Bounded Global Optimization for Polynomial Programming using Binary Reformulation and Linearization

Joseph W. Norman
University of Michigan
jwnorman@umich.edu

May, 2012

ABSTRACT
This paper describes an approximate method for global optimization of polynomial programming problems with bounded variables. The method uses a reformulation and linearization technique to transform the original polynomial optimization problem into a pair of mixed binary-linear programs. The solutions to these two integer-linear reformulations provide upper and lower bounds on the global solution to the original polynomial program. The tightness of these bounds, the error in approximating each polynomial expression, and the number of constraints that must be added in the process of reformulation all depend on the error tolerance specified by the user for each variable in the original polynomial program. As these error tolerances approach zero the size of the reformulated programs increases and the calculated interval bounds converge to the true global solution.

1 INTRODUCTION
This paper describes a method for approximating with interval bounds the global optimum of a polynomial program with bounded real variables and no special convexity properties. Such problems arise in many contexts including the analysis of parametric probability models [8, 9]. For that application and many others it is sufficient to generate interval bounds on the global minimum and maximum solutions of the polynomial optimization problems involved; it is not required to compute those solutions with infinite precision. The successive approximation method presented here uses a reformulation and linearization technique to create a hierarchy of pairs of mixed binary linear programs whose solutions bound the true global optimum of the original polynomial problem.

The method allows the user to adjust directly and incrementally how many variables and constraints are added in the reformulation stage, and to compute hard bounds on the error in approximating each polynomial before solving the reformulated problem; thus the user can control computation to trade the tightness of the generated interval bounds against the time required to compute them. The reformulated problems can be solved by well-developed branch-and-bound or branch-and-cut methods for mixed integer linear programming for which codes are available in commercial and free optimization software. The general global polynomial optimization problem is NP-hard and it remains an open question how well the bounds provided by the proposed method converge in practice and how best to control computation to produce the most useful answers with the smallest amount of computation.

The main contributions of this method are control of computation and definitive interval-based answers with predictable and explicit bounds on potential errors. Based on the parameters provided by the user, the method computes bounds on the possible error in the linear approximation of the objective and each constraint, before the reformulated optimization problem is solved. This predictability of error bounds means that the method can generate hard interval bounds on the global solution to each polynomial optimization problem, conditioned on the feasibility of that problem. Feasibility can sometimes be confirmed or refuted by
solving the reformulated problems; and even when feasibility cannot be established definitively, the method offers useful linear bounds on the set of points that are potentially feasible.

1.1 Preliminaries

Let us consider a polynomial program PP:

\[
\begin{align*}
\text{minimize} & \quad f(x_1, \ldots, x_n) \\
\text{subject to} & \quad g_1(x_1, \ldots, x_n) \leq 0, \; g_2(x_1, \ldots, x_n) \leq 0, \ldots, \; g_q(x_1, \ldots, x_n) \leq 0 \\
\text{and} & \quad \alpha_1 \leq x_1 \leq \beta_1, \; \alpha_2 \leq x_2 \leq \beta_2, \ldots, \; \alpha_n \leq x_n \leq \beta_n
\end{align*}
\]  

(1)

in which the objective \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) is a polynomial function of the variables \( x_1, \ldots, x_n \) as is each constraint \( g_j \). The lower bound on each real variable \( x_i \) is \( \alpha_i \) and the upper bound is \( \beta_i \). Each variable \( x_i \) may take positive, negative, or zero values. Fractional objectives can be accommodated using the Charnes-Cooper transformation [1]. It will be demonstrated how to generate from PP a pair of mixed integer-linear programs such that the global minimum of PP is bounded by the solution to each linear approximation.

Let us use \( x = (x_1, \ldots, x_n) \) to represent the set of values of \( x \) that satisfy the bound constraints included in the polynomial program PP:

\[ \Phi_x = \{ (x_1, \ldots, x_n) : \; \alpha_1 \leq x_1 \leq \beta_1, \ldots, \; \alpha_n \leq x_n \leq \beta_n \} \]  

(2)

The bound constraints \( \Phi \) and the polynomial constraints \( g_j \geq 0 \) define a semialgebraic set of feasible values for \( x \). Thus we can rewrite the polynomial optimization problem PP as follows:

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad g_1(x) \leq 0, \; g_2(x) \leq 0, \ldots, \; g_q(x) \leq 0 \\
\text{and} & \quad x \in \Phi_x
\end{align*}
\]  

(3)

Let us use \( f_{PP}(x^*) \) to denote the true global minimum solution to the problem PP, where \( x^* \) identifies a feasible (though not necessarily unique) point at which that minimum occurs. The goal is to develop a pair of mixed integer-linear programs \( LP^- \) and \( LP^+ \) using new variables \( w \) with the property that their solutions bound the true global minimum:

\[ f_{LP^-}(w^*) \leq f_{PP}(x^*) \leq f_{LP^+}(w^*) \]  

(4)

It is also desirable for the linear approximations to provide trustworthy information about the feasibility of the original nonlinear program, reporting whether feasibility has been confirmed, refuted, or not yet determined. The reformulation and linearization method presented below accomplishes these goals.

The tuple \( w \) of variables used in the reformulated programs contains three classes of variables: ‘unit variables’ denoted \( u_1, u_2, \ldots, u_\phi \) of which each binary variable \( u_i \in \{0, 1\} \) is allowed to take the integer value zero or one; ‘remainder variables’ denoted \( r_1, r_2, \ldots, r_n \) of which each real variable \( 0 \leq r_i \leq 1 \); and ‘unit-product variables’ denoted \( y_1, y_2, \ldots, y_\psi \) of which each real variable \( 0 \leq y_i \leq 1 \). The number of unit variables is \( \phi \) and the number of unit-product variables \( \psi \). Therefore the total number of variables in \( w \) used in reformulated programs is \( n + \phi + \psi \), where \( n \) is the number of variables in the original program PP. Thus the reformulated variables \( w \) are given by:

\[ w = (u_1, u_2, \ldots, u_\phi; r_1, r_2, \ldots, r_n; y_1, y_2, \ldots, y_\psi) \]  

(5)

As part of the reformulation process some number \( \rho \) of linear constraints are constructed and added to the problem; these constraints define a feasible set \( \Phi_w \) of values for the reformulated variables in \( w \). Thus the problem is lifted from the space \( \mathbb{R}^n \) to the space \( \mathbb{R}^{n+\phi+\psi} \) and linearized in that space. Analogous to the
bounding box $\Phi_x$ that limits the feasible values of $x \in \mathbb{R}^n$ a new bounding box $\Phi_w$ limits the feasible values of $w \in \mathbb{R}^{n+\phi+\psi}$.

In order to control the linearization and reformulation process the user supplies parameters $\sigma_1, \sigma_2, \ldots, \sigma_n$ where each $\sigma_i$ determines how the corresponding original variable $x_i$ is to be approximated. The tightness of the interval bounds computed on the global solution to PP, as well as the number $\phi$ of unit variables, the number $\psi$ of unit-product variables, and the number $\rho$ of constraints that must be used in the reformulation process, all depend on these user-supplied parameters.

