An exact reformulation algorithm for large nonconvex NLPs involving bilinear terms

Leo Liberti
CNRS LIX, École Polytechnique, F-91128 Palaiseau, France.
E-mail: liberti@lix.polytechnique.fr.

Constantinos C. Pantelides*
Centre for Process Systems Engineering, Department of Chemical Engineering and Chemical Technology, Imperial College London, London SW7 2BY, United Kingdom.
E-mail: c.pantelides@imperial.ac.uk.

28 November 2005

Abstract. Many nonconvex nonlinear programming (NLP) problems of practical interest involve bilinear terms and linear constraints, as well as, potentially, other convex and nonconvex terms and constraints. In such cases, it may be possible to augment the formulation with additional linear constraints (a subset of Reformulation-Linearization Technique constraints) which do not affect the feasible region of the original NLP but tighten that of its convex relaxation to the extent that some bilinear terms may be dropped from the problem formulation. We present an efficient graph-theoretical algorithm for effecting such exact reformulations of large, sparse NLPs. The global solution of the reformulated problem using spatial Branch-and Bound algorithms is usually significantly faster than that of the original NLP. We illustrate this point by applying our algorithm to a set of pooling and blending global optimization problems.

Keywords: Global optimization, bilinear, convex relaxation, NLP, Reformulation-Linearization Technique, RRLT constraints.

1. Introduction

We consider the solution of Nonlinear Programs (NLPs) of the following standard form:

* Corresponding author.

$$
\begin{aligned}
[P]: \quad & \min_z \; z_l && (1)\\
& Az = b && (2)\\
& z_i = z_j z_k \qquad \forall i,j,k \in B && (3)\\
& z_i = \frac{z_j}{z_k} \qquad \forall i,j,k \in F && (4)\\
& z_i = f_i(z_j) \qquad \forall i,j \in N && (5)\\
& z^L \leq z \leq z^U && (6)
\end{aligned}
$$

where $z = (z_1, \ldots, z_p) \in \mathbb{R}^p$ are the problem variables, l is an index in the set $\{1, \ldots, p\}$, $A = (a_{ij})$ is an $M \times p$ matrix of rank M, $b \in \mathbb{R}^M$, B, F are sets of index triplets $\{(i,j,k) \mid 1 \leq i,j,k \leq p\}$, N is a set of index pairs $\{(i,j) \mid 1 \leq i,j \leq p\}$, $f_i : \mathbb{R} \to \mathbb{R}$ are nonlinear continuous univariate functions and $z^L, z^U \in \mathbb{R}^p$ are variable bounds. Note that linear fractional terms (4) can be re-arranged to bilinear ones (3); the converse is also true provided the range of the variable appearing in the denominator does not include zero. In any case, the above standard form is practically important as it can be shown that all NLPs can automatically be reformulated to it using symbolic manipulations (Smith and Pantelides, 1999; Smith, 1996). Therefore, theoretical results, manipulations and solution algorithms derived on the basis of this standard form are generally applicable.

Spatial Branch-and-Bound (sBB) algorithms (Tuy, 1998) are among the most effective methods currently available for the global solution of nonconvex NLPs. An important requirement for any sBB algorithm is to be able to construct a tight convex underestimator of the NLP within any given region of the space of the variables. For the standard form $[P]$, the lower bound to the objective function can be generated by replacing the nonconvex constraints (3), (4), (5) with their convex relaxations; in this manner, a convex relaxation of the whole problem can be obtained in a simple and completely automatic fashion.

Tight convex relaxations for the most common nonconvex terms are available in the literature. One of the best known is that proposed by (McCormick, 1976) (see Example 1.1) for the relaxation of bilinear terms. Linear fractional terms like those appearing in constraint (4) are reformulated to $z_i z_k = z_j$ and replaced by the McCormick convex relaxation. For the nonlinear terms in constraint (5), the convex relaxation depends on the function f_i. When f_i is wholly concave or wholly convex, the function itself and its secant provide the tightest convex relaxation. For functions f_i which are partially concave and partially

convex, like e.g. $z_i = z_j^{2k+1}$, where $k \in \mathbb{N}$, building the convex relaxation may not be so straightforward (Liberti and Pantelides, 2003).

1.1 Example

Consider the problem

$$\begin{aligned} \min_{x,y,z,w} \quad & z \\ z - x + w &= 1 \\ w &= xy \\ (x^L, y^L, z^L, w^L) &\leq (x, y, z, w) \leq (x^U, y^U, z^U, w^U) \end{aligned}$$

in standard form. To obtain the convex relaxation, we replace the nonconvex constraint $w = xy$ with its McCormick convex relaxation:

$$w \geq x^L y + y^L x - x^L y^L \tag{7}$$
$$w \geq x^U y + y^U x - x^U y^U \tag{8}$$
$$w \leq x^U y + y^L x - x^U y^L \tag{9}$$
$$w \leq x^L y + y^U x - x^L y^U. \tag{10}$$

This paper presents a technique to automatically reformulate optimization problems in standard form $[P]$ in such a way that some of the nonlinear constraints (3) are replaced by linear constraints. This is possible because, in certain instances, feasible regions described by nonlinear constraints are, in fact, (linear) hyperplanes. We propose an automatic algorithm to identify such instances in large scale problems. The solution of the reformulated problems by sBB algorithms often requires reduced computational effort, sometimes even by several orders of magnitude.

The creation of new linear constraints via multiplication of existing linear constraints by problem variables was proposed in (Sherali and Adams, 1986; Sherali, 2002) under the name "Reformulation-Linearization Technique" (RLT). The RLT uses linear constraints built in this way to provide a lower bound to 0-1 mixed-integer linear and bilinear programming problems. In the case of continuous bilinear programming problems (Sherali and Alameddine, 1992), the RLT generates all possible new linear constraints that can be obtained by multiplying together bound factors ($z_j - z_j^L \geq 0$, $z_j^U - z_j \geq 0$ for all $j \leq p$) and constraint factors ($\sum_j a_{ij} z_j - b_i = 0$ for all $i \leq M$), and then linearizing the resulting constraint.

The RLT is an effective tightening technique, its main drawback being the excessive size of the resulting LP relaxation. In (Sherali and Alameddine, 1992), three different methods for limiting the size of the RLT constraint set were suggested, adopting a "dynamic generation"

approach that solves the LP without the RLT constraints and only generates and includes those RLT constraints which are valid cuts with respect to the relaxed optimum. A more advanced constraint filtering technique is proposed in (Sherali and Tuncbilek, 1997), based on the sign of the coefficient of the variable z_i in the original constraint and in the RLT constraint being generated.

The approach adopted in this paper is fundamentally different from those mentioned above as it applies RLT-type multiplications directly to the original NLP problem (and not its convex relaxation) deriving an exact reformulation which has fewer bilinear terms and more linear constraints — called "reduced RLT" (RRLT) constraints. This reformulation is a pre-processing step, applied once only before the start of the solution of the global optimization problem using branch-and-bound algorithms.

The rest of this paper is organized as follows. In Section 2, we introduce the basic concepts and ideas behind RRLT constraints. Section 3 considers the existence of RRLT constraints in a more formal manner. This provides the basis of the fast algorithm for the identification of RRLT constraints presented in Section 4. An example of the application of the algorithm is presented in detail in Section 5. The effects of RRLT constraints on the global solution of an important class of problems with bilinear constraints, namely pooling and blending problems, are considered in Section 6. Finally, Section 7 proposes an extension of the algorithm of Section 4 that may result in better reformulations at the expense of higher computational time and memory.

2. Basic concepts

This section introduces in an informal manner the motivation and general ideas behind the work presented in this paper. Let y be a single problem variable, and let $w = (w_1, \ldots, w_n)$ and $x = (x_1, \ldots, x_n)$ be problem variables (with $n < p$) such that the following constraints exist in the problem:

$$\forall j \leq n \ (w_j = x_j y). \tag{11}$$

Now suppose that the problem also involves the linear constraint:

$$\sum_{j=1}^{n} a_{ij} x_j \ = \ b_i \tag{12}$$

Multiplying this constraint by y and making use of (11) leads to a new linear constraint:

$$\sum_{j=1}^{n} a_{ij} w_j - b_i y \;=\; 0. \tag{13}$$

The linear constraint (13) is redundant with respect to the original constraints. Indeed, it can be used to replace one of the bilinear constraints $w_j = x_j y$ in problem $[P]$ without affecting the feasible region of the problem. To see this, assume $a_{ik} \neq 0$ for some $k \in \{1, \ldots, n\}$ and discard the nonlinear constraint $w_k = x_k y$ from the set (11) above. We now replace b_i in (13) with the left hand side of (12) to obtain:

$$\sum_{j=1}^{n} a_{ij} w_j - \sum_{j=1}^{n} a_{ij} x_j y = \sum_{j=1}^{n} a_{ij} (w_j - x_j y) = 0.$$

Since $w_j = x_j y$ for all $j \neq k$, the above reduces to $a_{ik}(w_k - x_k y) = 0$, which implies $w_k = x_k y$. We have thus recovered the discarded bilinear constraint from the other $n-1$ bilinear constraints and the new linear constraint (13). We call the latter a *reduced RLT (RRLT) constraint.*

The geometric significance of RRLT constraints is that, given any $i \leq m$, the set:

$$\{(w, x, y) \mid \forall j \leq n \; (w_j = x_j y) \wedge \sum_{j=1}^{n} a_{ij} x_j = b_i\} \tag{14}$$

is less nonlinear than might appear to be the case: in fact, provided that $a_{ik} \neq 0$ for some $k \leq n$, it is equal to the set:

$$\{(w, x, y) \mid \forall j \neq k \; (w_j = x_j y) \wedge \sum_{j=1}^{n} a_{ij} x_j = b_i \wedge \sum_{j=1}^{n} a_{ij} w_j - b_i y = 0\}, \tag{15}$$

where the kth bilinear constraint has been replaced by a linear constraint.

