \ \forall (i,j) \in I \times J & w_j^i & = & x_i x_j \\
\forall k \le n & Aw^k - x_k b & = & 0 \\
 & x^L \le & x & \le x^U \\
 & w^L \le & w & \le w^U,
\end{array}\right\} \qquad (14)$$

where the set $I \times J$ is empty if the rank of system (10) is $\frac{1}{2}n(n+1)$ and is determined by the nonbasic variables of the companion system otherwise. Supposing $I \times J$ is non-empty, problem (14) is nonconvex. Because in many global optimization techniques (like e.g. Branch-and-Bound) we need a convex relaxation of the problem, we can provide that by replacing the bilinear constraints with their respective tightest convex relaxations.

A linear relaxation for bilinear terms $w_j^i = x_i x_j$ was introduced in [McC76], and later proved to be the convex envelope for such bilinear terms [AKF83]. This envelope is widely used by the community (e.g. [Smi96, SP99, Adj98]). It consists of the following planar inequalities:

$$\begin{array}{rcl}
w_j^i & \ge & x_i^L x_j + x_j^L x_i - x_i^L x_j^L, \\
w_j^i & \ge & x_i^U x_j + x_j^U x_i - x_i^U x_j^U, \\
w_j^i & \le & x_i^U x_j + x_j^L x_i - x_i^U x_j^L, \\
w_j^i & \le & x_i^L x_j + x_j^U x_i - x_i^L x_j^U.
\end{array}$$

Furthermore, it is clear from Euclidean plane geometry that a linear convex relaxation for the term

$w_i^i = x_i^2$ can be obtained by employing the secant and the tangents at the interval endpoints:

$$\begin{array}{rcl} w_i^i & \leq & (x_i^L + x_i^U)x_i - x_i^L x_i^U, \\ w_i^i & \geq & 2x_i^L x_i - (x_i^L)^2, \\ w_i^i & \geq & 2x_i^U x_i - (x_i^U)^2 \\ w_i^i & \geq & 0. \end{array} \tag{15}$$

The above elementary relaxation is known as the "secant" relaxation for quadratic terms. This relaxation can be further tightened by considering inequalities $(x_i - a)^2 \geq 0$ for $a \in (x^L, x^U)$, i.e.

$$w_i^i \quad \geq \quad 2ax_i - a^2.$$

All these inequalities tighten the relaxation, and in fact if one considers the set of all such inequalities, as a spans the interval $[x^L, x^U]$, they define the same set as the inequality $w_i^i \geq x_i^2$.

By applying the method of reduction constraints together with McCormick's and secant relaxations for the possibly remaining bilinear terms, we end up with a convex relaxation of the original bilinear problem (2). For notational convenience when we carry out the discussion of the RLT, we shall refer to the conjunction of these methods (reduction constraints, McCormick and secant relaxations) as the RCMS method.

6 Comparison with RLT

In order to form a linear convex relaxation of problem (2), the RLT (Reformulation-Linearization Technique, [SA92]) applied to bilinear problems considers the following sets:

- the bound factor set $B_F = \{x_i - x_i^L \mid i \leq n\} \cup \{x_i^U - x_i \mid i \leq n\}$;
- the constraint factor set $C_F = \{\sum_{j=1}^n a_{ij}x_j - b_i \mid i \leq m\}$.

Note that for each $b \in B_F$ the constraint $b \geq 0$ is a valid problem constraint, and so is $c = 0$ for all $c \in C_F$.

The RLT procedure for forming the convex relaxation consists in creating new linear valid constraints (reformulation step) by multiplying together bound factors and constraint factors as follows:

1. for all $b_1, b_2 \in B_F$, $b_1 b_2 \geq 0$ is a valid constraint (generation via bound factors);
2. for all $b \in B_F$ and for all $c \in C_F$, $bc = 0$ is a valid constraint (mixed generation);
3. for all $c_1, c_2 \in C_F$, $c_1 c_2 = 0$ is a valid constraint (generation via constraint factors).

Having created all these new constraints, we define new variables $w_j^i = x_i x_j$ for all i, j between 1 and n, and we substitute them whenever a bilinear product appears in problem (2) or in the newly generated constraints (linearization step). Assuming there are t bilinear terms in the problem with the newly added constraints, we end up with a linear relaxation whose variable vector (x, w) is in $\mathbb{R}^{n+t}$. Let S_F be the region defined by the newly generated constraints. The linear convex relaxation of (2) is as follows:

$$\left.\begin{array}{lrcl} \min_x & c^T x & + & p^T w \\ & Ax & = & b \\ & Hx & \leq & d \\ & (x, w) & \in & S_F \\ & x^L \leq & x & \leq x^U \\ & w^L \leq & w & \leq w^U \end{array}\right\} \tag{16}$$

where $w^L, w^U \in \mathbb{R}^t$ are the variable bounds on the w variables (obtained by simple interval arithmetic on the bounds of the x variables via the defining relations $w^i_j = x_i x_j$).

We claim that the convex relaxations obtained with RCMS and with RLT as described above are identical, in the sense that their feasible regions are exactly the same; however, the RLT procedure generates too many constraints, i.e. it generates all possible factor products without discerning where it could be beneficial and where in fact it is not useful. As Epperly put it, "The difficulty with [the RLT] is that the LP size grows exponentially with the number of constraints. If supplemented by a method to determine which constraints are necessary, this technique would be much more useful" ([Epp95], p. 23).

In order to show that the two methods generate relaxations having identical feasible regions, we show that each constraint generated by the RLT is either also generated by the RCMS or it is a linear combination of constraints generated by the RCMS, and vice versa.

Consider the RLT generation via bound factors. For given problem variables x_i, x_j, with $i \neq j$, we can derive the following constraints:

$$\begin{aligned}
(x_i - x_i^L)(x_j - x_j^L) &\geq 0 \\
(x_i - x_i^L)(x_j^U - x_j) &\geq 0 \\
(x_i^U - x_i)(x_j - x_j^L) &\geq 0 \\
(x_i^U - x_i)(x_j^U - x_j) &\geq 0,
\end{aligned}$$

which are equivalent to:

$$\begin{aligned}
x_i x_j &\geq x_i^L x_j + x_j^L x_i - x_i^L x_j^L \\
x_i x_j &\leq x_i^L x_j + x_j^U x_i - x_i^L x_j^U \\
x_i x_j &\leq x_i^U x_j + x_j^L x_i - x_i^U x_j^L \\
x_i x_j &\geq x_i^U x_j + x_j^U x_i - x_i^U x_j^U,
\end{aligned}$$

which in fact are exactly the McCormick relaxation inequalities for the term $x_i x_j$. Notice that in the RCMS method we generate such inequalities for all bilinear terms left in the problem after the reduction constraints reformulation procedure, whereas in the RLT this limitation (i.e. generate such constraints only when the term $x_i x_j$ is present in the problem) is not explicitly stated. Thus the RLT generates more bound factor products than is really necessary.

Whenever $i = j$ we can derive the following constraints:

$$\begin{aligned}
(x_i - x_i^L)(x_i^U - x_i) &\geq 0 \\
(x_i - x_i^L)^2 &\geq 0 \\
(x_i^U - x_i)^2 &\geq 0,
\end{aligned}$$

which are equivalent to:

$$\begin{aligned}
x_i^2 &\leq (x_i^L + x_i^U)x_i - x_i^L x_j^U \\
x_i^2 &\geq 2x_i^L x_i - (x_i^L)^2 \\
x_i^2 &\geq 2x_i^U x_i - (x_i^U)^2,
\end{aligned}$$

which, together with the (obvious) constraint $x_i^2 \geq 0$, form exactly the secant relaxation for quadratic terms, which is considered in the RCMS. Again, the RLT potentially generates more of these constraints than is generally necessary.

Consider now the RLT mixed generation (multiplying one bound factor by one constraint factor). For any variable index l and constraint index i we can derive the following constraints:

$$\begin{aligned}(x_l - x_l^L)(\sum_{j=1}^n a_{ij}x_j - b_i) &= 0 \\ (x_l^U - x_l)(\sum_{j=1}^n a_{ij}x_j - b_i) &= 0,\end{aligned}$$

which, on substituting $w_j^l = x_l x_j$, are equivalent to:

$$\begin{aligned}(\sum_{j=1}^n a_{ij}w_j^l - b_i x_l) - x_l^L(\sum_{j=1}^n a_{ij}x_j - b_i) &= 0 \\ (\sum_{j=1}^n a_{ij}w_j^l - b_i x_l) - x_l^U(\sum_{j=1}^n a_{ij}x_j - b_i) &= 0.\end{aligned}$$

Now, the constraint $\sum_{j=1}^n a_{ij}w_j^l - b_i x_l = 0$ is a reduction constraint in the RCMS method, and the RCMS theory requires that all such constraints be generated. Obviously, the constraint $\sum_{j=1}^n a_{ij}x_j - b_i = 0$ is already part of the set of linear constraints, so subtracting a multiple of it from a reduction constraint does not add new information to the problem. In short, whenever this kind of constraints are generated by the RLT, they are a linear combination of constraints which the RCMS creates.