2 PRIOR WORK

2.1 Reformulation and Linearization

The method presented here is based on the principles set out by Li and Chang [7], which in turn reference the reformulation linearization technique of Sherali and Tuncbilek [11]; it is also a progression of the linearization and bounding techniques described in the author’s doctoral dissertation [8]. The main extension relative to the work of Li and Chang is that here there is a defined relationship between the solution to each linear reformulation and the true solution to the original polynomial program: the minimum solution of one linear program is less than or equal to the true global minimum, which is in turn less than or equal to the solution of the other linear program. The original work presented in [7] offers the exact solution to a problem related to the original polynomial problem, but with the exact qualifications on that relationship left unspecified. These desired solution properties are produced by explicit calculation of the possible error in approximating each polynomial expression through linearization combined with reasoning about the inequalities involved. Additionally, in this work a different method is used to linearize products of two or more variables. Finally, the number of variables and constraints necessary to reformulate an optimization problem by this method is calculated. As in the original work no assumptions are made about the convexity of the feasible set or objective function in the polynomial problem; hence the technique presented here works for maximization as well as minimization of objectives. For simplicity of presentation only minimization problems are described below.

The reformulation-linearization technique (RLT) described by Sherali and Tuncbilek [11] presents an approach to solving general polynomial programming problems which is quite similar. In each case the problem is lifted from its original $n$-dimensional space $\mathbb{R}^n$ to a higher-dimensional space via the addition of new variables, and linear constraints are imposed in that higher-dimensional space to provide approximations; the original RLT provides only outer bounds (e.g. upper bounds on the global maximum), whereas the present work provides inner bounds as well. Mechanically, the present work uses two kinds of linearization constraints (let us call them real-product-mean and binary-product-sum) which are different from the bound-factor, constraint-factor, and convex-variable-bounding constraints used in the traditional RLT. In Sherali and Tuncbilek’s work the new variables in the lifted space map one-to-one to the monomial terms that comprise the basis of the original polynomial space; thus the number of new variables is fixed. Depending on which classes of linearization constraints are added, the number of additional constraints varies; the user may choose to add whole classes of constraints or select one at a time in an ad-hoc way. However in the present work the number of additional variables can vary as well as the number of additional constraints. Furthermore, these are both determined in a strict algorithmic way according to a small set of parameters provided by the user.

One benefit of the approach taken here is that the user can make a sequence of reformulations at different granularities or resolutions. Moreover, the reformulation and subsequent linearization are separated from the branch-and-bound search used to solve the resulting linear programs (which are mixed integer linear programs in my case); that is, the branch/bound requirements are encoded in integer variables. This allows the current method to take advantage of well-developed codes for solving mixed integer linear programs.
This also allows the user to control the size of each reformulated program and thus the time required to solve it. We shall present in more detail the relation to the traditional reformulation linearization technique.

2.2 Semidefinite Programming and Real Algebraic Geometry

Besides the linear relaxations of the RLT, another option is to use semidefinite program relaxations of the original polynomial program, to compute a lower bound on the true minimum. As described by Lasserre [5, 6] this approach is in fact a generalization of the linear relaxation approach and is related to it through the theory of moments and its dual theory of representation of polynomials which have positive values over a semialgebraic set. Additionally, Floudas and Visweswaran [2, 3] present a technique to transform bilinear, quadratic, or polynomial programs to a new problem with partitioned variables and a certain convexity property; the transformed problem can be solved by a primal and relaxed dual approach.

There are many methods to solve special cases of the polynomial programming problem, e.g. those in which all polynomial functions have positive values or other instances in which favorable convexity conditions are met. For the broader class of constrained global optimization problems there are also a variety of methods, including outer approximation and branch-and-bound techniques, as reviewed by Horst and Tuy [4]. In the classification of Horst and Tuy, the present method is a successive approximation method. It is deterministic rather than stochastic. The inner or outer bounds at each stage are computed by relaxation of the original polynomial program to a mixed binary linear program. Branching and bounding can be carried out to solve each of these linear programs; however the branching and bounding occurs in the (binary part of) lifted space not the original space of the polynomial problem. The current method uses only the primal form of the polynomial program and does not incorporate the dual problem in its analysis.

Most successive approximation methods reformulate the problem in one step into a convex (linear or semidefinite) form; here we use a nonconvex intermediate (a mixed 0-1 integer linear program) which is NP-hard to solve but for which a great deal of work has been done to make algorithms that perform well on average. Note that successive approximation with semidefinite relaxations is one of the possible approaches to solving mixed binary linear programs which is distinct from the more common branch-and-bound or branch-and-cut methods.

It is not clear to me if one can recover from each SDP relaxation any bound on the error in the approximation (perhaps using the primal and dual solutions provides appropriate bounds on the global minimum), or if it is possible to determine error bounds on the approximation before solving it; it seems instead that after solving a particular relaxation it may be verified that it is indeed a global solution.

3 BINARY REFORMULATION AND LINEARIZATION

Here we develop the method to create a pair of mixed binary linear programs to bound the global solution to a polynomial program.

3.1 Binary Expansion and Basic Inequalities

**Proposition 1 (Reformulation with Unit Variables)** Consider a real-valued variable $x_i$ bounded by the constants $\alpha_i \leq x_i \leq \beta_i$. Given a positive constant $\kappa_i$ satisfying $0 < \kappa_i \leq \beta_i - \alpha_i$ it is possible to represent $x_i$ as a sum that involves some number $\sigma_i \geq 0$ of binary variables $u_{i,j} \in \{0,1\}$ and a nonnegative real variable $r_i \in [0,1]$:

$$x_i = \alpha_i + \kappa_i \sum_{j=1}^{\sigma_i} 2^{j-1} u_{i,j} + \kappa_i r_i$$

(6)
Let us call each \( u_{i,j} \) a unit variable and \( r_i \) the remainder variable; each \( \kappa_i \) is the error limit on the corresponding variable \( x_i \). In general it is necessary to add the constraint that the sum satisfies the original upper bound \( \beta_i \):

\[
\alpha_i + \kappa_i \sum_{j=1}^{\sigma_i} 2^{j-1} u_{i,j} + \kappa_i r_i \leq \beta_i
\]  

(7)

Otherwise it may be possible to choose values of \( \kappa_i, u_{i,j} \text{ and } r_i \) that would produce a value greater than \( \beta \).