Below, we consider an extreme example of such a reformulation.

2.1 Example
Let $(w, x, y) \in \mathbb{R}^3$ and consider the set:

$$A_1 = \{(w, x, y) \mid w = xy \wedge x = 1\}.$$

This set is defined as the intersection of the bilinear surface xy and the vertical plane $x = 1$ (see Figure 1). By multiplying $x = 1$ by y, we get the linear constraint $w - y = 0$ (the skew plane in Figure 1). It is evident that the set

$$A_2 = \{(w, x, y) \mid x = 1 \wedge w - y = 0\}$$

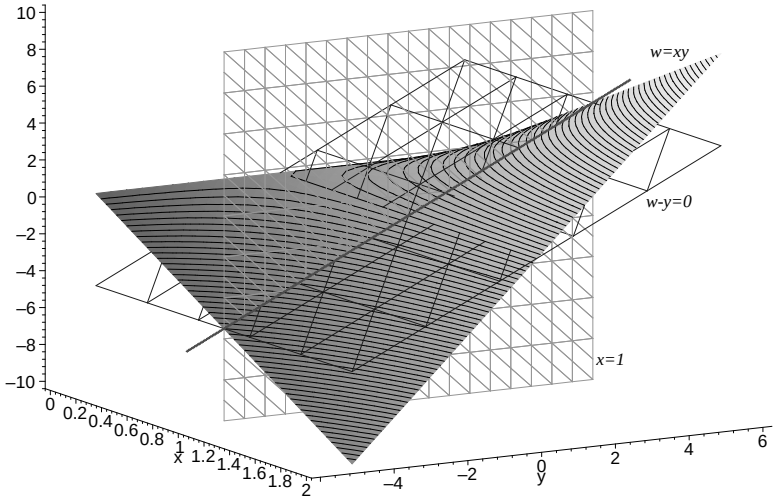

Figure 1. Linearity embedded in a bilinear constraint.

is the same as the set A_1. However, A_2 is defined only by linear relationships whereas A_1 is not. The convexification of A_1 in the context of an sBB algorithm (e.g. by introducing the McCormick relaxation of xy) would be both unnecessary and counter-productive.

A limitation of the reformulation technique presented above is that all constraints of the type $w_j = x_j y$ have to exist in the problem (1) before the linear constraint (13) can be created. Consider, however, a problem that already includes bilinear constraints $w_j = x_j y$ for all $j \leq n-1$, and two linear constraints:

$$\sum_{j=1}^{n-1} a_{ij} x_j + a_{in} x_n \ = \ b_i \tag{16}$$

$$\sum_{j=1}^{n-1} a_{hj} x_j + a_{hn} x_n \ = \ b_h \tag{17}$$

where $i, h \leq m$ and a_{in}, a_{hn} are nonzero constants. On multiplying (16) and (17) by y, we obtain the two linear constraints:

$$\sum_{j=1}^{n-1} a_{ij} w_j + a_{in} w_n - b_i y \;=\; 0 \tag{18}$$

$$\sum_{j=1}^{n-1} a_{hj} w_j + a_{hn} w_n - b_h y \;=\; 0. \tag{19}$$

where w_n is a new variable defined via a new bilinear constraint $w_n = x_n y$. By forming a suitable linear combination of the two constraints, the new variable w_n can be eliminated, and we now have a linear constraint similar to (13) which can be used to eliminate one of the original bilinear constraints in the manner indicated earlier.

3. Fundamental properties

We now proceed to provide a more rigorous foundation for the ideas introduced in section 2

Consider an optimization problem in standard form $[P]$ and subsets of problem variables $w, x \in \mathbb{R}^n$ and $y \in \mathbb{R}$. Suppose the problem includes the set of m linear equality constraints $Ax = b$, where the matrix A is of full row rank m.

Multiplying $Ax = b$ by y, we create m RRLT constraints of the form $Aw - yb = 0$, where the variable vector $w \in \mathbb{R}^n$ is defined as in Equation (11). The following theorem shows that the RRLT constraints can be used to replace m of the n bilinear constraints in (11) without changing the feasible region of the problem.

3.1 Theorem
Let $J \subseteq \{1, \ldots, n\}$ be an index set of cardinality $|J| = n - m$ and consider the sets:

$$C \;=\; \{(x, w, y) \mid Ax = b \wedge \forall j \leq n \; (w_j = x_j y)\} \tag{20}$$

$$R_J \;=\; \{(x, w, y) \mid Ax = b \wedge Aw - yb = 0 \wedge \forall j \in J \; (w_j = x_j y)\} \tag{21}$$

Then, there is at least one set J such that $C = R_J$.

Proof. The fact that $C \subseteq R_J$ for any set J is straightforward: if $(x, w, y) \in C$, then it also satisfies the constraints $Aw - yb = 0$, hence $(x, w, y) \in R_J$.

We shall now prove the converse inclusion. By virtue of the fact that $Ax = b$, the RRLT constraint system $Aw - yb = 0$ can be written as

$Aw - (Ax)y = 0$. If we now define $u_j = w_j - x_j y, \forall j \leq n$, this can further be written as the linear homogeneous system $Au = 0$. Since $\text{rank}(A) = m$, there exists a permutation π of the columns of A such that:

$$(A_0 \; A_1) \begin{pmatrix} u^{(0)} \\ u^{(1)} \end{pmatrix} = 0,$$

where A_0 is a non-singular $m \times m$ matrix, and $(u^{(0)^T}, u^{(1)^T})^T$ is a permutation of u with $u^{(0)} \in \mathbb{R}^m$ and $u^{(1)} \in \mathbb{R}^{n-m}$. Let J be the image of $\{n - m + 1, \ldots, n\}$ under the permutation π, and let $(x, w, y) \in R_J$. Since (x, w, y) satisfies $w_j = x_j y$ for all $j \in J$, then $u^{(1)} = 0$; since (x, w, y) also satisfies the RRLT constraint system, we have $Au = 0$, which implies $u^{(0)} = 0$ as well. Hence $w_j = x_j y$ for all $j \leq n$. □

The geometrical implications of the above theorem are that the intersection in $\mathbb{R}^n$ of a set of bilinear terms like those described above, and the linear form $Ax = b$ is a hypersurface containing a degree of linearity. By exploiting this linearity, we are able to replace some of the bilinear terms with linear constraints.

There is an interesting analogy between the above result and some of the RLT work carried out on persistency in MILPs with binary variables (Adams et al., 1998). It has been shown, in particular, that, if some of the binary variables attain integer values at the optimum of the relaxation, the added RLT constraints guarantee that those variables will have the same values at the optimum of the integer problem. Theorem 3.1 shows that enforcing a proper subset of the set of all bilinear constraints and a particular set of RLT constraints implies that the rest of the bilinear constraints are also satisfied. Propositions 8.11 and 8.12 in (Sherali and Adams, 1999) show a similar result to our Theorem 3.1 but with stronger assumptions. In particular, they show that the bilinear terms are automatically implied by the RLT relaxation provided a certain subset of the problem variables is at the range bound. Our result, on the other hand, does not make any assumptions regarding the variable values.

As has been shown above, the bilinear terms that have been eliminated are redundant with respect to the remaining terms and the newly introduced linear constraints. This does not, however, necessarily imply that the McCormick relaxations (cf. Equations (7)-(10)) of these eliminated terms are also redundant as far as the problem's convex relaxation is concerned. It is instructive to examine this point in more detail by considering a problem that involves a linear constraint

written, without loss of generality, in the form:

$$z_i = \sum_{j\in\mathcal{P}} \alpha_j z_j + \sum_{j\in\mathcal{N}} \alpha_j z_j + c \tag{22}$$

for a certain variable z_i, where the index sets $\mathcal{P}$ and $\mathcal{N}$ are such that $\alpha_j > 0, \forall j \in \mathcal{P}$ and $\alpha_j < 0, \forall j \in \mathcal{N}$. The problem also involves bilinear terms $w_j = z_j y, j \in \mathcal{P}\cup\mathcal{N}\cup\{i\}$ where y is one of the problem variables. The convex relaxation of this problem will involve the McCormick relaxations (7)-(10) of all the bilinear terms w_j.