Lastly, consider the RLT constraint factor generation: for constraint indices i, k we can derive the following constraint:

$$(\sum_{j=1}^n a_{ij}x_j - b_i)(\sum_{l=1}^n a_{kl}x_l - b_k) = 0,$$

which is equivalent to:

$$(\sum_{j=1}^n a_{ij}x_j(\sum_{l=1}^n a_{kl}x_l - b_k)) - b_i(\sum_{l=1}^n a_{kl}x_l - b_k) = 0.$$

Because the reduction constraints of the form $x_j(\sum_{l=1}^n a_{kl}x_l - b_k) = 0$ are created by the RCMS, the constraint above does not add new information to the relaxation.

The reasoning above has shown that the relaxation obtained via the RLT is no better than that obtained via the RCMS. By using the same algebraic relations above, we can also show the converse: each reduction constraint, McCormick relaxation and secant relaxation can be obtained by combining bound and constraint factors. However, because of the mathematical theory behind the creation of reduction constraints, the RCMS only creates those constraints which serve the purpose of reducing the number of nonlinear terms in the problem. The RLT on the other hand, if not used with some common-sense rule of thumb, creates *all* possible combinations of the bound and constraint factors, which is useless in most instances.

7 Reduction Constraints from Systems of Inequalities

In this section we shall explore some of the possibilities of using an inequality system $Hx \leq d$ present in formulation (2) in order to create some more reduction constraints. Most of the material in this section is not analysed in great detail: it should be seen as a collection of ideas for further research.

7.1 Reduction Inequality Constraints

The first possibility is to create reduction inequality constraints. Since for each $k \leq n$, the translated variable $x_k - x_k^L$ (where x_k^L is the lower bound of x_k) is always nonnegative, by multiplying $Hx \leq d$ and $x_k - x_k^L$ we get a valid system of relations $(x_k - x_k^L)Hx \leq (x_k - x_k^L)d$, that is $x_k Hx - x_k d - (x_k^L Hx - x_k^L d) \leq 0$. In terms of the w variables, this means

$$Hw^k - x_k d - x_k^L(Hx - d) \leq 0. \tag{17}$$

The same can be done with the multiplicative factor $x_k^U - x_k$, as it is always nonnegative. These are generally valid cuts for the convex relaxation of the problem obtained by substituting each of the w-defining constraints with their convex relaxations. Thus, they can be kept in the formulation of the convex relaxation of the problem. The RLT also generates systems like (17) from linear inequalities in the problem. It is an open problem whether some theoretical criterion for the generation of all useful (as opposed to just "all") inequality reduction constraints exists: at the present state of the matter this type of generation of reduction inequality constraints is still in the realm of heuristics (although some work with special reference to 0-1 mixed-integer programs has been carried out [SSA00]).

7.2 Deriving Equations from Inequalities

There are other ways to treat the inequality constraint system $Hx \leq d$. A natural observation is that any inequality can be transformed into an equation via the introduction of slack variables. Let $s = (s_1, \ldots, s_{m'})$, where $s_i \geq 0$ for all $i \leq m'$. Recall from formulation (2) that m' is the number of rows and the rank of matrix $H = (h_{ij})$, and the number of components of the right hand side vector d. We can then write $Hx \leq d$ as a system of equations

$$Hx + s = d, \tag{18}$$

together with the bounds $\forall i \leq m' \ (s_i \geq 0)$ on the s variables. First of all notice that since the i-th equation in system (18) depends on a variable s_i and none of the other equations in the system depend on it (and obviously none of the equations in the system $Ax = b$ depend on it either), the combined system $Ax = b \wedge Hx + s = d$ is a linearly independent system with $m + m'$ equations in $n + m'$ variables. Unfortunately just simply deriving reduction constraints directly from system (18) via multiplication of system (18) by each of the problem variables x would not work: we would end up with $m'n$ new w-defining constraints involving the s and x variables (all bilinear products $x_k s_i$ with $k \leq n, i \leq m'$) and a system of $m'n$ new reduction constraints (each of the n problem variables x multiplies system (18) which has m' equations). This new reduction constraint system would have a rank $r \leq m'n$, as we have seen in other sections of this paper. Thus we would not be able to eliminate more bilinear terms, with this method, than the ones we would need to create in order to carry it out.

7.3 Eliminating Slack Variables

If we managed to reduce the number of slack variables required to make $Hx \leq d$ into a system of equations, we might still be able to carry out the normal reduction constraints creation procedure and find we can substitute a greater number of bilinear terms than we need to create. To this purpose, observe that given a point x^* satisfying $Hx^* \leq d$, the slack variables s can be seen as a distance between x^* and the hyperplanes $P_1, \ldots, P_{m'}$ defined by $Hx = d$. Let T be the linear hypersurface defined by $Ax = b$. We can define the distance $s_i = \rho_T(x, P_i)$ between T and P_i at a point $x \in T$ as the length of the shortest segment perpendicular to T at the point x and delimited by the hyperplane P_i.

Because feasible points of problem (2) cannot be at just any distance from $Hx = d$, but are constrained to belong to the hypersurface T defined by $Ax = b$, it is possible, in theory, to find relations between the s variables that could help reduce their number.

7.4 Equidistant Hyperplanes

Suppose there is a subsystem

$$H'x \le d' \tag{19}$$

of $Hx \le d$, having $m'' \le m'$ inequalities, such that the hyperplanes defined by

$$H'x = d' \tag{20}$$

lie at the same distance s' from the hypersurface T defined by $Ax = b$. In that case s' would be the only required slack variable to make the subsystem (19) a system of equations, for the solution to problem (2) would necessarily be equidistant from each of the hyperplanes in system (20). Thus, from system

$$H'x + s' = d', \tag{21}$$

where $H' = (h'_{ij})$, $d' = (d'_1, \ldots, d'_{m''})$ and $s' \ge 0$, we can potentially derive more reduction constraints than we need to create new bilinear w-defining constraints between the x variables and s' (whether we actually can or not depends, as in the sections above, largely on the structure of H'). There is a *caveat*, however. Although system (21) is linearly independent (because the original system $Hx \le d$ was supposed by hypothesis to be defined by linearly independent hyperplanes, and adding a column to a linearly independent system preserves the linear independence), the combined system of equations $Ax = b \wedge H'x + s' = d'$ might not be linearly independent. If we eliminate s' from system (21) via elementary row operations we might end up with some nonzero rows in the x variables which are linearly dependent on $Ax = b$.

Carrying on with the argument, we need to find a maximal set $\mathcal{P}$ of hyperplanes in system (20) having the following property:

$$\forall P, Q \in \mathcal{P}\ \forall x \in T\ (\rho_T(x, P) = \rho_T(x, Q)).$$

Notice that there may be more than one such maximal sets, and in fact we are interested in finding them all as long as they contain at least two planes, as from each of these sets we can obtain a subsystem like (21) (obviously each of these subsystem would have a separate slack variable). Fig. 1 shows the geometrical interpretation in three dimensions of what we are trying to achieve: P, Q are planes which are equidistant from line T (the z coordinate axis). D and D' are shortest segments perpendicular to T and delimited by P and Q respectively. Notice P and Q are such that $l(D) = l(D')$ (where $l(\cdot)$ indicates the length of the segment) for whatever $x \in T$ we choose as a common segment delimiter.

Constructing an algorithm based on this idea is outside of the scope of this paper. However, as the material explained in this section rests on nothing more than linear n-dimensional Euclidean geometry, such an algorithm should not be exceedingly hard to devise.