This first proposition is a restatement of Equation 2.1 in [7], with additional detail provided here. The number \( \sigma_i \) of unit variables required to reformulate a variable \( x_i \) in this way is related to the error limit \( \kappa_i \):

\[
\sigma_i = \left\lceil \log_2 \left( \frac{\beta_i - \alpha_i}{\kappa_i} + \delta_i \right) \right\rceil
\]  

(8)

where the brackets indicate rounding up and \( \delta_i = 1 \) if \( x_i \) is discrete or 0 if \( x_i \) is continuous. The lowest possible error limit \( \kappa_i^* \) for a given number \( \sigma_i \) of unit variables is given by:

\[
\kappa_i^* = \frac{\beta_i - \alpha_i}{2^{\sigma_i} - \delta_i}
\]  

(9)

where again \( \delta_i = 1 \) if \( x_i \) is discrete. For each variable \( x_i \) the user may specify the number \( \sigma_i \) of unit variables and have the system compute the corresponding error limit \( \kappa_i^* \) using Equation [8] or the user may specify the desired error limit \( \kappa_i \) and have the system compute the number of unit variables required by Equation [8].

Note that in the case that \( \sigma_i = 0 \) the error limit \( \kappa_i = \beta_i - \alpha_i \) and no unit variables will be added; note also that a discrete variable with \( \sigma_i = 0 \) must have \( \alpha_i = \beta_i \) in which case \( \kappa_i \) is not needed for reformulation. Furthermore the error limit \( \kappa_i \) can be zero only for a fixed variable \( x_i \) with bounds \( \alpha_i = \beta_i \).

Also, it turns out that it is not necessary to add any unit variables for a variable \( x_i \) which appears only linearly in PP, in this case we will take \( \sigma_i = 0 \) but \( \kappa_i = \beta_i - \alpha_i \) (thus the reformulation of such an \( x_i \) is \( \kappa_i r_i \) which is equal to \( (\beta_i - \alpha_i)r_i \) with \( 0 \leq r_i \leq 1 \)). Finally, we can model a discrete variable with steps \( \kappa_i \) using the substitution above but omitting the remainder variable \( r_i \); such a variable would take values \( \{\alpha_i, \alpha_i + \kappa_i, \alpha_i + 2\kappa_i, \ldots, \alpha_i + \sigma_i \kappa_i\} \) with \( \alpha_i + \sigma_i \kappa_i \leq \beta_i \).

**Example 1** Consider the variables \( x = (x_1, x_2, x_3) \) bounded by \( 2 \leq x_1 \leq 5 \), \( 0 \leq x_2 \leq 10 \), and \( 4 \leq x_3 \leq 8 \). Let us use three unit variables to reformulate \( x_1 \) and two unit variables to reformulate each of \( x_2 \) and \( x_3 \). In other words \( \sigma_1 = 3 \), \( \sigma_2 = 2 \), and \( \sigma_3 = 2 \). According to Equation [9] the smallest possible error limits are then \( \kappa_1^* = 0.375 \), \( \kappa_2^* = 2.5 \), and \( \kappa_3^* = 1 \). The reformulated variables according to Proposition [1] are:

\[
\begin{align*}
x_1 &= 2 + 0.375u_{1,1} + 0.75u_{1,2} + 1.5u_{1,3} + 0.375r_1 \\
x_2 &= 2.5u_{2,1} + 5u_{2,2} + 2.5r_2 \\
x_3 &= 4 + u_{3,1} + 2u_{3,2} + r_3
\end{align*}
\]  

(10)

With the following constraints added by Equation [7]

\[
\begin{align*}
2 + 0.375u_{1,1} + 0.75u_{1,2} + 1.5u_{1,3} + 0.375r_1 & \leq 5 \\
2.5u_{2,1} + 5u_{2,2} + 2.5r_2 & \leq 10 \\
4 + u_{3,1} + 2u_{3,2} + r_3 & \leq 8
\end{align*}
\]  

(11)

**Proposition 2 (Linear bounds on real products)** Consider nonnegative real-valued variables \( r_1, r_2, \ldots, r_n \) each bounded by \( 0 \leq r_i \leq 1 \); assume \( n > 0 \). The difference between the mean and the product of these bounded variables is limited by the number of variables:

\[
0 \leq \left( \frac{r_1 + r_2 + \cdots + r_n}{n} \right) - (r_1 r_2 \cdots r_n) \leq \left( \frac{n-1}{n} \right)
\]  

(12)
which can be rewritten as:

\[ \begin{align*}
0 & \leq (r_1 + r_2 + \cdots + r_n) - n(r_1 r_2 \cdots r_n) \leq n - 1
\end{align*} \]  

(13)

This proposition is a generalization of Equation 2.5 in [7] with the following proof added here. It is clear that the relationships are true when every \( r_i \) is zero, in which case both the mean and product are zero; and when every \( r_i \) is one, in which case both the mean and product are one. For values other than these the greatest difference between the mean and the product will occur when one of the variables is zero and the others are one; in that case the mean will be \((n - 1)/n\) and the product will be zero, satisfying the equations above.

**Proposition 3 (Linear bounds on binary products)** Consider binary variables \( u_1, u_2, \ldots, u_n \) with each \( u_i \in \{0, 1\} \) and also one real-valued ‘remainder’ variable \( r \) that satisfies \( 0 \leq r \leq 1 \). The following relationships hold between the product of the binary variables, the remainder variable, and the individual binary variables:

\[ \begin{align*}
& u_1 u_2 \cdots u_n r \leq u_i, \ i = 1, \ldots, n & u_1 u_2 \cdots u_n r & \geq r + u_1 + u_2 + \cdots + u_n - n \\
& u_1 u_2 \cdots u_n r & \leq r
\end{align*} \]

(14)

If the real variable \( r \) is omitted the appropriate relationships between the sum and product of the unit variables are:

\[ \begin{align*}
& u_1 u_2 \cdots u_n \leq u_i, \ i = 1, \ldots, n & u_1 u_2 \cdots u_n & \geq 1 + u_1 + u_2 + \cdots + u_n - n \\
& u_1 u_2 \cdots u_n & \leq 1
\end{align*} \]

(15)

This third proposition is a restatement of Proposition 1 from [7] and the proof that appears there. It is clear that if any unit variable \( u_i = 0 \) then the product \( u_1 u_2 \cdots u_n = 0 \). In that case the sum \( u_1 + u_2 + \cdots + u_n \leq n - 1 \) as one of the terms is zero and each of the others is not greater than one. The constraints above are satisfied in this case. If every unit variable \( u_i = 1 \) then the sum \( u_1 + u_2 + \cdots + u_n = n \) and the product \( u_1 u_2 \cdots u_n = 1 \), in which case the constraints are also satisfied.

Let us introduce for this discussion a unit-product variable \( y = u_1 u_2 \cdots u_n r \) to represent the product of the binary variables \( u_1 \) through \( u_n \) and the real-valued variable \( r \) described above; we can use \( y = u_1 u_2 \cdots u_n \) for the product of the binary variables alone when no real variable \( r \) is included. Each unit-product variable \( y \) is continuous and bounded by \( 0 \leq y \leq 1 \).

### 3.2 Reformulation and Linearization of Polynomials

Now we have the tools to reformulate any polynomial expression from its native \( x \)-variables into a linear function of the unit variables \( u \), the unit-product variables \( y \), and the remainder variables \( r \). This procedure is a generalization of Propositions 2 and 3 from [7] which describe only the linearization of products of two or three variables.