In the spirit of the reformulations considered in this paper, we now multiply (22) by y to obtain:

$$w_i = \sum_{j\in\mathcal{P}} \alpha_j w_j + \sum_{j\in\mathcal{N}} \alpha_j w_j + cy \tag{23}$$

and then use this constraint to eliminate the bilinearity $w_i = z_i y$ from the problem. The convex relaxation of this problem will not include the McCormick constraints for w_i, but will still include those for $w_j, j \in \mathcal{P} \cup \mathcal{N}$.

Now consider multiplying the McCormick constraints (7) for $w_j, j \in \mathcal{P}$ by α_j, and the constraints (9) for $w_j, j \in \mathcal{N}$ by α_j, and adding all of these together to obtain:

$$\sum_{j\in\mathcal{P}} \alpha_j w_j + \sum_{j\in\mathcal{N}} \alpha_j w_j \;\geq\; \left(\sum_{j\in\mathcal{P}} \alpha_j z_j^L \;+\; \sum_{j\in\mathcal{N}} \alpha_j z_j^U\right)(y - y^L) + \left(\sum_{j\in\mathcal{P}} \alpha_j z_j + \sum_{j\in\mathcal{N}} \alpha_j z_j\right) y^L$$

By adding cy to both sides of the equation and making use of (22) and (23), the above simplifies to:

$$w_i \geq \left(\sum_{j\in\mathcal{P}} \alpha_j z_j^L + \sum_{j\in\mathcal{N}} \alpha_j z_j^U + c\right)(y - y^L) + y^L z_i \tag{24}$$

We now compare (24) with the McCormick constraint (7) for variable w_i, which can be written as:

$$w_i \geq z_i^L (y - y^L) + y^L z_i. \tag{25}$$

We can see that constraint (25) will be tighter than (24) if:

$$z_i^L > \sum_{j\in\mathcal{P}} \alpha_j z_j^L + \sum_{j\in\mathcal{N}} \alpha_j z_j^U + c \tag{26}$$

and, in such a case, the convex relaxation of the original problem (which contains constraint (25)) will be tighter than that of the reformulated problem (which does not).

Inequality criteria similar to (26) may be obtained for the other three McCormick constraints relating to the eliminated bilinear term w_i, as shown in Table I. Thus, the McCormick constraint for w_i shown in column 3 will be tighter than that implied by the McCormick constraints for w_j for $j \in \mathcal{P}$ and $j \in \mathcal{N}$ shown in the first two columns respectively, provided the inequality criterion shown in column 4 holds.

Table I. Comparison of tightness of convex relaxations for original and reformulated problems

McCormick constraints for $w_j, j \in \mathcal{P}$	McCormick constraints for $w_j, j \in \mathcal{N}$	McCormick constraint for w_i	Original convex relaxation tighter than reformulated one if
(7)	(9)	(7)	$z_i^L > \sum_{j\in\mathcal{P}} \alpha_j z_j^L + \sum_{j\in\mathcal{N}} \alpha_j z_j^U + c$
(8)	(10)	(8)	$z_i^U < \sum_{j\in\mathcal{P}} \alpha_j z_j^U + \sum_{j\in\mathcal{N}} \alpha_j z_j^L + c$
(9)	(7)	(9)	$z_i^L > \sum_{j\in\mathcal{P}} \alpha_j z_j^L + \sum_{j\in\mathcal{N}} \alpha_j z_j^U + c$
(10)	(8)	(10)	$z_i^U < \sum_{j\in\mathcal{P}} \alpha_j z_j^U + \sum_{j\in\mathcal{N}} \alpha_j z_j^L + c$

The inequalities in the last column of Table I may hold because of the original bounds imposed on the problem variables, but this may cease to be true after a certain degree of branching on the $z_j, j \in \mathcal{P}\cup\mathcal{N}$ variables during the solution of the reformulated problem. Thus, the inequalities are likely to stay (or become) true primarily due to branching on the variable z_i; of course, the reformulated problem will not normally branch on z_i as this no longer appears in a bilinear term.

In summary, some of the bilinear terms are redundant with respect to the nonlinear problem in its reformulated form but not necessarily in the corresponding convex relaxations. However, the additional tightness induced by this non-redundancy can be realized primarily at the expense of additional branching on the eliminated bilinear terms. In any case, as will be shown formally in section 4.1, the addition of the RRLT constraints always tightens the NLP's convex relaxation.

4. An algorithm for the identification of valid RRLT constraints

Section 3 has established that, in principle, RRLT constraints exist for any NLP involving bilinear terms and linear constraints. Here we are

concerned with two more practical questions. Section 4.1 introduces a criterion for determining whether the introduction of RRLT constraints is likely to be beneficial. Then Sections 4.2 and 4.3 present an efficient algorithm for the identification of such RRLT constraints based on the above criterion; this is particularly important in the case of large NLPs for which neither identification by inspection nor indiscriminate multiplication of linear constraints by problem variables are practical propositions.

4.1. Valid RRLT constraint sets

Given an NLP in the standard form [P], consider a subset $\mathcal{L}$ of its linear constraints (2). Let the set of variables that occur in these constraints be denoted by $\mathcal{V}(\mathcal{L})$, i.e. $\mathcal{V}(\mathcal{L}) \equiv \{k \mid \exists l \in \mathcal{L}\ (a_{lk} \neq 0)\}$.

Now consider multiplying the linear constraint set $\mathcal{L}$ by a problem variable z_l. This will create bilinear terms of the form $z_l z_k, \forall k \in \mathcal{V}(\mathcal{L})$. Some of these terms will already occur in the NLP, i.e. $\exists i : (i,l,k) \in B$. For each variable index l, let $K(l,\mathcal{L})$ be a subset of $\mathcal{V}(\mathcal{L})$ that leads to *new* bilinear terms, i.e.:

$$K(l,\mathcal{L}) \equiv \{k \mid k \in \mathcal{V}(\mathcal{L}) \wedge \nexists i : (i,l,k) \in B\}$$

The theorem of Section 3 indicates that we can now construct a problem that has the same feasible region as $[P]$ by replacing $|\mathcal{L}|$ bilinear constraints by linear constraints. The latter are said to form a *valid RRLT constraint set* if the substitution leads to a reduction in the overall number of bilinear constraints in comparison with the original problem $[P]$, i.e. if:

$$|K(l,\mathcal{L})| \ < \ |\mathcal{L}| \tag{27}$$

Valid RRLT constraints do not modify the feasible region of the original NLP, but they do reduce the size of the feasible region of its convex relaxation. We prove this in Theorem 4.1 below. Here the bounds z^L, z^U are assumed to be the tightest possible, i.e. for any index $i = 1, ..., p$ and value $z_i^* \in [z_i^L, z_i^U]$, the set $\{z | Az = b \wedge z \in [z^L, z^U] \wedge z_i = z_i^*]\}$ is non-empty.

4.1 Theorem
Let $l \leq p$ be a variable index, and let $Az = b$ be a subset $\mathcal{L}$ of $\nu \leq M$ linear constraints (2) which satisfies (27). Consider the following sets:

$$\begin{aligned} C_l \ &\equiv \ \{(z,w) \mid Az = b \wedge z \in [z^L, z^U] \wedge w_i = z_l z_j \ \forall (i,l,j) \in B \\ &\qquad \wedge\, z_l^U > z_l^L \wedge z_j^U > z_j^L\} \\ R_l \ &\equiv \ C_l \cap \{(z,w) \mid Aw = z_l b\}. \end{aligned}$$

Suppose that C_l is non empty, and let $\bar{C}_l, \bar{R}_l$ be the corresponding convex relaxations obtained by replacing the bilinear constraints with the respective McCormick envelopes. Then $\bar{R}_l$ is a proper subset of $\bar{C}_l$.

Proof. By definition we have $\bar{R}_l \subseteq \bar{C}_l$. We have to prove that $\bar{R}_l \neq \bar{C}_l$. The system $Az = b$ can be written in the form:

$$A'z' + A''z'' \ = \ b'$$

where the partitioning is such that the bilinear terms $w' \equiv z_l z'$ already occur in the original NLP, while $w'' \equiv z_l z''$ do not. Since condition (27) holds, we must have $z'' \in \mathbb{R}^{\nu'}$ where $\nu' = |K(l, \mathcal{L})| < \nu$, and we can apply Gaussian elimination with row pivoting on A'' to bring the above system to the form:

$$\begin{pmatrix} \tilde{A}' \\ \bar{A}' \end{pmatrix} z' + \begin{pmatrix} \tilde{A}'' \\ 0 \end{pmatrix} z'' = \begin{pmatrix} \tilde{b}' \\ \bar{b}' \end{pmatrix}. \tag{28}$$

By multiplying the bottom block row of the above equation by z_l, we obtain:

$$\bar{A}'w' \ = \ z_l \bar{b}'. \tag{29}$$

where the matrix $\bar{A}'$ must have full rank, otherwise the assumption of full rank of the original set of linear constraints (2) would be contradicted.