8 Application to Mixed-Integer Programming

We have shown in this paper how to substitute bilinear terms with linear constraints: this implies that the convex relaxation of the problem is much tighter, since there are less nonconvex terms to relax. This, in turn, suggests an application of this theory to mixed-integer programming. It is a well-known fact that in the field of global optimization of mixed-integer nonlinear programs (MINLPs), a continuous reformulation of the MINLP can be obtained by relaxing 0-1 integer variables z to continuous variables $0 \le \bar{z} \le 1$ and by including the nonconvex constraint

$$\bar{z} - \bar{z}^2 = 0 \tag{22}$$

in the formulation. Smith noted ([Smi96], p.209-210; [SP97]) that the "secant" convex relaxation (15) of the quadratic term in constraint (22) is such that the integrality requirements of z would be completely lost when solving the convex relaxation of the problem. Thus, whereas this kind of continuous reformulation

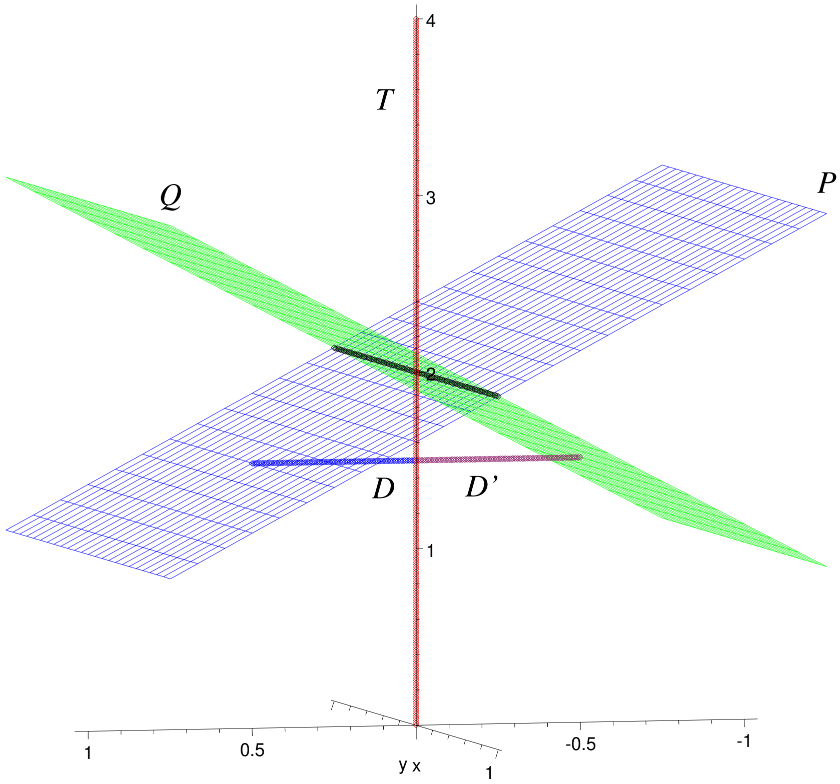

Figure 1: Hyperplanes P, Q are equidistant from T (D, D' have the same length).

for MINLPs works well when performing NLP local optimization, it might not perform so efficiently in global optimization with Branch-and-Bound techniques which require a convex relaxation of the problem.

By reformulating constraint (22) to

$$\begin{aligned} \bar{z} - \bar{w} &= 0 \\ \bar{w} &= \bar{z}^2 \end{aligned}$$

we get a w-defining constraint for the quadratic term $\bar{z}^2$. Supposing the hypotheses of prop. (3.4) were satisfied, and we managed to substitute all bilinear terms with reduction constraints, then the convex relaxation of the MINLP would embed the vital information about the integrality of z, and the MINLP could thus be solved to global optimality via straightforward LP or NLP techniques. But even in the case where we cannot substitute all bilinear terms with reduction constraints, we might at least be able to substitute the quadratic term $\bar{z}^2$ with a reduction constraint. If that happens, the convex relaxation would have a higher chance to be solved to integrality of $\bar{z}$ than it would otherwise, since the relaxation is much tighter.

This concept can be generalized to any kind of integer variable. Supposing z can take the integer values $\alpha_1, \ldots, \alpha_n$, we can reformulate z to a continuous variable $\bar{z}$ such that $\min_{i \le n} \alpha_i \le \bar{z} \le \max_{i \le n} \alpha_i$, and include the following nonconvex constraint:

$$(\bar{z} - \alpha_1) \cdots (\bar{z} - \alpha_n) = 0 \tag{23}$$

in the formulation of the problem. By reformulating this constraint to

$$\begin{aligned}
\bar{w}_{n-1} &= \bar{z}-\alpha_n \\
\bar{w}_{n-2} &= \bar{w}_{n-1}(\bar{z}-\alpha_{n-1}) \\
&\vdots \\
\bar{w}_1(\bar{z}-\alpha_1) &= 0
\end{aligned}$$

we end up with $n-1$ bilinear products $\bar{w}_i\bar{z}$ for all $i \leq n-1$. Again depending on the structure of the problem we might be able to eliminate such bilinear terms, thus making sure that the integrality requirements of z have more chances of being carried over to the convex relaxation of the MINLP.

9 Examples

In this section we shall present three worked-out examples which illustrate the applicability and usefulness of the methods explained in the paper.

Example 1. Consider the problem

$$\left.\begin{array}{ll}\min_x & -x_1^2+x_2^2-x_1x_2+3x_1-x_2 \\ & x_1+x_2=1.\end{array}\right\}$$

The reformulation is

$$\left.\begin{array}{ll}\min_{x,w} & -w_1^1+w_2^2-w_2^1+3x_1-x_2 \\ & x_1+x_2=1 \\ & w_1^1=x_1^2 \\ & w_2^1=x_1x_2 \\ & w_2^2=x_2^2.\end{array}\right\}$$

By multiplying the linear constraint $x_1+x_2=1$ by x_1 and then by x_2 we get the reduction constraint systems (having only one equation each) $w_1^1+w_2^1-x_1=0$ and $w_2^1+w_2^2-x_2=0$. From these, by substituting $x_1=x_1^2+x_1x_2$ and $x_2=x_1x_2+x_2^2$, we get $(w_1^1-x_1^2)+(w_2^1-x_1x_2)=0$ and $(w_2^1-x_1x_2)+(w_2^2-x_2^2)=0$. The corresponding "companion" system in the z variables (cf. eqn. (10), where $z_1^1=w_1^1-x_1^2$, $z_2^1=w_2^1-x_1x_2$ and $z_2^2=w_2^2-x_2^2$) is

$$\left.\begin{array}{rcl} z_1^1+z_2^1 &=& 0 \\ z_2^1+z_2^2 &=& 0.\end{array}\right\} \tag{24}$$

First of all notice that system (24) corresponds to $Bz=0$ where

$$B=\begin{pmatrix}1&1&0\\0&1&1\end{pmatrix}$$

and $z=(z_1^1,z_2^1,z_2^2)$, which clearly has rank 2, and nonbasic variable z_2^2. We should therefore keep the relation $w_2^2=x_2^2$ and discard the rest. For the sake of a complete explanation, however, let us follow the algorithm of section 4.2 as well. We first tackle the system $z_1^1+z_2^1=0$. This has rank 1 and is already in upper-triangular form. We impose the nonbasic variable $z_2^1=0$ and keep the resulting relation $w_2^1=x_1x_2$ in the formulation of the problem. Now we consider the second system, namely $z_2^1+z_2^2=0$. We remove the first column: the system is now reduced to $z_2^2=0$, which is a square system having rank 1; thus the algorithm terminates. Hence all we need to keep, in the formulation of the problem, is the relation $w_2^1=x_1x_2$. Notice that this is a different answer to that obtained by looking at system (24). This is not a mistake, however. It is easy to see, from system (24), that we may impose either $z_2^1=0$ or $z_2^2=0$ (or, for that matter, also $z_1^1=0$) to determine a square system of rank 2 with the unique zero solution for the remaining z variables. Notice also that in this case the algorithm in section 4.2 gives a set of bilinear constraints to keep that has minimal size.

The problem can now be expressed in form (14) as

$$\left.\begin{array}{ll}\min_{x,w} & -w_1^1 + w_2^2 - w_2^1 + 3x_1 - x_2 \\ & x_1 + x_2 = 1 \\ & w_1^1 + w_2^1 - x_1 = 0 \\ & w_2^1 + w_2^2 - x_2 = 0 \\ & w_2^1 = x_1 x_2. \end{array}\right\}$$

This is a big step forward, because a problem with three bilinear terms has been reduced to a problem with only one bilinear term. We can now replace the bilinear w-defining constraint $w_2^1 = x_1x_2$ with its McCormick convex relaxation to obtain a convex relaxation of the whole problem.