#### 3.2.1 General Polynomial Form

Consider that any polynomial function \( g(x_1, x_2, \ldots, x_n) \) can be represented as the sum of several terms \( c_k m_k \) where each term is the product of a real coefficient \( c_k \) and a monomial \( m_k \), the latter of which is the product of several \( x \)-variables:

\[ g(x_1, x_2, \ldots, x_n) = c_1 m_1 + c_2 m_2 + \cdots + c_t m_t, \quad c_k \in \mathbb{R}, \quad m_k = \prod_{i \in I_k} x_i \]

(16)

*There is a typographical error in Equation 2.5 in [7]: the final term should be \( \frac{1}{4} \omega_1 \omega_2 \) instead of \( \frac{1}{4} \omega_1 \omega_2 \).
Let us adopt the convention that the first monomial $m_1 = 1$ so that the coefficient $c_1$ is the constant term in the polynomial $g$. Each tuple $I_k$ of indices identifies the original $x$-variables included in the product for the monomial $m_k$. An index $i$ may occur in $I_k$ more than once; if a tuple $I_k$ is empty then the corresponding product of zero variables is taken to be unity. Note that the size $|I_k|$ of the tuple of indices for a monomial $m_k$ is the degree of that monomial. Let us use $\Omega$ to represent the list of all monomials which occur in a particular optimization problem PP, and $t$ for the number of such monomials.

**Example 2** Consider the following polynomial program PP1 which is the example problem $PP(\Omega)$ presented in Sherali and Tuncbilek [11] and reproduced as Example 1 in Li and Chang [7]:

$$\begin{align*}
\text{minimize:} & \quad 5x_2 + x_3 + x_1^2 - 2x_1x_2 - 3x_1x_3 + 5x_2x_3 \\
& \quad - x_3^2 + x_1x_2x_3 \\
\text{subject to:} & \quad 4x_1 + 3x_2 + x_3 \leq 20 \\
& \quad x_1 + 2x_2 + x_3 \geq 1 \\
& \quad 2 \leq x_1 \leq 5 \\
& \quad 0 \leq x_2 \leq 10 \\
& \quad 4 \leq x_3 \leq 8
\end{align*}$$

The list of monomial terms (basis) used in PP1 is:

$$m = \{1, x_1, x_2, x_1 x_2, x_3, x_1 x_2 x_3, x_1^2, x_1 x_3, x_2 x_3, x_3^2\} \quad (17)$$

The first term $m_1 = 1$ has the empty tuple $I_1 = ()$ of indices and degree 0; the last term $m_{10} = x_3^2$ has the tuple $I_{10} = (3, 3)$ of indices and degree 2. The number of terms in $m$ is $t = 10$.

### 3.2.2 Reformulation of Monomials with Sums of Unit Variables

Now consider just one monomial $m_k = \prod_{i \in I_k} x_i$ of a polynomial $g$ as given in Equation (16). Using Proposition [11] we can substitute a sum involving several unit variables and a remainder variable for each original variable $x_i$ included in the product that defines this monomial $m_k$, where each $\alpha_i$ is the lower bound of the corresponding variable $x_i$, and each $\sigma_i$ is the number of unit variables necessary to represent $x_i$ within a tolerance of $\kappa_i$. Carrying out the multiplication to distribute the product over the sum yields an expression for the monomial $m_k$ of the form:

$$m_k = \prod_{i \in I_k} \left( \alpha_i + \kappa_i \sum_{j=1}^{\sigma_i} 2^{j-1} u_{i,j} + \kappa_i r_i \right) = z_{k,1} + z_{k,2} + \cdots + z_{k,s_k} \quad (18)$$

where each $z_{k,\ell}$ designates the $\ell$-th element in the sum that constitutes the distributed product. With reference to Equation (18) the number $s_k$ of such elements is limited by:

$$\prod_{i \in I_k} (\sigma_i + 1) \leq s_k \leq \prod_{i \in I_k} (\sigma_i + 2) \quad (19)$$

as the sum used to replace each variable $x_i$ according to Proposition [11] contains $\sigma_i$ unit variables, one remainder variable, and one constant lower bound $\alpha_i$ if that bound is nonzero.

Now, it is clear from the structure of the inner sum and outer product shown in Equation (18) that every element $z_{k,\ell}$ in the sum must be the product of some numbers of constant lower bounds $\alpha_i$, unit variables $u_i$, and remainder variables $r_i$:

$$z_{k,\ell} = (\alpha_{i_1} \alpha_{i_2} \cdots \alpha_{i_{s_k}}) \left( \kappa_{i_1} 2^{j-1} u_{i_1,j_1} \kappa_{i_2} 2^{j-1} u_{i_2,j_2} \cdots \kappa_{i_{s_k}} 2^{j-1} u_{i_{s_k},j_{s_k}} \right) \left( \kappa_{h_1} r_{h_1} \kappa_{h_2} r_{h_2} \cdots \kappa_{h_{s_{h}}} r_{h_{s_{h}}} \right) \quad (20)$$
where \( n_\alpha \) is the number of constants, \( n_u \) the number of unit variables, and \( n_r \) the number of remainder variables. Note with reference to Equation (13) that the number of each kind of element is limited by the degree \( d_k = |I_k| \) of the monomial \( m_k \):

\[
n_\alpha \leq d_k, \quad n_u \leq d_k, \quad n_r \leq d_k
\]  

(21)

Let us introduce the constant \( a_{k,t} \) to simplify notation:

\[
a_{k,t} = \frac{1}{n_r} \left( \alpha_{i_1} \alpha_{i_2} \cdots \alpha_{i_n} \right) \left( k_1 k_2 \cdots k_{n_u} 2^{j_1-1} 2^{j_2-1} \cdots 2^{j_{n_u}-1} \right) \left( k_{n_u+1} k_{n_u+2} \cdots k_{n_r} \right)
\]  

(22)

where we take the product of an empty set of items to be unity and where we omit division by \( n_r \) if \( n_r = 0 \). With this constant \( a_{k,t} \) we can rewrite the element \( z_{k,t} \) shown in Equation (20) as:

\[
z_{k,t} = a_{k,t} n_r \left( u_{i_1,j_1} u_{i_2,j_2} \cdots u_{i_{n_u},j_{n_u}} \right) \left( r_{h_1} r_{h_2} \cdots r_{h_{n_r}} \right)
\]  

(23)

where \( n_r \) is omitted if it is zero. Note that \( a_{k,t} \) may be positive or negative.