Consider a point $(z, w) \in \bar{R}_l$; naturally, it satisfies (29) and we also have that $(z, w) \in \bar{C}_l$. By definition of C_l, each of the variables z_l or z_j involved in bilinear products has a non-zero range. Therefore, the McCormick envelopes for each and every element of the vector w' (where $w = (w', w'')$ as above) are non-empty, compact and of positive volume. Consequently, there exists a direction in which w' can be perturbed by a non-zero amount and still be feasible in $\bar{C}_l$. In other words, there must be an index i such that, if $w'_{[i]}$ is the vector obtained by w' by perturbing the i-th component by a non-zero amount, the vector $(z, w'_{[i]})$ is in $\bar{C}_l$. Let us now assume that also $(z, w'_{[i]}) \in \bar{R}_l$, i.e.:

$$\bar{A}'w'_{[i]} \ = \ z_l \bar{b}'. \tag{30}$$

By subtracting (29) from (30), we obtain:

$$\bar{A}'(w'_{[i]} - w') \ = \ 0. \tag{31}$$

However, equation (31) cannot hold for all i, as this would imply that $\bar{A}'$ is the zero matrix. Therefore, starting from a point $w' \in \bar{R}_l$,

we can always construct a perturbed vector $w'_{[i]}$ such that $w'_{[i]} \in \bar{C}_l$ but $w'_{[i]} \notin \bar{R}_l$, which implies that $\bar{R}_l$ is a proper subset of $\bar{C}_l$. □

The above theorem makes use of the fact that bilinear envelopes are loose everywhere other than at the variable bounds (Al-Khayyal and Falk, 1983) to show that valid RRLT constraint sets are always useful, since they tighten the convexification of the original problem. However, despite the apparent simplicity of criterion (27), applying it in a brute force manner as a means of identifying RRLT constraint systems is impractical: for an NLP containing M linear constraints, one would have to examine $2^M - 1$ subsets $\mathcal{L}$ for each candidate multiplier variable z_l.

4.2. A graph-theoretical representation of linear constraints

Here we consider a fast graph-theoretical algorithm for the identification of linear constraint subsets $\mathcal{L}$ that satisfy (27) for a given multiplier variable z_l. We start by constructing a bipartite graph (Korte and Vygen, 2000; Goodaire and Parmenter, 1998; Harary, 1971) $\mathcal{G}_l$ where the set of nodes is partitioned into two disjoint subsets $\mathcal{N}^L$ and $\mathcal{N}_l^V$. The former correspond to the set of linear constraints in $[P]$ while the latter correspond to those problem variables which do not appear in any bilinear term (3) multiplied by z_l, i.e.:

$$\mathcal{N}_l^V \equiv \{k | \nexists i : (i, l, k) \in B\}$$

The set of edges $\mathcal{E}_l$ in graph $\mathcal{G}_l$ is defined as:

$$\mathcal{E}_l \equiv \{(\lambda, k) | \lambda \in \mathcal{N}^L \wedge k \in \mathcal{N}_l^V \wedge a_{\lambda k} \neq 0\}$$

Thus, edge (λ, k) exists if variable z_k occurs in linear constraint λ.

Suppose, now, that $\mathcal{L}$ is a subset of the nodes $\mathcal{N}^L$. Let $\bar{K}(l, \mathcal{L})$ denote the subset of nodes $\mathcal{N}_l^V$ that are connected to nodes in $\mathcal{L}$, i.e.:

$$\bar{K}(l, \mathcal{L}) \equiv \{k | k \in \mathcal{N}_l^V \wedge \exists \lambda \in \mathcal{L} : (\lambda, k) \in \mathcal{E}_l\}$$

In can be verified that, in view of the definitions of sets $\mathcal{V}$, $\mathcal{N}_l^V$ and $\mathcal{E}_l$, the two sets $K(l, \mathcal{L})$ and $\bar{K}(l, \mathcal{L})$ are, in fact, identical. Consequently, a valid RRLT constraint set (i.e. one that satisfies criterion (27)) will correspond to a *dilation* in graph $\mathcal{G}_l$, i.e. a subset $\mathcal{L}$ of, say, ν nodes of $\mathcal{N}^L$ that are connected to fewer than ν nodes of $\mathcal{N}_l^V$. This provides a practical basis for the identification of valid RRLT constraint sets using efficient existing algorithms for the determination of dilations in bipartite graphs.

4.3. Efficient identification of dilations

Given a candidate multiplier variable z_l, the algorithm shown in Figure 2 returns the corresponding valid RRLT constraint set $\mathcal{L}_l$. The algorithm works by constructing the bipartite graph $\mathcal{G}_l$ for each candidate multiplier variable z_l, and then identifying dilations in this graph. Dilations in bipartite graphs are closely related to matchings, i.e. subsets μ of the edges such that no node is adjacent to more than one edge in μ. Given a (possibly empty) matching μ, a matching of larger cardinality may be obtained if one can identify an alternating augmenting path p, i.e. one whose first and last nodes are not adjacent to any edge in μ, and whose edges alternate between not belonging and belonging to μ. If such a path can be found, then a matching of cardinality $|\mu| + 1$ can be obtained from μ by replacing all edges in $\mu \cap p$ by edges $p \backslash \mu$ (Hopcroft and Karp, 1973; Korte and Vygen, 2000). A recursive procedure for the identification of an augmenting path emanating from a node λ is described in (Duff, 1981).

The theoretical complexity of the algorithm is at most of the order (number of constraint nodes $\times$ number of edges). However, as observed by (Duff, 1981), the computational complexity of such algorithms in practice is usually nearer to (number of constraint nodes + number of edges). This results in an efficient algorithm that is practically applicable to large problems.

5. A detailed example

In order to illustrate the operation of the algorithm proposed in Section 4, we consider the following bilinear optimization problem in standard form:

$$\left.\begin{array}{ll} \min_z & z_{24} \\ & z_{24} = c^T z \\ & Az' = b \\ & z_i = z_j z_k \quad \forall (i,j,k) \in B \\ & 0 \leq z \leq 10, \end{array}\right\} \tag{32}$$

where:

$$c^T = (0,0,0,0,0,0,1,1,3,1,2,2,1,2,1,-1,2,-1,4,3,6,9,1),$$

$z = (z_1, \ldots, z_{23})^T$, $z' = (z_1, \ldots, z_6)^T$, $b = (1,2,-1,1)^T$,

$$A = \begin{pmatrix} 1 & & 2 & & 1 & 1 \\ 2 & -1 & & 1 & & 3 \\ & 1 & & 6 & 2 & -3 \\ 2 & & & 1 & 3 & \end{pmatrix}$$

```
1 PROCEDURE ValidReducedRLTConstraints(l, L_l)

2 Set L_l := ∅
3 Construct bipartite graph G_l as described in Section 4.2
4 FOR each variable node k DO
5      Set Assigned[k] := 0
6 END FOR

7 FOR each constraint node λ DO
8      FOR each constraint node λ' DO
9          Set ConstraintVisited[λ'] := FALSE
10      END FOR
11      FOR each variable node k DO
12          Set VariableVisited[k] := FALSE
13      END FOR
14      Search for augmenting path from node λ

15      IF path not found THEN
16          Set L_l := L_l ∪ {λ'|ConstraintVisited[λ'] = TRUE}
17      END IF
18 END FOR

19 END ValidReducedRLTConstraints
```

Figure 2. Algorithm for identification of set of valid RRLT constraint set $\mathcal{L}$ for variable z_l.

and:

$$\begin{aligned} B = \{&(7,1,1),(8,2,2),(9,4,4),(10,5,5),(11,6,6),(12,1,4),\\ &(13,1,5),(14,1,6),(15,2,4),(16,2,5),(17,2,6),(18,3,4),\\ &(19,3,5),(20,3,6),(21,4,5),(22,4,6),(23,5,6)\ \} \end{aligned} \tag{33}$$

The above is, effectively, a bilinear programming problem involving 6 variables $z_1, \ldots, z_6$ which are, in turn, related via 4 linear constraints. The problem involves all bilinear combinations (including simple squares) of these 6 variables apart from z_1z_2, z_1z_3, z_2z_3, z_3^2. New variables $z_7, \ldots, z_{23}$ have been introduced to represent these bilinear terms, and these are linearly combined to form the objective function represented by variable z_{24}.

In this problem, it is quite easy to see that only variables $z_1, \ldots, z_6$ can possibly give rise to RRLT constraints when multiplied by the linear constraints $Az' = b$. We consider each of these 6 variables in turn, applying to it procedure `ValidReducedRLTConstraints` of Figure 2.

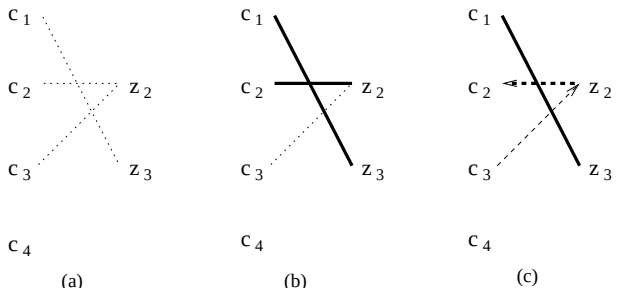

Figure 3. Generation of valid RRLT constraints for multiplier variable z_1.