Example 2. Consider the problem

$$\left.\begin{array}{ll}\min_{x} & x_1^2 + x_2^2 + x_3^2 - x_1x_2 - x_2x_3 - x_3 \\ & x_1 + x_2 + 2x_3 = 1 \\ & x_1 + 2x_2 + x_3 = 1. \end{array}\right\}$$

After the reformulation, this becomes

$$\left.\begin{array}{ll}\min_{x,w} & w_1^1 + w_2^2 + w_3^3 - w_2^1 - w_3^2 - x_3 \\ & x_1 + x_2 + 2x_3 = 1 \\ & x_1 + 2x_2 + x_3 = 1 \\ & w_1^1 = x^2 \\ & w_2^1 = x_1x_2 \\ & w_2^2 = x^2 \\ & w_3^2 = x_2x_3 \\ & w_3^3 = x_3^2. \end{array}\right\}$$

Notice that the bilinear constraint $w_3^1 = x_1x_3$ is missing from the formulation, but we shall need it when considering the reduction constraint systems. The linear constraints of this problem correspond to the system $Ax = b$, where

$$A = \begin{pmatrix} 1 & 1 & 2 \\ 1 & 2 & 1 \end{pmatrix} \tag{25}$$

$x = (x_1, x_2, x_3)$, and $b = (1, 1)$. Multiplying this by x_1, x_2, x_3 we get the reduction constraint systems

$$\begin{array}{rcl} w_1^1 + w_2^1 + 2w_3^1 - x_1 & = & 0 \\ w_1^1 + 2w_2^1 + w_3^1 - x_1 & = & 0 \\ w_2^1 + w_2^2 + 2w_3^2 - x_2 & = & 0 \\ w_2^1 + 2w_2^2 + w_3^2 - x_2 & = & 0 \\ w_3^1 + w_3^2 + 2w_3^3 - x_3 & = & 0 \\ w_3^1 + 2w_3^2 + w_3^3 - x_3 & = & 0. \end{array}$$

If we let

$$\begin{array}{rcl} z_1^1 & = & w_1^1 - x_1^2 \\ z_2^1 & = & w_2^1 - x_1x_2 \\ z_3^1 & = & w_3^1 - x_1x_3 \\ z_2^2 & = & w_2^2 - x_2^2 \\ z_3^2 & = & w_3^2 - x_2x_3 \\ z_3^3 & = & w_3^3 - x_3^2, \end{array}$$

the reduction constraint systems above correspond to the companion system $Bz = 0$, where

$$B = \begin{pmatrix} 1 & 1 & 2 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 2 & 0 \\ 0 & 1 & 0 & 2 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 2 \\ 0 & 0 & 1 & 0 & 2 & 1 \end{pmatrix}$$

and $z = (z_1^1, z_2^1, z_3^1, z_2^2, z_3^2, z_3^3)$. This matrix has row echelon form (obtained without row or column permutations)

$$\begin{pmatrix} 1 & 1 & 2 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 2 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

and thus rank 5. This means that the reduction constraint systems can replace 5 out of 6 w-defining constraints. In particular, we should keep the one that corresponds to the nonbasic variable z_3^3, i.e. $w_3^3 = x_3^2$.

Let us now turn to the algorithm of section 4.2 applied to the matrix A in eq. (25). Writing A in row echelon form, we have

$$A = \begin{pmatrix} 1 & 1 & 2 \\ 0 & 1 & -1 \end{pmatrix}, \tag{26}$$

so $\mathrm{rk}(A) = 2$ and $\mathrm{nonbasic}(A, z^1) = \{z_3^1\}$. A^- (i.e. A without the first column) clearly has rank 2 and all variables are basic, so we can stop. So it is sufficient to keep the relation $w_3^1 = x_1 x_3$ in the formulation of the problem to recover all the other w-defining constraints through the reduction constraints. Again, the choice of the w-defining constraint to keep is not unique; and again the algorithm in section 4.2 gives a set of bilinear constraints to keep that has minimal size.

Example 3. This is a more complex problem, expressed in formulation (1), which will illustrate most of the techniques described in this paper. Notice that in this example we tackle a MINLP, as x_5 is a binary variable.

$$\left.\begin{array}{ll} \min_x & x_1^2 + x_1x_2 - x_1x_3 - 2x_1x_4 + x_2^2 + 3x_2x_4 - x_2x_5 + x_3x_4 + 3x_4^2 + 2x_4x_5 - \\ & \quad -x_1 - x_4 - x_6 + e^{-x_2x_3} \\ & x_1 + x_2 - x_3 + x_4 + x_5 = 1 \\ & x_2 - x_4 - x_5 = -1 \\ & x_1 + 2x_2 - 2x_3 \geq 0 \\ & 2x_1 + 7x_2 - x_3 \leq 0 \\ & e^{x_3x_4} - \log(x_6) - x_2x_6 \leq 1 \\ & \forall i \leq 4 \; x_i \in [0, 10] \\ & x_5 \in \{0, 1\} \\ & x_6 \in [1, 2] \end{array}\right\}$$

The reformulated problem is:

$$\left.\begin{array}{ll}\min_{x,w} & w_1^1 + w_2^1 - w_3^1 - 2w_4^1 + w_2^2 + 3w_4^2 - w_5^2 + w_4^3 + 3w_4^4 + 2w_5^4 - x_1 - x_4 - x_6 + e^{-w_3^2} \\
& x_1 + x_2 - x_3 + x_4 + x_5 = 1 \\
& x_2 - x_4 - x_5 = -1 \\
& x_1 + 2x_2 - 2x_3 \geq 0 \\
& 2x_1 + 7x_2 - x_3 \leq 0 \\
& e^{w_4^3} - \log(x_6) - w_6^2 \leq 1 \\
& x_5 - w_5^5 = 0 \\
& w_1^1 = x_1^2 \\
& w_2^1 = x_1 x_2 \\
& w_3^1 = x_1 x_3 \\
& w_4^1 = x_1 x_4 \\
& w_2^2 = x_2^2 \\
& w_3^2 = x_2 x_3 \\
& w_4^2 = x_2 x_4 \\
& w_5^2 = x_2 x_5 \\
& w_6^2 = x_2 x_6 \\
& w_4^3 = x_3 x_4 \\
& w_4^4 = x_4^2 \\
& w_5^4 = x_4 x_5 \\
& w_5^5 = x_5^2 \\
& \forall i \leq 4\ (x_i \in [0, 10]) \\
& x_5, w_5^5 \in [0, 1] \\
& x_6 \in [1, 2] \\
& w_5^2, w_5^4 \in [0, 10] \\
& w_6^2 \in [0, 20] \\
& \forall i \leq j \leq 4\ (w_j^i \in [0, 100])
\end{array}\right\}$$

We have derived the bounds on the w variables by means of interval arithmetic on the bilinear products in the x variables. Notice the formulation is missing bilinear terms

$$\begin{array}{rcl} w_5^1 & = & x_1 x_5 \\ w_3^3 & = & x_3^2 \\ w_5^3 & = & x_3 x_5, \end{array}$$

which we shall need to include in order to recover a full set of w variables. Notice also that only one of the possible bilinear terms involving x_6 (namely, w_6^2) is present: however, we shall not add all the missing w-defining constraints involving x_6 in the formulation: since the linear constraints do not depend on x_6, no reduction constraint could eliminate the bilinear terms in x_6. In order to obtain a convex relaxation for this problem, the bilinear constraint $w_6^2 = x_2 x_6$ will just have to be replaced by its McCormick envelope.

By multiplying the two linear equation constraints in the problem

$$\left.\begin{array}{rcl} x_1 + x_2 - x_3 + x_4 + x_5 & = & 1 \\ x_2 - x_4 - x_5 & = & -1 \end{array}\right\} \tag{27}$$

by each variable x_i for $i \leq 5$ we can derive a set of ten reduction constraints. The companion system is a 10×15 matrix with rank 9, which would mean that we only need to include six bilinear terms out of a possible fifteen.