**Example 3** The ninth term in the list \( m \) of monomial terms used in the problem PP1 shown in Equation (17) is \( m_9 = x_2 x_3 \). Substituting the reformulation for each variable given in Equation (10) this term becomes:

\[
m_9 = (2.5u_{2,1} + 5u_{2,2} + 2.5r_2) \left( 4 + u_{3,1} + 2u_{3,2} + r_3 \right)
\]  

(24)

Carrying out the multiplication gives the following sum of 12 terms:

\[
m_9 = 10u_{2,1} + 2.5u_{2,2} u_{3,1} + 5u_{2,1} u_{3,2} + 2.5u_{2,1} r_3 + 20u_{2,2} + 5u_{2,2} u_{3,1} + 10u_{2,2} u_{3,2} + 5u_{2,2} r_3 + 2.5u_{3,1} r_2 + 5u_{3,2} r_2 + 2.5r_2 r_3
\]  

(25)

These terms are the elements \( z_{9,1} \) through \( z_{9,12} \) as described in Equation (18). Let us examine a few of these elements. For the fourth element \( z_{9,4} = 2.5u_{2,1} u_{3,1} \) we have: the number of unit variables \( n_u = 2 \), the number of remainder variables \( n_r = 0 \), and the constant \( a_{9,4} = 2.5 \). For the sixth element \( z_{9,6} = 2.5u_{2,1} r_3 \) we have: \( n_u = 1 \), \( n_r = 1 \), and \( a_{9,6} = 2.5 \). For the last element \( z_{9,12} = 2.5r_2 r_3 \) we have: \( n_u = 0 \), \( n_r = 2 \), and \( a_{9,12} = \frac{1}{2}(2.5) = 1.25 \).

### 3.2.3 Linearization of Products of Remainder and Unit Variables

We can use the propositions above to linearize each element \( z_{k,t} \) of the sum that constitutes the reformulated monomial term \( m_k \) in the following way. First let us introduce the notation \( \left[ z_{k,t} \right] \) with double square brackets to denote the linearized version of \( z_{k,t} \). Let us begin to compute this linearized form by replacing the product of remainder variables in \( z_{k,t} \) with their mean:

\[
\left[ z_{k,t} \right] = a_{k,t} n_r \left( u_{i_1,j_1} u_{i_2,j_2} \cdots u_{i_{n_u},j_{n_u}} \right) \left( r_{h_1} + r_{h_2} + \cdots + r_{h_{n_r}} \right)
\]  

(26)

Simplifying this expression yields:

\[
\left[ z_{k,t} \right] = a_{k,t} \left( u_{i_1,j_1} u_{i_2,j_2} \cdots u_{i_{n_u},j_{n_u}} \right) \left( r_{h_1} + r_{h_2} + \cdots + r_{h_{n_r}} \right)
\]  

(27)

Note that if the element \( z_{k,t} \) contains no remainder variables \( n_r = 0 \) the linearized form is unchanged from the original. Let us now replace the product of the unit variables \( u_{i_1,j_1} u_{i_2,j_2} \cdots u_{i_{n_u},j_{n_u}} \) and each remainder variable \( r_h \) in Equation (27) with a new unit-product variable \( y_{k,t,h} \) such that:

\[
y_{k,t,h} = \left( u_{i_1,j_1} u_{i_2,j_2} \cdots u_{i_{n_u},j_{n_u}} \right) r_h, \quad h = 1, 2, \ldots, n_r
\]  

(28)
The linearized element \([z_{k,\ell}]\) can now be written as a weighted sum of several unit-product variables:

\[
[z_{k,\ell}] = a_{k,\ell} (y_{k,\ell,1} + y_{k,\ell,2} + \cdots + y_{k,\ell,n_r}), \quad \text{if } n_r > 0
\]  

(29)

In the case that there are no remainder variables in the element \(z_{k,\ell}\) and thus \(n_r = 0\) let us introduce a single unit-product variable:

\[
y_{k,\ell,1} = (u_{i_1,j_1} u_{i_2,j_2} \cdots u_{i_{n_u},j_{n_u}})
\]  

(30)

and express the linearized element \([z_{k,\ell}]\) appropriately:

\[
[z_{k,\ell}] = a_{k,\ell} y_{k,\ell,1}, \quad \text{if } n_r = 0
\]  

(31)

To simplify notation let us introduce the variable \(n_{k,\ell}\) which equals \(n_r\) if \(n_r > 0\), or one otherwise. Thus each linearized element \([z_{k,\ell}]\) can be expressed:

\[
[z_{k,\ell}] = a_{k,\ell} \sum_{h=1}^{n_{k,\ell}} y_{k,\ell,h}, \quad n_{k,\ell} = \begin{cases} 1 & \text{if } n_r = 0 \\ n_r & \text{if } n_r > 0 \end{cases}
\]  

(32)

**Example 4** Let us return to the elements that comprise the expanded form of the monomial term \(m_9 = x_2 x_3\) described in Example 3 above. For the element \(z_{9,2} = 2.5 u_{2,1} u_{3,1}\), the linearized form \([z_{9,2}] = 2.5 y_{9,2,1}\) using the solitary unit-product variable \(y_{9,2,1}\) introduced according to Equation 30. For the element \(z_{9,4} = 2.5 u_{2,1} r_3\) the linearized form \([z_{9,4}] = 2.5 y_{9,4,1}\) using the unit-product variable \(y_{9,4,1} = u_{2,1} r_3\). For the element \(z_{9,12} = 2.5 r_2 r_3\) the constant \(a_{9,12} = 1.25\) according to Equation 22 and the linearized form \([z_{9,12}] = 1.25 (r_2 + r_3)\) according to Equation 27. According to Equation 28 we could introduce two trivial unit-product variables \(y_{9,12,1} = r_2\) and \(y_{9,12,2} = r_3\) and express the linearized term \([z_{9,12}] = 1.25 (y_{9,12,1} + y_{9,12,2})\) using the standard form given in Equation 32.

### 3.2.4 Linear Constraints on Unit-Product Variables

For each element \(z_{k,\ell}\) of the sum shown in Equation 18 one unit-product variable \(y_{k,\ell,h}\) has been introduced for each of the \(n_r\) remainder variables in Equation 27 (or a single product variable \(y_{k,\ell,1}\) if \(n_r = 0\)). In order to satisfy Proposition 3 it is necessary to add the following constraints for each unit-product variable \(y_{k,\ell,h}\) thus introduced:

\[
\begin{align*}
y_{k,\ell,h} &\leq u_{i_1,j_1} \\
y_{k,\ell,h} &\leq u_{i_2,j_2} \\
&\vdots \\
y_{k,\ell,h} &\leq u_{i_{n_u},j_{n_u}} \\
y_{k,\ell,h} &\geq r_h + u_{i_1,j_1} + u_{i_2,j_2} + \cdots + u_{i_{n_u},j_{n_u}} - n_u
\end{align*}
\]  

(33)

where \(r_h\) refers to the remainder variable associated with the unit-product variable \(y_{k,\ell,h}\) and with the substitution of unity in the place of \(r_h\) if \(n_r = 0\). The number of added constraints for each element \(z_{k,\ell}\) is \(n_r(n_u + 1)\). For a unit-product variable \(y_{k,\ell,h} = r_1\) which equals some remainder variable (i.e. \(n_r = 1\) and \(n + u = 0\)) or a unit-product variable \(y_{k,\ell,h} = u_{i,j}\) which equals some single unit variable (i.e. \(n_r = 0\) and \(n_u = 1\)), the constraints implied by Equation 33 are trivial and need not be added; it is sufficient to note the identities. In fact, in such cases the original variables can be used and new unit-product variables need not be introduced.