5.1. Valid RRLT constraints by multiplication with variable z_1

The bipartite graph $\mathcal{G}_1$ constructed as described in Section 4.2 for variable z_1 is shown in Figure 3a. Nodes $c_1, \ldots, c_4$ correspond to the four linear constraints. Edges correspond to the creation of new bilinear terms if a particular constraint were to be multiplied by z_1. For example, edges (c_1, z_1) and (c_1, z_3) exist because, if constraint c_1 were to be multiplied by z_1, then this would give rise to new terms z_1^2 and $z_1 z_3$ respectively.

As there are initially no assignments of variable nodes to constraint nodes, all edges are shown as dotted lines in Figure 3a. We now consider each constraint node in turn (lines 7-18 of Figure 2), attempting to construct an augmenting path emanating from it (line 14).

Starting from constraint node c_1, an augmenting path is found immediately to its adjacent variable node z_3. This is assigned to c_1; this assignment is indicated as a solid line in Figure 3b.

Considering constraint c_2 also results in an augmenting path being found immediately, this time assigning z_2 to c_2. The resulting assignment is also shown as a solid line in Figure 3b.

So far, we have not identified any dilations. However, if we now attempt to construct an augmenting path emanating from constraint node c_3, we note that c_3 is adjacent to only one variable node, namely z_2 which is already assigned to node c_2. Thus, there are no augmenting paths starting from c_3, visiting nodes c_3, c_2 in the process. Consequently, line 16 of Figure 2 adds c_2 and c_3 to the set of linear constraints $\mathcal{L}_1$ to be multiplied by z_1.

Finally, constraint node c_4 is completely isolated in $\mathcal{G}_1$, i.e. there are no edges incident to it. Thus, c_4 is also added to set $\mathcal{L}_1$.

As there are no more constraint nodes to be considered, we conclude that $\mathcal{L}_1 = \{c_2, c_3, c_4\}$. Essentially, the algorithm has identified

automatically that multiplying constraints c_2:

$$2z_1 - z_2 + z_4 + 3z_6 = 2$$

and c_3:

$$z_2 + 6z_4 + 2z_5 - 3z_6 = -1$$

by z_1 creates only one new bilinear term, namely z_1z_2, beyond those that already exist in the problem. Consequently, if we define a new variable $z_{25} \equiv z_1z_2$, we end up with two linear constraints:

$$2z_7 - z_{25} + z_{12} + 3z_{14} = 2z_1$$

and

$$z_{25} + 6z_{12} + 2z_{13} - 3z_{14} = -z_1$$

respectively that can be used to eliminate two bilinear terms from the problem.

Moreover, the algorithm has also detected that multiplying constraint c_4:

$$2z_1 + z_4 + 3z_5 = 1$$

by z_1 does not generate any new bilinear terms. The new linear constraint:

$$2z_7 + z_{12} + 3z_{13} = z_1$$

obtained by this multiplication can be used to eliminate one more of the problem's bilinear terms. Overall, the number of bilinear terms in the problem can be reduced by two via the use of valid RRLT constraints.

5.2. Valid RRLT constraints by multiplication with variables z_2 or z_3

The bipartite graphs $\mathcal{G}_2$ and $\mathcal{G}_3$ are shown in Figures 4 and 5 respectively, together with a graphical depiction of the application of the algorithm steps to them. In these figures, solid lines indicate edges that belong to the current matching. Dashed arrows show an alternating augmenting path being traced from a constraint node; a thicker dashed line (going from a z to a c node) indicates edges on this path that are part of the current matching, while thin dashed lines (going from a c to a z node) are edges that do not belong to the current matching.

It turns out that $\mathcal{L}_2 = \{c_2, c_3, c_4\}$. In this case, the algorithm has detected that c_3 can be multiplied by z_2 without generating any new bilinear terms. Also, the multiplication of c_2 and c_4 by z_2 leads to the creation of only one new bilinear term, z_1z_2 and two linear constraints.

The algorithm also determines that $\mathcal{L}_3 = \{c_2, c_3, c_4\}$. The alternating path $c_4 \to z_1 \to c_2 \to z_2 \to c_3$ (shown in Figure 5(e)) indicates that

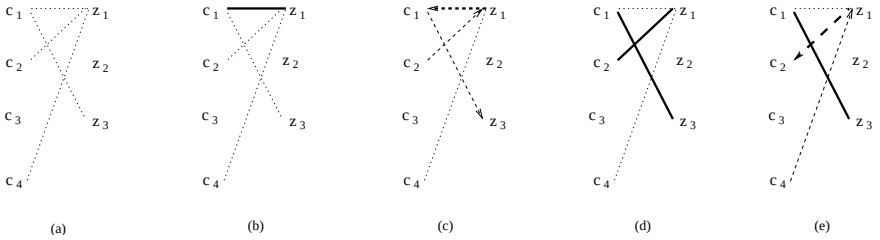

Figure 4. Generation of valid RRLT constraints for multiplier variable z_2.

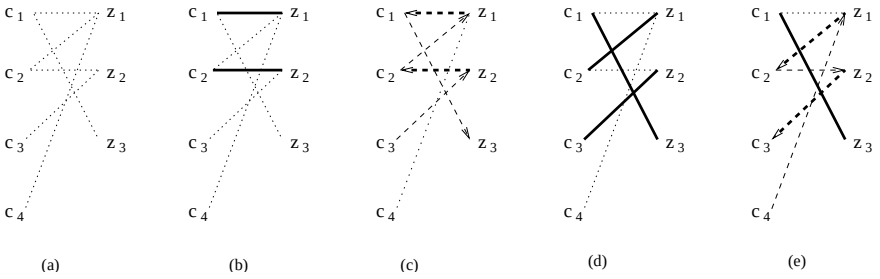

Figure 5. Generation of valid RRLT constraints for multiplier variable z_3.

multiplying these three constraints by z_3 will result in the creation of only two new bilinear terms, namely z_1z_3 and z_2z_3

5.3. Valid RRLT constraints by multiplication with variables z_4, z_5 or z_6

The bipartite graphs $\mathcal{G}_l$ for $l = 4, 5, 6$ are very simple as they contain no variable nodes (i.e. $\mathcal{N}_l^V = \emptyset$). This results in the constraint sets:

$$\mathcal{L}_l = \{c_1, c_2, c_3, c_4\},\ l = 4, 5, 6$$

This is just a consequence of the fact that this problem already contains all possible bilinear terms involving z_4 or z_5 or z_6. Consequently, multiplying any linear constraint by one of these variables does not result in any new bilinear terms and can, therefore, be used to generate a valid RRLT constraint.

5.4. The reformulated NLP

In summary, the operations described in Sections 5.1-5.3 above result in a set of 21 RRLT constraints obtained by multiplying each constraint subset $\mathcal{L}_l, l = 1, \ldots 6$ by variable z_l. Three new bilinear terms were introduced: $z_{25} = z_1z_2$, $z_{26} = z_1z_3$, $z_{27} = z_2z_3$, augmenting the

original set B (cf. Equation (3)) with the triplets $(25, 1, 2)$, $(26, 1, 3)$ and $(27, 2, 3)$.

In order to determine which bilinear terms may be replaced by these linear constraints, we write the latter in matrix form:

$$Bz'' + Cz' \;=\; 0 \tag{34}$$

where $z' = (z_1, \ldots, z_6)^T \in \mathbb{R}^6$ and $z'' = (z_7, \ldots, z_{23}, z_{25}, \ldots, z_{27})^T \in \mathbb{R}^{20}$, $B : \mathbb{R}^{20} \to \mathbb{R}^{21}$ and $C : \mathbb{R}^6 \to \mathbb{R}^{21}$. By performing Gaussian elimination with row pivoting on B (and replicating the same row operations on C) we obtain a system of the form:

$$\begin{pmatrix} U & \tilde{B} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \xi \\ \zeta \end{pmatrix} = \begin{pmatrix} \tilde{C} \\ 0 \end{pmatrix} z'$$

where $\xi = (z_7, \ldots, z_{23}, z_{25})^T \in \mathbb{R}^{18}$, $\zeta = (z_{26}, z_{27})^T \in \mathbb{R}^2$, $U : \mathbb{R}^{18} \to \mathbb{R}^{18}$ is a nonsingular, upper-triangular matrix, $\tilde{B} : \mathbb{R}^2 \to \mathbb{R}^{18}$ and $\tilde{C} : \mathbb{R}^6 \to \mathbb{R}^{18}$. Thus, the RRLT constraints determine the variables ξ in terms of ζ and z' via the solution of $U\xi = \tilde{C}z' - \tilde{B}\zeta$, and consequently, the corresponding triplets (i, j, k) can be deleted from the set B in Equation (33). Overall, the reformulation of the original problem (32) involves the introduction of the RRLT constraints (34) and a much smaller set of just two triplets, namely $B = \{(26, 1, 3), (27, 2, 3)\}$.