Now notice that summing the two linear equations (27) together we get the equation $x_1 + 2x_2 - x_3 = 0$, which describes a linear 5-dimensional hypersurface T in $\mathbb{R}^6$. Notice also that the two linear inequality constraints only involve variables x_1, x_2, x_3: thus we can consider T as a linear 2D surface (i.e., a plane) in relation to the planes in $\mathbb{R}^3$ described by the equations derived by the inequality system

$$\begin{array}{rcl} x_1 + 2x_2 - 2x_3 & = & 0 \\ 2x_1 + 7x_2 - x_3 & = & 0 \end{array}$$

In this case it is not difficult to see that the plane T, defined by $x_1 + 2x_2 - x_3 = 0$, has the same distance from the two planes above, for T is a bisecting plane for the angle between the other two planes. To see this, notice that the projections of the planes onto the x_1, x_3 axes are given by the lines $x_3 = \frac{1}{2}x_1$ and $x_3 = 2x_1$, and the projection of T onto the same axes is $x_3 = x_1$; furthermore, the projections of the planes onto the x_2, x_3 axes are given by the lines $x_3 = x_2$ and $x_3 = 7x_2$, and the projection of T onto the same axes is $x_3 = 2x_2$. In both cases, the projections of T bisect the angle between the projections of the other two planes (this is an easy argument in elementary geometry).

Thus, we can reformulate the two linear inequality constraints as two linear constraints involving x_1, x_2, x_3 and a slack variable $s \geq 0$. Because of the observations made in section 7.4, we can use s as a slack variable for both constraints. Thus the inequality constraints become:

$$\left.\begin{array}{rcl} x_1 + 2x_2 - 2x_3 - s & = & 0 \\ 2x_1 + 7x_2 - x_3 + s & = & 0 \end{array}\right\} \tag{28}$$

In order to generate some more reduction constraints from the above equations, we need to create the following w-defining bilinear constraints:

$$\begin{array}{rcl} w_s^1 & = & x_1 s \\ w_s^2 & = & x_2 s \\ w_s^3 & = & x_3 s \\ w_s^4 & = & x_4 s \\ w_s^5 & = & x_5 s \\ w_s^s & = & s^2; \end{array}$$

We get additional reduction constraints by multiplying s by the two linear equations in (27) and by multiplying the eqns. in (28) by $x_1, \ldots, x_5, s$. Thus, for each $i \leq 5$ we get a reduction constraint system $Aw^i - x_i b = 0$ plus the added system $Aw^s - sb = 0$, where

$$A = \begin{pmatrix} 1 & 1 & -1 & 1 & 1 & \\ & 1 & & -1 & -1 & \\ 1 & 2 & -2 & & & -1 \\ 2 & 7 & -1 & & & 1 \end{pmatrix},$$

$w^i = (w_1^i, \ldots, w_5^i, w_s^i)$, $w^s = (w_1^s, \ldots, w_5^s, w_s^s)$ and $b = (1, -1, 0, 0)$. The corresponding companion system $Bz = 0$ is such that $z = (z_1^1, \ldots, z_5^1, z_s^1, z_2^2, \ldots, z_5^2, z_s^2, \ldots, z_5^5, z_s^5, z_s^s)$, B is a 24×21 matrix with rank 18 and nonbasic variables z_5^5, z_s^5, z_s^s. This means that we only need to keep the three corresponding w-defining constraints in the formulation: adding the reduction constraints will recover the rest of the bilinear terms. Unfortunately, the constraint corresponding to z_5^5 is crucial for the definition of x_5 as a binary variable: leaving it in the formulation implies that forming the convex relaxation of the whole problem would replace that term with its corresponding convex relaxation (15), and lose the definition of integrality of x_5 (see section 8). To avoid this situation we need to choose another nonbasic variable set from the companion system. We apply a permutation to the variable set $\{x_1, x_2, x_3, x_4, x_5\}$, like for example the cycle (2 5) (which interchanges x_2 and x_5), which will cause the the vector z to become

$$\begin{array}{rcl} z & = & (z_1^1, z_5^1, z_3^1, z_4^1, z_2^1, z_s^1, \\ & & z_5^5, z_3^5, z_4^5, z_2^5, z_s^5, \\ & & z_3^3, z_4^3, z_2^3, z_s^3, \\ & & z_4^4, z_2^4, z_s^4, \\ & & z_2^2, z_s^2, \\ & & z_s^s). \end{array}$$

The companion system then has rank 18 and nonbasic variables z_4^4, z_s^4, z_s^s. Thus we only need to keep the bilinear relations stemming from these three variables in order to recover the rest.

For the sake of completeness, we carry out the algorithm of section (4.2). Applying Gaussian elimination to A, we get a 6×4 matrix having rank 4:

$$\begin{pmatrix} 1 & 1 & -1 & 1 & 1 & \\ & 1 & & -1 & -1 & \\ & & -1 & & & -1 \\ & & & 3 & 3 & \end{pmatrix}.$$

Following the algorithm, this has nonbasic variables z_5^1, z_s^1. We eliminate the first column and find that the rank of the matrix thus obtained is still 4. We re-apply Gaussian elimination, and permute columns 4 and 5 (corresponding to variables z_5^2 and z_s^2) to find the matrix

$$\begin{pmatrix} 1 & -1 & 1 & & 1 \\ & 1 & -2 & & 2 \\ & & -2 & -1 & -2 \\ & & & -\frac{3}{2} & \end{pmatrix},$$

which has rank 4 and nonbasic variable z_5^2 (the nonbasic variable corresponds to the last column, but keep in mind that z_5^2 and z_s^2 have been permuted, so the last column is actually z_5^2). If we remove the first column we find that the rank decreases to 3, so we remove the first row as well and end up with

$$\begin{pmatrix} 1 & -2 & & 2 \\ & -2 & -1 & -2 \\ & & -\frac{3}{2} & \end{pmatrix},$$

which has rank 3 and nonbasic variable z_5^3. Again, removing the first column we find that the rank decreases to 2, so we remove the first row to get

$$\begin{pmatrix} -2 & -1 & -2 \\ & -\frac{3}{2} & \end{pmatrix},$$

which has rank 2 and nonbasic variable z_5^4. Removing the first column now produces a matrix with rank 2, so the algorithm terminates. Thus, the algorithm identifies the set $\{z_5^1, z_s^1, z_5^2, z_5^3, z_5^4\}$ of z variables whose corresponding bilinear constraints we should keep in the formulation. Notice that the size of this set is not minimal (as the companion matrix produces sets of size 3), but it does not include the bilinear term z_5^5 (crucial for the integrality of x_5) and it is still smaller than the set of bilinear terms that we should have kept had we not considered deriving reduction constraints from inequalities.

So, finally, at the end of the reduction constraint creation process we obtain the following reformulated problem:

$$
\left.
\begin{array}{ll}
\min_{x,w} & w_1^1 + w_2^1 - w_3^1 - 2w_4^1 + w_2^2 + 3w_4^2 - w_5^2 + w_4^3 + 3w_4^4 + 2w_5^4 - x_1 - x_4 - x_6 + e^{-w_3^2} \\
 & x_1 + x_2 - x_3 + x_4 + x_5 = 1 \\
 & x_2 - x_4 - x_5 = -1 \\
 & x_1 + 2x_2 - 2x_3 - s = 0 \\
 & 2x_1 + 7x_2 - x_3 + s = 0 \\
 & e^{w_4^3} - \log(x_6) - w_6^2 \leq 1 \\
 & x_5 - w_5^5 = 0 \\
 & w_1^1 + w_2^1 - w_3^1 + w_4^1 + w_5^1 - x_1 = 0 \\
 & w_2^1 - w_4^1 - w_5^1 + x_1 = 0 \\
 & w_2^1 + w_2^2 - w_3^2 + w_4^2 + w_5^2 - x_2 = 0 \\
 & w_2^2 - w_4^2 - w_5^2 + x_2 = 0 \\
 & w_3^1 + w_3^2 - w_3^3 + w_4^3 + w_5^3 - x_3 = 0 \\
 & w_3^2 - w_4^3 - w_5^3 + x_3 = 0 \\
 & w_4^1 + w_4^2 - w_4^3 + w_4^4 + w_5^4 - x_4 = 0 \\
 & w_4^2 - w_4^4 - w_5^4 + x_4 = 0 \\
 & w_5^1 + w_5^2 - w_5^3 + w_5^4 + w_5^5 - x_5 = 0 \\
 & w_5^2 - w_5^4 - w_5^5 + x_5 = 0 \\
 & w_s^1 + w_s^2 - w_s^3 + w_s^4 + w_s^5 - x_s = 0 \\
 & w_s^2 - w_s^4 - w_s^5 + x_s = 0 \\
 & w_1^1 + 2w_2^1 - 2w_3^1 - w_s^1 = 0 \\
 & 2w_1^1 + 7w_2^1 - w_3^1 + w_s^1 = 0 \\
 & w_2^1 + 2w_2^2 - 2w_3^2 - w_s^2 = 0 \\
 & 2w_2^1 + 7w_2^2 - w_3^2 + w_s^2 = 0 \\
 & w_3^1 + 2w_3^2 - 2w_3^3 - w_s^3 = 0 \\
 & 2w_3^1 + 7w_3^2 - w_3^3 + w_s^3 = 0 \\
 & w_4^1 + 2w_4^2 - 2w_4^3 - w_s^4 = 0 \\
 & 2w_4^1 + 7w_4^2 - w_4^3 + w_s^4 = 0 \\
 & w_5^1 + 2w_5^2 - 2w_5^3 - w_s^5 = 0 \\
 & 2w_5^1 + 7w_5^2 - w_5^3 + w_s^5 = 0 \\
 & w_s^1 + 2w_s^2 - 2w_s^3 - w_s^s = 0 \\
 & 2w_s^1 + 7w_s^2 - w_s^3 + w_s^s = 0 \\
 & w_6^2 = x_2 x_6 \\
 & w_4^4 = x_4^2 \\
 & w_s^4 = x_4 s \\
 & w_s^s = s^2 \\
 & \forall i \leq 4 \; (x_i \in [0,10]) \\
 & x_5, w_5^5 \in [0,1] \\
 & x_6 \in [1,2] \\
 & w_6^2 \in [0,20] \\
 & \forall i \leq j \leq 4 \; (w_j^i \in [0,100]) \\
 & s \geq 0 \\
 & \forall i \leq 4 \; (w_5^i \in [0,10]) \\
 & \forall i \leq 5 \; (w_s^i \geq 0) \\
 & w_s^s \geq 0
\end{array}
\right\}
$$