Let us use \(\Phi_w\) to designate the set of values of \(w\) that satisfy the linear constraints shown in Equation 33 for all unit-product variables introduced, as well as the upper bound constraints as shown in Equation 7 for all reformulated original variables. Let us use \(\rho\) to designate the number constraints used to specify the set \(\Phi_w\).
Example 5 Let us use the elements discussed in Examples 3 and 4 to illustrate the constraints on unit-product variables. The complete linearization of the reformulated monomial term $m_9$ is given by:

$$[m_9] = 10u_{2,1} + 20u_{2,2} + 11.25r_2 + 1.25r_3 + 2.5y_{9,2,1} + 5y_{9,3,1} + 2.5y_{9,4,1} + 5y_{9,6,1} + 10y_{9,7,1} + 5y_{9,8,1} + 2.5y_{9,10,1} + 5y_{9,11,1}$$  (34)

Compare this with the reformulated but not yet linearized term shown in Equation 25; note that the corresponding terms appear in different orders in the two equations. The linearized form above uses the following non-trivial unit-product variables:

$$
\begin{align*}
y_{9,2,1} &= u_{2,1}u_{3,1} \\
y_{9,3,1} &= u_{2,1}u_{3,2} \\
y_{9,4,1} &= u_{2,1}r_3 \\
y_{9,6,1} &= u_{2,2}u_{3,1} \\
y_{9,7,1} &= u_{2,2}u_{3,2} \\
y_{9,8,1} &= u_{2,2}r_3 \\
y_{9,10,1} &= u_{3,1}r_2 \\
y_{9,11,1} &= u_{3,2}r_2 
\end{align*}
$$  (35)

These unit-product variables are accompanied by the constraints described in Equation 33. For example, for the unit-product variable $y_{9,2,1} = u_{2,1}u_{3,1}$, Equation 33 implies the following constraints:

$$
\begin{align*}
u_{2,1} &\geq y_{9,2,1} \\
u_{3,1} &\geq y_{9,2,1} \\
u_{2,1} + u_{3,1} &\leq 1 + y_{9,2,1}
\end{align*}
$$  (36)

Similarly for the unit-product variable $y_{9,4,1} = u_{2,1}r_3$ Equation 33 implies:

$$
\begin{align*}
u_{2,1} &\geq y_{9,4,1} \\
u_{2,1} + r_3 &\leq 1 + y_{9,4,1} \\
r_3 &\geq y_{9,4,1}
\end{align*}
$$  (37)

For $z_{9,12}$ the constraints from Equation 33 on the unit-product variables $y_{9,12,1} = r_2$ and $y_{9,12,2} = r_3$ are trivial and need not be added.

3.2.5 Reformulation and Linearization of Entire Polynomials

Recall from Equation 18 that a polynomial term $m_k$ representing the product of several variables $x_i$ can be represented as the sum of several elements $z_{k,\ell}$

$$m_k = z_{k,1} + z_{k,2} + \cdots + z_{k,s_k}$$  (38)

where each element $z_{k,\ell}$ is the product of some number of constants, unit variables, and remainder variables as shown in Equation 20. Using the reformulation technique above a linear approximation $[m_k]$ can be generated for each element $z_{k,\ell}$ in this sum; adding these approximations yields a linear approximation $[m_k]$ for the polynomial term $m_k$:

$$[m_k] = [z_{k,1}] + [z_{k,2}] + \cdots + [z_{k,s_k}]$$  (39)

From the above and Equation 32 it is clear that the linear approximation for each monomial term $m_k$ is a weighted sum of unit-product variables $y_{k,\ell,h}$:

$$[m_k] = \sum_{\ell=1}^{s_k} \sum_{h=1}^{n_{k,\ell}} a_{k,\ell}y_{k,\ell,h}$$  (40)
where \( s_k \) is the number of elements in the sum and each \( n_{k,\ell} \) is the number of remainder variables included in the product that defines each element, or 1 if there are none. The reformulation also requires constraints involving the unit-product variables \( y_{k,\ell,h} \), the unit variables \( u_{i,j} \), and the remainder variables \( r_1 \) as shown in Equation [33]. It is clear that this reformulation technique could be applied successively to each monomial \( c_k m_k \) in a polynomial \( g \) as given in Equation [16] to yield a linear approximation \([g]\) of that polynomial:

\[
[g] = c_1 \cdot [m_1] + c_2 \cdot [m_2] + \cdots + c_1 \cdot [m_t] = \sum_{k=1}^{t} \sum_{\ell=1}^{s_k} \sum_{h=1}^{n_{k,\ell}} c_{k,\ell} y_{k,\ell,h}
\]

(41)

### 3.2.6 Bounds on Error from Linearization

In this section we shall consider the error introduced by the linear reformulation process described above. The error is a function of the error limits \( \kappa_1, \kappa_2, \ldots, \kappa_n \) set by the user for the variables \( x_i \) used in the polynomial expression \( g \). It follows from Equations [23] and [27] that for \( n_r > 1 \) the difference between a linearized element \([z_{k,\ell}]\) and its true value \( z_{k,\ell} \) is exactly:

\[
[z_{k,\ell}] - z_{k,\ell} = a_{k,\ell} (u_{i_{1,j_1}}u_{i_{2,j_2}} \cdots u_{i_{n_{n,u}},j_{n_u}}) (r_{h_1} + r_{h_2} + \cdots + r_{h_{n_r}} - r_{r_1}r_{r_2} \cdots r_{r_{n_r}})
\]

(42)

Note that the constant \( a_{k,\ell} \) contains the product of several error limits \( \kappa \) as shown in Equation [22]. In the case that \( n_r = 0 \) or \( n_r = 1 \) no error is introduced; error is introduced only by linearizing elements with \( n_r \geq 2 \) that contain the products of two or more continuous remainder variables.

According to Proposition 2 the difference \([z_{k,\ell}] - z_{k,\ell}\) shown in Equation [42] is bounded by:

\[
0 \leq [z_{k,\ell}] - z_{k,\ell} \leq a_{k,\ell} (u_{i_{1,j_1}}u_{i_{2,j_2}} \cdots u_{i_{n_{n,u}},j_{n_u}}) (n_r - 1), \text{ if } a_{k,\ell} > 0
\]

(43)

As the product \( (u_{i_{1,j_1}}u_{i_{2,j_2}} \cdots u_{i_{n_{n,u}},j_{n_u}}) \) of the unit variables must be zero or one these error bounds are simplified to:

\[
0 \leq [z_{k,\ell}] - z_{k,\ell} \leq a_{k,\ell} (n_r - 1), \text{ if } a_{k,\ell} > 0, n_r > 0
\]

(44)

with the corresponding bounds for a negative coefficient \( a_{k,\ell} \):

\[
a_{k,\ell} (n_r - 1) \leq [z_{k,\ell}] - z_{k,\ell} \leq 0, \text{ if } a_{k,\ell} < 0, n_r > 0
\]

(45)

Using these relationships it is possible to create a pair of functions to bound the error in the linear approximation of a polynomial.