6. Computational results

We have chosen a selection of bilinear test problems relating to pooling and blending problems to test the efficiency of our algorithm. Pooling and blending involves the determination of optimal amounts of different raw materials that need to be mixed to produce required amounts of end-products with desired properties. Such problems occur frequently in the petrochemical industry and are well known to exhibit multiple local minima. There is a wealth of publications on the pooling problem in the literature (Foulds et al., 1992; Sherali and Alameddine, 1992; Visweswaran and Floudas, 1993; Ben-Tal et al., 1994; Quesada and Grossmann, 1995; Visweswaran and Floudas, 1996; Adhya et al., 1999; Tawarmalani and Sahinidis, 2004; Audet et al., 2004; Tawarmalani and Sahinidis, 2002a; Sahinidis and Tawarmalani, 2005). Application of RLT-type constraints to the problem formulation was carried out both explicitly (Sherali and Alameddine, 1992; Quesada and Grossmann, 1995) and implicitly (Tawarmalani and Sahinidis, 2004; Tawarmalani and Sahinidis, 2002a) as part of the so-called *pq*-formulation.

Here we wish to investigate the effects of our problem reformulation carried out as a pre-processing step. We note that the formulation proposed in (Audet et al., 2004) is expressed in such a way (bilinear products distributed over sums) that the benefits of the application of reduced RLT constraints are already included (see below). However, our main objective is to investigate the extent to which a completely *general* graph-theoretical procedure can *automatically* derive formulations that are comparable in efficiency to the best known formulations for specific problem classes (such as pooling and blending).

Here, we use the general blending problem formulation found in (Adhya et al., 1999):

$$\min_{f,q,x} \sum_{j=1}^{p} \sum_{i=1}^{n_j} c_{ij} f_{ij} - \sum_{k=1}^{r} d_k \sum_{j=1}^{p} x_{jk} \tag{35}$$

$$\sum_{i=1}^{n_j} f_{ij} - \sum_{k=1}^{r} x_{jk} = 0, \quad \forall j \leq p \tag{36}$$

$$q_{jw} \sum_{k=1}^{r} x_{jk} - \sum_{i=1}^{n_j} \lambda_{ijw} f_{ij} = 0, \quad \forall j \leq p \; \forall w \leq l \tag{37}$$

$$\sum_{j=1}^{p} x_{jk} \leq S_k, \quad \forall k \leq r \tag{38}$$

$$\sum_{j=1}^{p} q_{jw} x_{jk} - Z_{kw} \sum_{j=1}^{p} x_{jk} \leq 0, \quad \forall k \leq r \; \forall w \leq l \tag{39}$$

$$f^L \leq f \leq f^U, q^L \leq q \leq q^U, x^L \leq x \leq x^U, \tag{40}$$

where f_{ij} is the flow of input stream i into pool j, x_{jk} is the total flow from pool j to product k and q_{jw} is the w-th quality of pool j; p is the number of pools, r the number of products, l the number of qualities, n_j the number of streams; $c_{ij}, d_k, S_k, Z_{kw}, \lambda_{ijw}$ are given parameters.

When the blending problem (35)-(40) is reformulated to the standard form $[P]$, new variables u_j are created to replace the term $\sum_{k=1}^{r} x_{jk}$ in constraint set (37). Thus, we introduce the linear constraints:

$$\sum_{k=1}^{r} x_{jk} - u_j \; = \; 0, \quad \forall j \leq p \tag{41}$$

and re-write (37) as

$$t_{jw} - \sum_{i=1}^{n_j} \lambda_{ijw} f_{ij} = 0, \quad \forall j \leq p \;\; \forall w \leq l$$

where we have introduced new variables t_{jw} derived via the bilinear constraints:

$$t_{jw} = q_{jw}u_j, \quad \forall j \leq p \ \ \forall w \leq l.$$

More new variables z_{jwk} are created in the standard form to replace the bilinear terms $q_{jw}x_{jk}$ in constraint set (39):

$$z_{jwk} = q_{jw}x_{jk}, \quad \forall j \leq p \ \ \forall w \leq l \ \ \forall k \leq r$$

which allows (39) to be re-written in linear form as:

$$\sum_{j=1}^{p} z_{jwk} - Z_{kw} \sum_{j=1}^{p} x_{jk} \leq 0, \quad \forall k \leq r \ \forall w \leq l.$$

The standard form reformulation of (35)-(40) is shown below:

$$\min_{f,q,x} \sum_{j=1}^{p} \sum_{i=1}^{n_j} c_{ij} f_{ij} - \sum_{k=1}^{r} d_k \sum_{j=1}^{p} x_{jk}$$

subject to linear constraints:

$$\begin{aligned}
\sum_{i=1}^{n_j} f_{ij} - \sum_{k=1}^{r} x_{jk} = 0 \quad & \forall j \leq p \\
t_{jw} - \sum_{i=1}^{n_j} \lambda_{ijw} f_{ij} = 0 \quad & \forall j \leq p \ \forall w \leq l \\
\sum_{k=1}^{r} x_{jk} - u_j = 0 \quad & \forall j \leq p \\
\sum_{j=1}^{p} x_{jk} \leq S_k \quad & \forall k \leq r \\
\sum_{j=1}^{p} z_{jwk} - Z_{kw} \sum_{j=1}^{p} x_{jk} = 0 \quad & \forall k \leq r \ \forall w \leq l
\end{aligned}$$

with bilinear terms:

$$B = \{(t_{jw}, q_{jw}, u_j), (z_{jwk}, q_{jw}, x_{jk}) \mid j \leq p, w \leq l, k \leq r\}$$

and bounds:

$$f^L \leq f \leq f^U, \quad q^L \leq q \leq q^U, \quad x^L \leq x \leq x^U.$$

We now apply the algorithm of this paper to the above standard form. This results in a set of RRLT constraints derived by multiplying constraints (41) by the quality variables q_{jw}:

$$\begin{aligned}
\forall j \leq p \ \forall w \leq l \quad q_{jw}(\textstyle\sum_{k=1}^{r} x_{jk} - u_j) \ &= \\
\textstyle\sum_{k=1}^{r} q_{jw}x_{jk} - q_{jw}u_j \ &= \\
\textstyle\sum_{j=1}^{r} z_{jwk} - t_{jw} \ &= 0.
\end{aligned}$$

These constraints can be used to eliminate the bilinear constraints $t_{jw} = q_{jw}u_j$.

Another way to interpret what is happening in this particular case is that the constraint sets (37) and (39) (the bilinear constraints in the general blending problem formulation above) define more bilinear products than is really necessary: if we were to re-write the first term on the left hand side of (37) as $\sum_{k=1}^{r} q_{jw}x_{jk}$, we would not need to create all the variables t_{jw}. Yet another way of saying this is that distributing products over sums is advantageous. This is in accordance with the considerations found in (Tawarmalani and Sahinidis, 2002b), p. 73. However, the important point is that, here, this reformulation is determined automatically by a general-purpose algorithm.

Table II. Pooling and blending test problem statistics.

	n_{org}	m_{org}	n_{std}	m^l_{std}	$\|BT\|$	$\|RC\|$	$\|NB\|$
Haverly 1	9	8	18	11	6	2	0
Haverly 2	9	8	18	11	6	2	0
Haverly 3	9	8	18	11	6	2	0
Foulds 2	26	16	51	21	20	4	0
Foulds 3	168	48	313	57	136	8	0
Foulds 4	168	48	313	57	136	8	0
Foulds 5	100	40	173	45	68	4	0
Ben-Tal 4	10	8	19	11	6	2	0
Ben-Tal 5	41	27	94	32	48	8	0
Example 1	21	30	64	33	40	8	0
Example 2	25	42	88	45	60	12	0
Example 3	38	49	132	53	90	18	0
Example 4	26	35	77	38	48	8	0

Table II summarizes the main characteristics of the test problems. Here, n_{org}, m_{org} are respectively the number of variables and constraints in the original problem formulation, n_{std} is the number of variables in the standard form $[P]$, m^l_{std} is the number of *linear* constraints in the standard form, $|RC|$ is the number of RRLT constraints created, $|BT|$ is the number of bilinear constraints in the standard form before RRLT constraint creation and $|NB|$ is the number of new bilinear terms created during the RRLT constraint creation procedure. We note that $|NB| = 0$ in all test problems considered.

We now proceed to consider how the addition of RRLT constraints affects the tightness of the convex relaxation of the NLP in the context

of its solution using deterministic global optimization algorithms. Table III compares the number of nodes needed to solve the problems of Table II using nine different codes. The first six are computer codes described in the literature:

1 = (Foulds et al., 1992)
2 = (Visweswaran and Floudas, 1993)
3 = (Ben-Tal et al., 1994)
4 = (Visweswaran and Floudas, 1996)
5 = (Adhya et al., 1999)
6 = (Tawarmalani and Sahinidis, 2004)
7 = (Audet et al., 2004)
8 = (Sahinidis and Tawarmalani, 2005)

and their performance is shown in Table III, taken from the corresponding papers. Codes 9, 10 and 11 correspond to an implementation of sBB based on the algorithm proposed by (Smith and Pantelides, 1999). The difference between them is that:

- code 9 solves the original problem (no RRLT constraints)
- code 10 includes both RRLT constraints and all bilinear terms found in the original problem
- code 11 includes the RRLT constraints but eliminates the redundant bilinear terms.