which only contains three of the possible 21 bilinear terms in the $x_1, \ldots, x_5, x_s$ variables, plus the bilinear term $x_2 x_6$.

10 Numerical Results

The methods and algorithms described above were implemented in a software framework for global optimization called $oo\mathcal{OPS}$, i.e. object-oriented Optimization System [LTKP01]. Within this framework, using an implementation of the spatial Branch-and-Bound algorithm [SP99] as the global solution module, and SNOPT [Gil99] as the local solution module, all the ideas proposed in this paper were put to test, by solving the three examples of section (9). The tests have been carried out on a Pentium III class machine running at 850MHz with 384MB RAM.

Table 1 shows the results of these tests. By "reformulated problem" we mean the full reformulation, taking into account reduction constraints and the removal of unnecessary bilinear terms. "Iterations" is the number of main iterations taken by the Branch-and-Bound procedure to arrive at the global solution (the same as the number of explored regions). The CPU time is just the user time (as opposed to the total CPU time, i.e. the sum of user and system time). $|V|$ is the number of problem variables and $|B|$ is the number of bilinear terms in the problem.

	Original Problem				Reformulated problem			
	Iterations	CPU Time	$\lvert V\rvert$	$\lvert B\rvert$	Iterations	CPU Time	$\lvert V\rvert$	$\lvert B\rvert$
Example 1	63	0.1 s	2	3	1	0.02 s	5	1
Example 2	15	0.13 s	3	5	7	0.14 s	9	1
Example 3	35	15.13 s	6	12	1	3.56 s	28	4

Table 1: Numerical results.

In all cases the main results were confirmed:

- the reformulation is correct: the global solution arrived at by solving the original problem was the same as the solution reached by solving the reformulated problem;
- the reformulation is advantageous: the Branch-and-Bound procedure took less iterations to arrive at the global solution.

Notice that in example 3 — a MINLP with a binary variable — we found the global solution at the first iteration. This means that the solution of the convex relaxation was feasible in the original problem: and this, in turn, means that the integrality of the binary variable had not been lost in the convex relaxation, as explained in section 8. Furthermore, example 3 is the only one in the test set to include some of the novel techniques mentioned in section 7 for the generation of reduction constraints from inequality systems. It is also worth pointing out that in examples 1 and 3 there were substantial CPU time savings (of about 80% of the total time taken to solve the original problem). This result is all the more significant as example 3 is the largest and most complicated of the test set.

This investigation seems to point out that methods which act on the formulation of the problem are extremely effective in cutting computational costs when compared to heuristic decisions in the Branch-and-Bound algorithm (like e.g. the choice of branching rule). One of the reasons for this is surely that reformulation methods are usually a pre-processing step, and thus only add a fixed amount of computational time to the algorithm (which is not proportional to the number of iterations). Furthermore, when a reformulation is based on theoretical concepts rather than heuristic ideas, it is easier to understand its implications and to be able to forecast what kind of computational time saving it might involve.

11 Applicability to Sparse Problems

As noted in section 1, although the theory always holds true, the methods derived in this paper rest on the assumption that the matrix Q, in problem formulation (1), is dense. In this section we shall endeavour to explain why this is the case and how the problems arising from this assumption can be tackled.

The technique we proposed to replace bilinear terms with reduction constraints involves muliplying all existing linear constraints by all the problem variables. The number of bilinear terms deleted is equal to the rank t of the companion system (10). The companion system has mn rows and $\frac{1}{2}n(n+1)$ columns, where we have assumed $m \leq n$. On the basis of the conjecture of section 3, it is extremely likely that the companion system has a rank t which is strictly less than the number of its columns. We have shown in theorem 3.5 that in that case we need to keep precisely $r = \frac{1}{2}n(n+1) - t$ bilinear terms in the formulation of the problem, i.e. those bilinear terms that correspond to a set of nonbasic variables of the companion system.

The problem we face, when Q is a sparse matrix, is that the number of bilinear terms in the original problem formulation (1) might be less than r. And even if it is not, it might happen that each set of nonbasic variables of the companion system involves bilinear terms which are not present in the original problem formulation. In short, when Q is sparse it is very likely that after the reformulation we shall end up with a bigger set of bilinear terms in the problem.

Haverly's pooling problem, as formulated in [ATS99], provides an example of this occurrence:

$$\left.\begin{array}{rl}
\min & 6x_1 + 16x_2 + 10x_3 - 9x_4 - 15x_5 - 9x_6 - 15x_7 \\
\text{s.t.} & x_1 + x_2 - x_4 - x_5 = 0 \\
& x_3 - x_6 - x_7 = 0 \\
& x_4 + x_5 - x_9 = 0 \\
& x_4 + x_6 \leq 100 \\
& x_5 + x_7 \leq 200 \\
& -3x_1 - x_2 + x_8 x_9 = 0 \\
& -\frac{5}{2}(x_4 + x_6) + 2x_6 + x_8 x_4 \leq 0 \\
& -\frac{3}{2}(x_5 + x_7) + 2x_7 + x_8 x_5 \leq 0 \\
& 0 \leq x_1, x_9 \leq 300 \\
& 0 \leq x_2, x_3, x_4, x_6 \leq 100 \\
& 0 \leq x_5, x_7 \leq 200 \\
& 0 \leq x_8 \leq 10
\end{array}\right\} \tag{29}$$

In this formulation, Haverly's pooling problem has 3 linear equality constraints, 9 problem variables and 3 bilinear terms. The resulting companion system (10) has 27 rows, 45 columns and rank 24. Therefore $r = 45 - 24 = 21$: at the end of the reformulation process we go from three bilinear terms to 21. We might conclude that this problem is not susceptible of reformulation via reduction constraints: but this is false. In fact, it is easy to notice that multiplying constraint $x_4 + x_5 - x_9 = 0$ by x_8 produces a reduction constraint $w_8^4 + w_8^5 + w_9^8 = 0$ (where $w_j^i = x_i x_j$ for all $i, j \leq n$) which could allow us to eliminate one of the three bilinear terms.

This should not be seen as a counter-example to the theory and methods derived in this paper. It simply shows that, if the problem is sparse, we need to identify subsets of the original problem where we can apply this theory profitably. By "subset of a problem" here we mean a subset of problem variables and constraints where we can successfully apply the ideas of section 3. Supposing we were able *a priori* to restrict our attention to the constraint subset $\{x_4 + x_5 - x_9 = 0\}$ with the multiplier variable x_8, the conclusions drawn by the theory would still hold.