**Proposition 4 (Scalar bounds on error in linear approximation)** For a polynomial \( g \) as given in Equation [16] let us define the lower error bound \( E^- (g) \) to be the amount by which the linearization \([g]\) might underestimate the true value of \( g \) and the upper error bound \( E^+ (g) \) to be the amount by which the linearization \([g]\) might overestimate the true value of \( g \):

\[
E^- (g) \leq [g] - g \leq E^+ (g)
\]

(46)

With reference to the error bounds on elements \( z_{k,\ell} \) given in Equations [44] and [45], the sum of unit-product variables \( y_{k,\ell,h} \) that defines each linearized item \([z_{k,\ell}]\) shown in Equation [33] and the sum of unit-product variables that defines each linearized polynomial \([g]\) shown in Equation [41] these error bounds \( E^- (g) \) and \( E^+ (g) \) can be computed as follows. The lower error bound \( E^- (g) \) is given by:

\[
E^- (g) = \left\{ \sum_{k=1}^{t} \sum_{\ell=1}^{s_k} c_{k,\ell} (n_{k,\ell} - 1) : c_{k,\ell} a_{k,\ell} < 0 \right\}
\]

(47)
where the sum includes an element only when the product of coefficients $c_k a_{k,t}$ is negative. The corresponding expression for the upper error bound $E^+(g)$ is:

$$E^+(g) = \left\{ \sum_{k=1}^{s} \sum_{t=1}^{r} c_k a_{k,t} (n_{k,t} - 1) : c_k a_{k,t} > 0 \right\}$$

including elements only for positive products of coefficients $c_k a_{k,t}$. Note that the product $c_k a_{k,t} (n_{k,t} - 1)$ is nonzero only when $n_{k,t} > 1$ (equivalently when $n_r > 1$).

**Example 6** Returning to the element $z_{9,12}$ given in Example 3 you can see from the reformulations for the individual variables $x_2$ and $x_3$ given in Example 1 that the element $z_{9,12} = 2.5 r_2 r_3$ is in fact the product $(k_2 r_2) (k_3 r_3)$. Thus according to Equation 22 the constant $a_{9,12} = \frac{1}{2} k_2 k_3$. Equation 44 says that the difference between the linearized element and its original reformulation must be bounded by:

$$0 \leq [z_{9,12}] - z_{9,12} \leq \frac{1}{2} k_2 k_3 (2 - 1)$$

Substituting $k_2 = 2.5$ and $k_3 = 1$ gives $0 \leq [z_{9,12}] - z_{9,12} \leq 1.25$. Error bounds for complete polynomials can be computed using Equations 47 and 48: For example the bounds on the objective function:

$$f(x_1, x_2, x_3) = 5x_2 + x_3 + x_1^2 - 2x_1 x_2 - 3x_1 x_3 + 5x_2 x_3$$

of the problem PP1 in Example 2 turn out to be $E^-(f) = 0$ and $E^+(f) = 1.25$.

### 3.2.7 Linear Bounds on Polynomials

The reformulation and linearization procedure above, along with the computed error bounds, allow us to compute linear expressions that provide upper and lower bounds on any polynomial $g(x)$. These bounds are valid for all values of $x$ within the feasible set $\Phi_x$.

**Proposition 5 (Linear Bounds on Polynomials)** Consider a polynomial $g$ which is a function of the variables $x = (x_1, x_2, \ldots, x_n)$. Let $\Phi_x$ denote the set of values of $x$ that satisfy the bound constraints $\alpha_i \leq x_i \leq \beta_i$ as shown in Equation 3. Let $[g]$ denote the linear reformulation of $g$ according to the procedure given above. Let us introduce the notation $\|g\|$ for the linear lower bound on the polynomial $g$:

$$\|g\| = [g] - E^-(g)$$

And similarly $\|g\|$ for the linear upper bound:

$$\|g\| = [g] - E^+(g)$$

The construction above guarantees that the linear bounds are correct for all feasible values of the original variables $x$ and the corresponding values of the reformulated variables $w$:

$$\|g\|(w) \leq g(x(w)), \quad \forall x(w) \in \Phi_x$$

$$\|g\|(w) \geq g(x(w)), \quad \forall x(w) \in \Phi_x$$

where $x(w)$ is the point in original variables corresponding to the reformulated point $w$. The construction guarantees that every point $x$ in $\Phi_x$ has at least one corresponding point $w$ in $\Phi_w$, and that each point $w$ maps to a unique point $x(w)$; however there may be several feasible points $w(x)$ to represent any given $x$. Note that the reformulation $[g]$ is a linear function of the unit variables, remainder variables, and unit-product variables which constitute the vector $w$ shown in Equation 5. As each error bound $E^-(g)$ and $E^+(g)$ is a real number, the linear bounds $\|g\|$ and $\|g\|$ defined above are therefore also linear functions of the variables in $w$. 

12
3.3 Pair of Bounding Mixed Binary Linear Programs

We can use the linear bounds on polynomials described above to generate more and less restrictive versions of the original polynomial program PP (inner and outer approximations). As each reformulated program will contain several binary unit variables as well as continuous remainder variables and unit-product variables, it will be a mixed integer linear program whose integer variables are binary. For the optimistic case in which a lower bound on the global minimum is desired, the linear lower bound \( f \) on the objective function \( f \) should be used in the optimistic reformulated program \( LP^-_o \). Also, each constraint \( g_i \leq 0 \) in the polynomial program PP should be replaced in the linear program \( LP^-_o \) with its linear lower bound \( g_i \leq 0 \), which is less restrictive. For standardization each constraint \( g_i \geq 0 \) with the inequality in the opposite direction should be replaced with the equivalent constraint \(-g_i \leq 0\), and each equality constraint \( g_i = 0 \) replaced with the equivalent pair of constraints \( g_i \leq 0 \) and \(-g_i \leq 0\). Thus the looser problem \( LP^-_o \) is given by:

\[
\begin{align*}
\text{minimize} & \quad \|f\|(w) \\
\text{subject to} & \quad g_1 \cdot (w) \leq 0, \quad g_2 \cdot (w) \leq 0, \quad \ldots, \quad g_q \cdot (w) \leq 0 \\
& \quad w \in \Phi_w
\end{align*}
\]

Let us use \( w^- \) to designate the point at which the minimum solution to the optimistic program \( LP^-_o \) occurs, and \( \|f\|(w^-) \) to denote the value of the linearized version of the objective function at that point. The linear reformulation \( LP^-_o \) can be infeasible only if the original polynomial program PP is infeasible. It can happen that the relaxed program \( LP^-_o \) is feasible although the original program PP is not.