In Table IV we have listed comparative results for the lower bounds to the objective functions obtained at the root node of each spatial branch-and-bound search carried out by codes 9,10 and 11 (in columns "LB 9", "LB 10", "LB 11"). Column "Opt" contains the globally optimal objective function values.

Columns 1-8 of Table III demonstrate the steady progress that has been achieved in spatial branch-and-bound algorithms over the last one and a half decade. The incorporation of the RRLT constraints in the convex relaxation improves solution performance dramatically (cf. columns 10 and 11 of Table III), allowing the solution of four problems that were not solvable after 10000 nodes by code 9. In general, a significant reduction in the number of nodes that need to be examined is observed; this is the effect of the tightening of the convex relaxation as is well illustrated by the comparison of the lower bounds between codes 9 and 10 (or 11) shown in Table IV.

Codes 10 and 11 perform very similarly when applied to most examples. In three cases (Foulds 2 and Examples 1 and 3), retaining the McCormick inequalities generated by the redundant bilinear constraints does in fact have a beneficial effect. However, in two other cases

Table III. Number of nodes required for global solution of pooling and blending problems. The '-' sign indicates that the algorithm did not converge within 10000 iterations.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Haverly 1	5	7	3	12	3	3	9	1	3	1	1
Haverly 2		19	3	12	9	9	13	1	5	3	3
Haverly 3			3	14	5	3	7	1	3	3	3
Foulds 2	9				1	1	1	0	23	9	15
Foulds 3	1				1	1	-	0	-	3	1
Foulds 4	25				1	1	-	0	-	2	2
Foulds 5	125				1	1	-	0	-	1	1
Ben-Tal 4		47	25	7	3	3	43	1	5	1	1
Ben-Tal 5		42	283	41	1	1	39	0	-	1	1
Example 1					6174	1869	245	28	65	3	25
Example 2					10743	2087	267	17	415	3	3
Example 3					79944	7369	537	31	37	3	243
Example 4					1980	157	693	1	31	31	5

Table IV. Objective function lower bounds at the root node of the sBB run for codes 9, 10 and 11, compared with globally optimal objective function value.

	LB 9	**LB 10**	**LB 11**	**Opt**
Haverly 1	-1200.1	-400.0	-400.0	-400.0
Haverly 2	-2216.4	-1000.0	-1000.0	-600.0
Haverly 3	-1550.0	-800.0	-800.0	-750.0
Foulds 2	-2145.6	-1133.3	-1133.3	-1100.0
Foulds 3	-93.0	-8.0	-8.0	-8.0
Foulds 4	-96.9	-8.0	-8.0	-8.0
Foulds 5	-100.0	-8.0	-8.0	-8.0
Ben-Tal 4	-950.0	-450.0	-450.0	-450.0
Ben-Tal 5	-6100.0	-3500.0	-3500.0	-3500.0
Example 1	-577.1	-572.1	-572.4	-549.8
Example 2	-577.1	-572.1	-572.4	-549.8
Example 3	-574.4	-570.9	-571.1	-561.0
Example 4	-1068.1	-1029.0	-1029.0	-877.6

(Foulds 2 and Example 4), the opposite is the case; for the problem considered here, more detailed examination has revealed that this is due to the fact that the upper bounding problem has fewer bilinear terms, and hence the local NLP solver used for the solution of this non-convex problem is more effective. A comparison of the objective function lower bound at the root node between 10 and 11 (see Table IV) also shows very little advantage in favour of the (potentially) tighter formulation 10.

Overall, the results indicate that the incorporation of the RRLT constraints is vastly beneficial. Retaining or discarding the redundant bilinear terms has only a second-order effect on the code performance.

We wish to emphasize that the important conclusion that can be drawn from Table III is not that codes 10 and 11 are somehow "better" than those presented in earlier literature. The main comparison of interest is between codes 9 and 10, 11 which are entirely identical with the exception of the RRLT constraints that are automatically generated by code 10, and the elimination of redundant bilinear terms in code 11. This provides another demonstration of the fact that, as in the case of the branch-and-bound solution of mixed integer optimization problems, problem (re-)formulation is often much more important than algorithmic details. Nevertheless, automatic reformulation and basic algorithmic improvements are largely complementary, and need to be applied together in practical codes.

7. Generalization of the graph-theoretical algorithm

The graph-theoretical algorithm described in Section 4 for the identification of useful RRLT constraints has some limitations. These arise from the fact that potential multiplications are considered separately for each variable, which may result in some useful multiplications being missed. To understand this point, consider the following very simple example:

$$\left.\begin{array}{lll} \min & x_1^2 + x_2^2 & \\ & x_1 + x_2 & = 1 \\ & 0 \leq x_1, x_2 \leq 10. & \end{array}\right\} \tag{42}$$

On multiplying the linear constraint $x_1 + x_2 = 1$ by x_1 we obtain $x_1^2 + x_1 x_2 = x_1$, which introduces one new bilinear term $x_1 x_2$; thus, this multiplication would not appear to bring any benefit. Similarly, on multiplying the same linear constraint by x_2, we would get $x_1 x_2 + x_2^2 = x_2$ and thus, again, one new bilinear term $x_1 x_2$; hence this multiplication

would not be considered to be beneficial either. Consequently, the algorithm of Section 4 applied to this system will not create any RRLT constraint.

However, considering the combined effect of the two multiplications, we note that they produce two new linearly independent RRLT constraints while introducing only one new bilinear term (namely, x_1x_2). These two RRLT constraints can be used to eliminate two bilinear terms from the problem, e.g. x_1^2 and x_2^2, leading to the problem reformulation:

$$\begin{array}{rrll} \min & w_1 + w_3 & & \\ & x_1 + x_2 & = 1 & \\ & w_1 + w_2 & = x_1 & \\ & w_2 + w_3 & = x_2 & \\ & w_2 & = x_1x_2 & \\ & 0 \leq x_1, x_2 & \leq 10 & \\ & 0 \leq w_i & \leq 100, & i = 1, 2, 3. \end{array}$$

Essentially, the algorithm of Section 4 correctly identifies that multiplying the linear constraint by either x_1 or x_2 results in a bilinear term x_1x_2 that did not occur in the original problem. What it misses is the fact that it is the *same* bilinear term in both cases. This is unfortunate as such reformulation may be very beneficial. For instance, we have found that the numerical solution of the example above in its original form with a simple sBB algorithm requires the examination of 96 nodes, while the reformulated one can be solved in a single node.

This motivates an extension to the algorithm of Section 4 to consider simultaneously the multiplication of linear constraints by each and every system variable, while still aiming to identify a minimal set of useful multiplications. Instead of creating one bipartite graph for each multiplier variable, we consider one unified bipartite graph comprising two disjoint sets of nodes:

- The ρ-nodes which comprize $m \times n$ nodes ρ_{ij} representing the potential multiplication of constraint i by variable x_j.
- The σ-nodes which comprize n^2 nodes σ_{hk} representing the bilinear terms x_hx_k.

An edge connecting node ρ_{ij} and σ_{hk} exists if the multiplication of constraint i by variable x_j would give rise to a new bilinear term x_hx_k; obviously either $h = j$ or $k = j$ holds.

Having created the bipartite graph, we attempt to trace an augmenting path emanating from each node ρ_{ij}. If no such path is found,

then we must have identified a dilation involving ν nodes of type ρ and $\nu - 1$ nodes of type σ. This implies that the variable-constraint multiplications corresponding to these ρ nodes will result in ν new constraints but only $\nu - 1$ new bilinear terms – which is exactly what we are trying to establish.

We apply this generalized algorithm to the simple example problem (42). Assuming that the constraint $x_1 + x_2 = 1$ is labelled with index $i = 1$, the unified bipartite graph is shown in Figure 6. Tracing an augmenting path from node ρ_{11} is succesful, resulting in node σ_{12} being assigned to it. However, no augmenting path emanating from node ρ_{12} can be found. Instead, we identify a dilation comprising nodes ρ_{11}, σ_{12} and ρ_{12}, i.e. in this case, $\nu = 2$. This simply implies that the reformulated problem should involve multiples of the linear constraint by both variables, which introduces a single bilinear term.

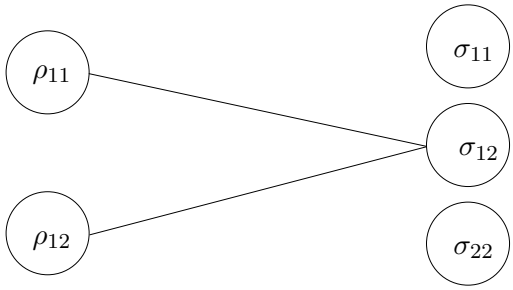

Figure 6. Unified bipartite graph for problem (42) in the generalized algorithm.