11.1 Eliminating Zero Column Variables

By using the "trick" described in this section it is sometimes possible to discard some bilinear terms corresponding to nonbasic variables in the companion system. This is useful to reduce the number of bilinear terms required to apply the reduction constraints theory, and it is particularly significant when sparse problems are considered. Let x_k (for $k \le n$) be a problem variable whose corresponding column in the matrix A is zero. Assume

$$A = \begin{pmatrix} a_{11} & \dots & a_{1,k-1} & 0 & a_{1,k+1} & \dots & a_{1n} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ a_{n1} & \dots & a_{n,k-1} & 0 & a_{n,k+1} & \dots & a_{nn} \end{pmatrix}.$$

How does this reflect on the matrix B (see p. 7) of the companion system (10)? Because of lemma 4.1, if we let $\nu(k) = \frac{1}{2}k(2n-k+1)$, the $\nu(k)$-th column of B is a zero column. In other words the rest of the z variables are independent of z_k^k. From this we can conclude that the w-defining constraint $w_k^k = x_k^2$ is not necessary to enforce the validity of the other w-defining constraints. Thus, we only need to keep x_k^2 if it is present in the original problem fomulation. If it is not, we shall not need to include it, even though it corresponds to a nonbasic variable of the companion system.

Suppose now that matrix A has more than one zero column: say columns $k_1, \dots, k_u$ where $u < n$ are all zero columns. It follows that for all $j \le u$ bilinear terms $x_{k_j}^2$ are not needed in the formulation to infer the validity of any of the other w-defining constraints. But it is also easy to show that in this case there are other columns, in B, that only have zero entries. By the characterization of matrix B in section 4 we can see that columns of B corresponding to variables $w_{k_j}^{k_i}$ for all $i < j \le u$ are also zero. Thus, we shall only need to keep the bilinear terms $x_{k_i}x_{k_j}$ if they are present in the original formulation, but we shall not need to include them otherwise, even though they correspond to nonbasic variables of the companion system.

11.2 Modelling an Optimal Procedure for Reduction Constraint Creation

In this section we will rigorously formulate the problem of building the largest possible set of reduction constraints whilst keeping the number of needed bilinear terms at a minimum. To this end, we want to identify all maximal subsets of linear constraints and problem variables so that forming reduction constraints from them will minimize the number of bilinear constraints needed to apply the theory of reduction constraints. We require these subsets to be maximal in the sense that adding elements to them will cease to make the application of the theory successful. In what follows, we shall refer to the rank of a set of linear constraints: by that we mean the rank of the linear system of equations composed by the linear constraints in the set.

For all $i \le m$, let $c_i = \sum_{i=1}^n a_{ij}x_j - b_i$. Then for each $i \le m$ we have that $c_i = 0$ is a constraint of the system $Ax = b$. For each $k \in \mathbb{N}$ let C_k, V_k be index subsets such that $C_k \subseteq \{1, \dots, m\}$ and $V_k \subseteq \{1, \dots, n\}$. Let $F = \{(C_k, V_k) \mid k \in \mathbb{N}\}$ and $\mathcal{F}$ be the family of all such sets F. For each $F \in \mathcal{F}$, we define T_k to be the set of reduction constraints obtained by multiplying each constraint c_i (for $i \in C_k$) by each variable x_j (for $j \in V_k$) and substituting each bilinear term $x_j x_l$ appearing in the products $x_j c_i$ with its corresponding w_l^j variable; we define Z_k to be the set of constraints (in the z variables) in each of the reduction companion systems (see defn. 3.2) of T_k; and we define U_k to be the set of bilinear terms appearing in the w-defining constraints (see defn. 2.1) needed to create the reduction constraints in T_k. Let $Z = \bigcup_{k\in\mathbb{N}} Z_k$, $T = \bigcup_{k\in\mathbb{N}} T_k$, $U = \bigcup_{k\in\mathbb{N}} U_k$, and let t be the rank of system Z. Let U_0 be the set of bilinear terms already present in the original problem (1). Finally let $\tau(F) = |U \cup U_0| - t$ be the number of bilinear terms needed in the problem after the reduction constraint creation process. We look for a set $F \in \mathcal{F}$ that satisfies the following discrete optimization problem:

$$\left.\begin{array}{ll} \min_{F\in\mathcal{F}} & \tau(F) \\ & \tau(F) < |U_0| \end{array}\right\} \tag{30}$$

that is, we require that the number of bilinear terms present in the problem after the reduction constraint creation process be minimal, and in particular strictly less than the number of bilinear terms present in the original problem. Notice that in the discussion above, although it was not made explicit, the set U_0 does not depend on the choice of F, whereas the sets T_k, Z_k, U_k, T, Z, U do, as does t. Let F be a solution of (30) and $K = |F|$. Notice that since the Z_k, T_k and U_k are in general not disjoint systems of sets, we have $|R| \leq \sum_{k=1}^{K} |T_k|$, $|Z| \leq \sum_{k=1}^{K} |Z_k|$, $|U \cup U_p| \leq |U_0| + \sum_{k=1}^{K} |U_k|$ and $t \leq \sum_{k=1}^{K} \mathrm{rk}(Z_k)$. This emphasizes the interdependencies of the maximal subset pairs (C_k, V_k) for different k's and further complicates an already difficult problem.

Any procedure for finding a global solution F of problem (30) will be an optimal creation process for reduction constraints with respect to any given original problem (1): by keeping $\tau(F)$ globally minimal, we make sure that there is no better choice of the sets C_k, V_k; and by requiring that $\tau(F) < |U_p|$ we keep the number of needed bilinear terms strictly less than the number of bilinear terms in the original problem.

A direct procedure for the global minimization of $\tau(F)$ over all the possible choices of $F \in \mathcal{F}$ would involve the use of a discrete Branch-and-Bound algorithm for the intelligent enumeration of all such families F. The solution of such a problem may or may not be an acceptable idea, depending on the size of the problem and the amount of time given for pre-processing the original problem (1). The discussion of such an algorithm is outside of the scope of this paper, but it should not be too hard to implement, given the amount of literature on discrete Branch-and-Bound algorithms [Wol98, AHU83, KV00].

A possible simplification of problem (30), which would still gather a useful result, is to find a feasible solution F rather than an optimal one.

11.3 Algorithm for Reduction Constraint Creation in Sparse Problems

The algorithm presented in this section identifies a subset of reduction constraints the creation of which is convenient in terms of added bilinear terms, although the result is, in general, not optimal in the sense of problem (30). This algorithm is based on an analysis of the companion matrix B (see section 4). Recall that the companion system is $Bz = 0$, where $z = (z_1^1, \ldots, z_n^1, z_2^2, \ldots, z_n^2, \ldots, z_n^n)$ is such that each column of B corresponds to a bilinear term in the problem variables x. In particular, column z_j^i corresponds to the bilinear term $x_i x_j$ for all $i \leq j \leq n$. In this algorithm we shall use some of the terminology of section 11.2.

1. Construct the companion system $Bz = 0$ in the usual way as explained in section 4, and let $R(w; x) = 0$ be the corresponding reduction constraint system.
2. Reduce B to row echelon form with nonzero entries along the diagonal, and replicate the same row operations on system $R(w; x) = 0$. Delete from B any row with no nonzero entries, and delete the corresponding rows from the system $R(w; x) = 0$. Let $\rho(i)$ be the reduction constraint in $R(w; x) = 0$ that corresponds to the i-th equation of the companion system $Bz = 0$.
3. Mark the columns of B corresponding to the z variables defined by the original problem bilinear terms in U_0.
4. Find the subsystem $B'z' = 0$ consisting of all the rows of B which have a nonzero entry in the marked columns; discard from B' all the columns consisting only of zeroes (and adjust the variable vector z' accordingly, so that system $B'z' = 0$ now has no nonzero columns). Let m' be the number of rows and n' be the number of columns of of B'. For all $i \leq m'$ let $\delta(i)$ be the index in B of the i-th row of B'.
5. For each $i \leq m'$, let $\omega(i)$ be the set of column indices corresponding to nonzero entries in the i-th row; let $\zeta(i)$ be the number of columns in $\omega(i)$ which have been marked in step 3. Associate a cost

$\phi(i)$ with the i-th row in B': let $\phi(i)$ be initially the number of nonzero entries of row i, and then set $\phi(i) \leftarrow \phi(i) - \zeta(i)$.