As has already been mentioned, the worst-case computational complexity of the dilation-finding algorithm of Section 4.3 is proportional to the product of the number of nodes from which the augmenting paths emanate and the number of edges; on the other hand, the average-case complexity is nearer to the sum of these two numbers. The original procedure described in Section 4 was applied to a bipartite graph with m linear constraint nodes; in principle, each of these nodes could be connected with each and every one of the n variable nodes; consequently, the number of edges is bounded from above by mn. Therefore, the worst-case and average complexities of this procedure are $O(m^2n)$ and $O(m + mn)$ respectively. Of course, the procedure has to be applied separately for each candidate multiplier variable; therefore, the corresponding total complexities are $O(m^2n^2)$ and $O(mn + mn^2)$ respectively. On the other hand, the procedure described in this section is applied to a bipartite graph with mn ρ-nodes and n^2 σ-nodes. Each ρ-node is potentially connected with up to n σ-nodes, and therefore the number of edges is bounded from above by mn^2 edges. Conse-

quently, the worst-case and average complexities are $O(m^2n^3)$ and $O(mn + mn^2)$ respectively. In conclusion, the procedure of this section has worse worst-case complexity than that of Section 4, but similar average complexity.

On the other hand, the memory requirements of the two procedures are very different. With the original algorithm, a graph consisting of $m+n$ nodes and up to mn vertices needs to be stored in memory at any one time, whereas the unified bipartite graph of this section will have $mn + n^2$ nodes and up to mn^2 edges. For extremely large problems, these requirements may be excessive and special attention may have to be paid to the implementation of the algorithm (one could create the graph "on-the-fly" as the algorithm progresses, deriving the nodes and edges from the problem data at each step).

8. Concluding remarks

The work presented in this paper is based on the fact that geometrical intersections of hyperplanes and nonlinear hypersurfaces corresponding to bilinearities may embed a higher degree of linearity than what is apparent by mere inspection of the defining equations. We have shown that it is possible to exploit this fact so as to reformulate an NLP involving such equality constraints to a form with fewer bilinearities and more linear constraints.

The basic idea of the reformulation is to multiply subsets of the NLP's linear constraints by one of the system variables x. This creates new linear constraints expressed in terms of variables w, each one of which corresponds to a bilinear product, viz. $w_{jk} \equiv x_j x_k$. In general, some of these w variables will already exist in the original NLP while others are new variables introduced by the multiplication.

Unlike the original RLT procedures proposed in (Sherali and Alameddine, 1992; Sherali, 2002), our algorithm is limited to linear equality constraints. However, it is more selective in that it identifies multiplications which result in "valid" sets of RRLT constraints, i.e. sets in which the number of constraints exceeds the number of new bilinear terms introduced by the multiplication. Moreover, this operation is applied once only as a pre-processing step of the original non-convex NLP, relying on the use of graph theoretical algorithms for the identification of valid RRLT constraint sets. These algorithms have the advantage of being quite fast even when applied to relatively large systems. Also, they may sometimes provide useful insight on how best to formulate broad classes of problems (such as pooling/blending) since their results depend only on the constraint structure and not on the specific nu-

merical values of the various coefficients that appear in any particular problem instance. On the other hand, they may fail to identify some valid RRLT constraint sets in cases in which the matrix A'' in Equation (28) is numerically singular but structurally non-singular.

Acknowledgement

The authors wish to acknowledge the invaluable help of Dr. Xueya Zhang of the Centre of Process Systems Engineering, Imperial College London, in obtaining the computational results presented in Table III using a modified version of the algorithm in (Smith and Pantelides, 1999). We are also grateful for the financial support of the United Kingdom's Engineering and Physical Sciences Research Council (EPSRC) under Platform Grant GR/N08636.

References

Adams, W., J. Lassiter, and H. Sherali: 1998, 'Persistency in 0-1 Polynomial Programming'. *Mathematics of Operations Research* **2**(23), 359–389.

Adhya, N., M. Tawarmalani, and N. Sahinidis: 1999, 'A Lagrangian Approach to the Pooling Problem'. *Industrial and Engineering Chemistry Research* **38**, 1956–1972.

Al-Khayyal, F. and J. Falk: 1983, 'Jointly constrained biconvex programming'. *Mathematics of Operations Research* **8**, 273–286.

Audet, C., J. Brimberg, P. Hansen, S. Le Digabel, and N. Mladenović: 2004, 'Pooling Problem: Alternate Formulations and Solution Methods'. *Management Science* **50**(6), 761-776.

Ben-Tal, A., G. Eiger, and V. Gershovitz: 1994, 'Global Minimization by Reducing the Duality Gap'. *Mathematical Programming* **63**, 193–212.

Duff, I.: 1981, 'On algorithms for obtaining a maximum transversal'. *ACM Trans. Math. Software* **7**, 315–330.

Foulds, L., D. Haughland, and K. Jornsten: 1992, 'A Bilinear Approach to the Pooling Problem'. *Optimization* **24**, 165–180.

Goodaire, E. and M. Parmenter: 1998, *Discrete Mathematics with Graph Theory*. London: Prentice-Hall.

Harary, F.: 1971, *Graph Theory*. Reading, MA: Addison-Wesley, second edition.

Hopcroft, J.E. and Karp, R.M.: 1973, 'An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs'. *SIAM Journal on Computing* **2**, 225–231.

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

Liberti, L. and C. Pantelides: 2003, 'Convex Envelopes of Monomials of Odd Degree'. *Journal of Global Optimization* **25**, 157–168.

McCormick, G.: 1976, 'Computability of Global Solutions to Factorable Nonconvex Programs: Part I — Convex Underestimating Problems'. *Mathematical Programming* **10**, 146–175.

Pantelides, C.: 1988, 'The Consistent Initialization of Differential-Algebraic Systems'. *SIAM J. Sci. Stat. Comput.* **9**, 213–231.

Pardalos, P. and H. Romeijn (eds.): 2002, *Handbook of Global Optimization*, Vol. 2. Dordrecht: Kluwer Academic Publishers.

Quesada, I. and I. Grossmann: 1995, 'Global optimization of bilinear process networks and multicomponent flows'. *Computers & Chemical Engineering* **19**(12), 1219–1242.

Sahinidis, N. and M. Tawarmalani: 2005, 'Accelerating Branch-and-Bound through a Modeling Language Construct for Relaxation-Specific Constraints'. *Journal of Global Optimization*, **32**(2), 259-280.

Sherali, H.: 2002, 'Tight Relaxations for Nonconvex Optimization Problems using the Reformulation-Linearization/Convexification Technique (RLT)'. In (Pardalos and Romeijn, 2002), pp. 1–63.

Sherali, H. and W. Adams: 1986, 'A Tight Linearization and an Algorithm for 0-1 Quadratic Programming Problems'. *Management Science* **32**(10), 1274–1290.

Sherali, H. and W. Adams: 1999, *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems*. Dordrecht: Kluwer Academic Publishers.

Sherali, H. and A. Alameddine: 1992, 'A New Reformulation-Linearization Technique for Bilinear Programming Problems'. *Journal of Global Optimization* **2**, 379–410.

Sherali, H. and C. Tuncbilek: 1997, 'New reformulation linearization/convexification relaxations for univariate and multivariate polynomial programming problems'. *Operations Research Letters* **21**, 1–9.

Smith, E.: 1996, 'On the Optimal Design of Continuous Processes'. Ph.D. thesis, Imperial College of Science, Technology and Medicine, University of London.

Smith, E. and C. Pantelides: 1999, 'A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs'. *Computers & Chemical Engineering* **23**, 457–478.

Tawarmalani, M. and N. Sahinidis: 2002a, *Convexification and Global Optimization in continuous and mixed-integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*. Dordrecht: Kluwer Academic Publishers.

Tawarmalani, M. and N. Sahinidis: 2002b, 'Exact Algorithms for Global Optimization of Mixed-Integer Nonlinear Programs'. In (Pardalos and Romeijn, 2002), pp. 1–63.

Tawarmalani, M. and N. Sahinidis: 2004, 'Global Optimization of Mixed Integer Nonlinear Programs: A Theoretical and Computational Study'. *Mathematical Programming* **99**, 563–591.

Tuy, H.: 1998, *Convex Analysis and Global Optimization*. Dordrecht: Kluwer Academic Publishers.

Visweswaran, V. and C. Floudas: 1993, 'New Properties and Computational Improvement of the GOP Algorithm for Problems with Quadratic Objective Functions and Constraints'. *Journal of Global Optimization* **3**, 439–462.

Visweswaran, V. and C. Floudas: 1996, 'New Formulations and Branching Strategies for the GOP Algorithm'. In: I. Grossmann (ed.): *Global Optimization in Engineering Design*. Dordrecht, Kluwer Academic Publishers.