6. For each $j \leq n'$ let $\psi(j)$ be the set I of row indices such that for all $i \in I$ the i-th row has a nonzero entry in the j-th column of B'.
7. Order the columns of B' so that column j_1 is less than column j_2 if $|\psi(j_1)| > |\psi(j_2)|$ (columns such that $|\psi(j_1)| = |\psi(j_2)|$ can retain their natural order). To this ordering there corresponds a permutation π of the columns acting on the set $\{1, \ldots, n'\}$.
8. Initialize $k = 1$ and $\Gamma = \{1, \ldots, m'\}$.
9. Find j such that $\pi(j) = k$. For all $i \in \psi(j) \cap \Gamma$ set $\phi(i) \leftarrow \phi(i) - 1$, and $\Gamma \leftarrow \Gamma \setminus \psi(j)$ (i.e., discard from Γ all indices in $\psi(j)$).
10. If $k = n'$ or if $\Gamma = \emptyset$ then go to step 12.
11. Set $k \leftarrow k + 1$ and go back to step 9.
12. Discard from B' all the rows whose associated cost ϕ is strictly positive. Update the map δ which relates the indices of B' to the original indices in B, and the number of rows m' of B'.
13. Discard from B' all the columns with no nonzero entries, and update z' accordingly, so that $B'z' = 0$ is a system with no nonzero columns.
14. Let $R'(w; x)$ be the set of equations of system $R(w; x) = 0$ indexed by the set $\{\delta(i) \mid i \leq m'\}$. Discard from R' all columns, among the first $\frac{1}{2}n(n+1)$, containing no nonzero entries, to obtain the reduced reduction constraint system $R'(w'; x) = 0$.

The system $R'(w'; x) = 0$ is a set of reduction constraints such that the w'-defining constraints needed to define it, minus the rank of B', is less than $|U_0|$. The crux of this algorithm is the loop in steps 8-11, where the cost of reduction constraints that have common bilinear terms is decreased. The algorithm works because in step 12 we discard all reduction constraints with a positive cost. The remaining reduction constraint system is such that any added bilinear term is counterbalanced by a reduction constraint, and no other bilinear term is added to the problem.

This algorithm has been tested on Haverly's pooling problem in formulation (29) and has correctly identified the reduction constraint stemming from multiplying constraint $x_4 + x_5 - x_9 = 0$ by variable x_8.

Another attempt to construct an efficient (in terms in the number of reduction constraints versus the number of needed bilinear terms) and fast algorithm for the creation of reduction constraints, based on graph theory, is currently under way [LP02].

12 Conclusion

In this paper we have shown how certain nonconvex problems can be easily reformulated to exhibit more linearity than what is apparent at a first glance; more precisely, some of the bilinear terms present in the original problem can be substituted by special linear constraints called *reduction constraints*. We have provided a theory describing the properties of these reduction constraints and their realm of applicability, and derived algorithms to make their construction process automatic. By combining reduction constraints with existing convex relaxation methods for bilinear terms we derived a convex relaxation for the original bilinear problem. We showed that this convex relaxation is equivalent to that obtained by the RLT (Reformulation-Linearization Technique), thus demonstrating that the concepts on which the RLT is based can give rise to a method which is not heuristic but precise (because it explains why and when a bilinear term can be replaced by a reduction constraint and it shows exactly which reduction constraints need to be created). We then explored some of the ideas that can be used in deriving reduction constraints from systems of inequalities; and we explained how reduction constraints can be beneficial in the solution of mixed-integer nonlinear programming problems (MINLPs). We gave three worked-out examples of the usefulness of reduction constraints, and some numerical results of tests which confirm the conclusions drawn by the theory. Finally, we explained how to apply the methods derived in this paper to sparse bilinear problems.

12.1 Future Work

As a concluding remark, we would like to point out directions for future work.

1. In most cases, applying a permutation to the original set of variables $\{x_1, \ldots, x_n\}$ gives rise to a different set of resulting nonbasic variables in the companion system, and in turn to a different choice of the set of bilinear constraints to keep in the formulation. It would be a significant step forward in this research to have a theory which, given a permutation, could tell us directly what the resulting set of nonbasic variables of the companion system is. This would involve tracking the action of the permutation through the construction of the companion system and also through the application of Gaussian elimination: it does not seem like a straightforward piece of work. Nonetheless, with such a theory it would be possible to choose *a priori* which bilinear constraints to keep and which to discard: in view of the application to MINLPs (section 8), this is very important.
2. Stratos' idea (applicable to problems in form (2): choose a set of nonbasic variables of the companion system such that the substituted bilinear terms are only those which are nonconvex (might be left with a totally convex objective function, which can then be solved to global optimality by a local optimization software).
3. All the paragraphs in section 7 need deeper analysis and understanding; particularly the method in subsection 7.1, which is still at "heuristic" stage.
4. The algorithm modelled in section 11.2 needs an implementation: and there is no proof that an optimal reduction constraint creation process actually has to have exponential complexity.
5. Relations involving higher degree monomials in $\mathbb{R}[x]$ can be obtained from system $Ax = b$ by multiplying it by polynomials $M(x) \in \mathbb{R}[x]$. How can this fact be used in extending reduction constraints to polynomial problems?

References

[ADFN98] 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.

[Adj98] C.S. Adjiman. *Global Optimization Techniques for Process Systems Engineering.* PhD thesis, Princeton University, June 1998.

[AHU83] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *Data Structures and Algorithms.* Addison-Wesley, Reading, MA, 1983.

[AKF83] F.A. Al-Khayyal and J.E. Falk. Jointly constrained biconvex programming. *Mathematics of Operations Research*, 8(2):273–286, 1983.

[ATS99] N. Adhya, M. Tawarmalani, and N.V. Sahinidis. A lagrangian approach to the pooling problem. *Industrial and Engineering Chemistry Research*, 38:1956–1972, 1999.

[EP97] T.G.W. Epperly and E.N. Pistikopoulos. A reduced space branch and bound algorithm for global optimization. *Journal of Global Optimization*, 11:287:311, 1997.

[Epp95] T.G.W. Epperly. *Global Optimization of Nonconvex Nonlinear Programs using Parallel Branch and Bound.* PhD thesis, University of Winsconsin – Madison, 1995.

[Gil99] P.E. Gill. *User's Guide for SNOPT 5.3.* Systems Optimization Laboratory, Department of EESOR, Stanford University, California, feb 1999.

[KB00] P. Kesavan and P.I. Barton. Decomposition algorithms for nonconvex mixed-integer nonlinear programs. *AIChE Symposium Series*, 96(323):458–461, 2000.

[KV00] B. Korte and J. Vygen. *Combinatorial Optimization, Theory and Algorithms.* Springer-Verlag, Berlin, 2000.

[LP02] L. Liberti and C.C. Pantelides. Reformulation algorithm for nonconvex nlps. *Working paper, CPSE, Imperial College, London UK*, Sept. 2002.

[LTKP01] L. Liberti, P. Tsiakis, B. Keeping, and C.C. Pantelides. *oo$\mathcal{OPS}$.* Centre for Process Systems Engineering, Chemical Engineering Department, Imperial College, London, UK, 1.24 edition, jan 2001.

[McC76] G.P. McCormick. Computability of global solutions to factorable nonconvex programs: Part i — convex underestimating problems. *Mathematical Programming*, 10:146–175, 1976.

[PR02] P.M. Pardalos and H. Edwin Romeijn, editors. *Handbook of Global Optimization*, volume 2. Kluwer Academic Publishers, Dordrecht, 2002.

[RS95] H. S. Ryoo and N. V. Sahinidis. Global optimization of nonconvex nlps and minlps with applications in process design. *Computers & Chemical Engineering*, 19(5):551–566, May 1995.

[SA92] H.D. Sherali and A. Alameddine. A new reformulation-linearization technique for bilinear programming problems. *Journal of Global Optimization*, 2:379–410, 1992.

[She02] H.D. Sherali. Tight relaxations for nonconvex optimization problems using the reformulation-linearization/convexification technique (rlt). In Pardalos and Romeijn [PR02], pages 1–63.

[Smi96] E.M.B. Smith. *On the Optimal Design of Continuous Processes.* PhD thesis, Imperial College of Science, Technology and Medicine, University of London, October 1996.

[SP97] E.M.B. Smith and C.C. Pantelides. Global optimisation of nonconvex minlps. *Computers and Chemical Engineering*, 21:S791–S796, 1997.

[SP99] E.M.B. Smith and C.C. Pantelides. A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex minlps. *Computers and Chemical Engineering*, 23:457–478, 1999.

[SSA00] H.D. Sherali, J.C. Smith, and W.P. Adams. Reduced first-level representations via the reformulation-linearization technique: Results, counterexamples, and computations. *Discrete Applied Mathematics*, 101:247–267, 2000.

[VEH96] R. Vaidyanathan and M. El-Halwagi. Global optimization of nonconvex minlps by interval analysis. In I.E. Grossmann, editor, *Global Optimization in Engineering Design*, pages 175–193, Dordrecht, 1996. Kluwer Academic Publishers.

[Wol98] L.A. Wolsey. *Integer Programming.* Wiley, New York, 1998.