Similarly, for the pessimistic case in which an upper bound on the global minimum is desired, the objective function and each constraint in PP should be replaced with its linear upper bound to produce the tighter reformulated linear program \( LP^+_o \):

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad g_1(x) \leq 0, \quad g_2(x) \leq 0, \quad \ldots, \quad g_q(x) \leq 0 \\
& \quad x \in \Phi_x
\end{align*}
\]

Let us use \( w^+ \) to denote the point at which the minimum solution \( \|f\|(w^+) \) of the pessimistic program \( LP^+_o \) occurs. The pessimistic reformulation \( LP^+_o \) can be infeasible even if the original program PP is feasible; however if \( LP^+_o \) is infeasible then PP must be infeasible as well.

We must use care in interpreting the solutions to the reformulated linear programs \( LP^-_o \) and \( LP^+_o \). Let us use \( x^* \) to designate a point at which the global minimum solution \( f(x^*) \) to PP occurs. Let us say that a point \( x \) in the reformulated variables is polynomial feasible if the corresponding point \( x(w) \) in the original variables satisfies the bound constraints \( x_1 \leq x_1 \leq \beta_1 \) and the polynomial constraints \( g_i \leq 0 \) in PP.

The properties of the solutions to the reformulated programs are as follows. The solution to \( LP^-_o \) places a lower bound on the true solution to PP (there is no better solution):

\[
\|f\|(w^-) \leq f(x^*)
\]  \hspace{1cm} (51)

If it happens that \( w^- \) is polynomial feasible, then the value of the original polynomial objective function \( f \) evaluated at the corresponding point \( x(w^-) \) is an upper bound on the true solution to PP (there is a solution at least that good):

\[
f(x^*) \leq f(x(w^-)), \quad \text{if } x(w^-) \in \Phi_x \text{ and } g_1(x(w^-)) \leq 0, \quad g_2(x(w^-)) \leq 0, \quad \ldots, \quad g_q(x(w^-)) \leq 0
\]

However if the solution point \( w^- \) to \( LP^-_o \) is not polynomial feasible then the loose reformulation \( LP^-_o \) does not provide any upper bound on the true solution to PP; in that case it is necessary to use an alternate means to generate an upper bound. One way is to use the pessimistic reformulation \( LP^+_o \) to compute an
upper bound on the global minimum solution to PP. If the tighter program $LP^+_w$ is feasible then its solution is an upper bound on the global minimum:

$$f(x^*) \leq \|f\|(w^+)$$  \hspace{1cm} (53)

If the reformulation $LP^+_w$ is feasible, then by the construction of $LP^+_w$ the point $x(w^+)$ in the original variables corresponding to the solution point $w^+$ to $LP^+_w$ must be polynomial feasible. Therefore we can use the value of the original objective function $f$ at that point as a tighter upper bound on the global minimum:

$$f(x^*) \leq f(x(w^+)) \leq \|f\|(w^+)$$  \hspace{1cm} (54)

In the case that $LP^+_w$ is infeasible it does not provide an upper bound on the global solution to PP; and such a result does not prove that PP is infeasible.

**Example 7** Let us consider problem PP1 from Example 2. The solution $\|f\|(w^-) = -124.799$ to the optimistic reformulation $LP_1^{-}(3,2,2)$ occurs at the point $w^- : (u_{1,1} = 0, u_{1,2} = 1, u_{1,3} = 0, u_{2,1} = 0, u_{2,2} = 0, u_{3,1} = 1, u_{3,2} = 1, r_1 = 0.666667, r_2 = 0, r_3 = 1)$. This value $\|f\|(w^-)$ is a lower bound on the true global minimum of PP1. The corresponding point in the original variables $x(w^-) : (x_1 = 3, x_2 = 0, x_3 = 8)$ happens to be polynomial feasible, satisfying the bound constraints and two additional constraints in PP1. The original objective function evaluated at this point has value $f(x_1 = 3, x_2 = 0, x_3 = 8) = -119$. This value $f(x(w^-))$ is an upper bound bound on the true global minimum of PP1. Thus the true global minimum solution to PP1 lies in the interval $[-124.799, -119]$. It turns out that the global minimum to PP1 is indeed $-119$ which we can prove by reformulating the problem using smaller error limits and correspondingly more unit and unit-product variables.

**Example 8** Let us consider a different polynomial optimization problem PP2 which is used as Example 2 in Li and Chang [7]:

minimize : $0.6224x_3x_4 + 19.84x_1^2x_3 + 3.1661x_3^2x_4 + 1.7781x_2x_3^2$

subject to : $x_1 \geq 0.0193x_3$
$x_2 \geq 0.0954x_3$
$1.3333x_1^2\pi + x_3^2x_4\pi \geq 750.173$
$x_4 \leq 240$

and : $x_1 \in \{1, 1.0625, 1.125, 1.1875, 1.25, 1.3125, 1.375\}$
$x_2 \in \{0.625, 0.6875, 0.75, 0.8125, 0.875, 0.9375, 1\}$
$47.5 \leq x_3 \leq 52.5$
$90 \leq x_4 \leq 112$
$\pi = 3.14159$

Note that we can accommodate the discrete variables $x_1$ and $x_2$ by reformulating each of them with binary unit variables as in Proposition [1] but without the continuous remainder variable $r_1$ or $r_2$. The rest of the method works without modification. In this case the reformulation of each variable is given by:

\[
x_1 = 1 + 0.0625u_{1,1} + 0.125u_{1,2} + 0.25u_{1,3} \\
x_2 = 0.625 + 0.0625u_{2,1} + 0.125u_{2,2} + 0.25u_{2,3} \\
x_3 = 47.5 + 0.00976562u_{3,1} + 0.0195312u_{3,2} + 0.0390625u_{3,3} + 0.078125u_{3,4} + 0.15625u_{3,5} + 0.3125u_{3,6} + 0.625u_{3,7} + 1.25u_{3,8} + 2.5u_{3,9} + 0.00976562r_3 \\
x_4 = 90 + 0.6875u_{4,1} + 1.375u_{4,2} + 2.75u_{4,3} + 5.5u_{4,4} + 11u_{4,5} + 0.6875r_4
\]
using the error tolerances \((\kappa_1, \kappa_2, \kappa_3, \kappa_4) = (0.0625, 0.00976562, 0.6875, 0)\) and the corresponding numbers of unit variables \((\sigma_1, \sigma_2, \sigma_3, \sigma_4) = (3, 9, 5, 0)\) The loose reformulation \(L P_2^{\sigma_1, \sigma_2, \sigma_3, \sigma_4}\) of this problem \(P P_2\) has the solution \(||f||(w^-) = 6395.51\) which is a lower bound on the true minimum \(f(x^+)\) as shown in Equation [51]. This lower bound occurs at the reformulated point \(w^-\) : \((u_{1,1} = 0, u_{1,2} = 0, u_{1,3} = 0, u_{2,1} = 0, u_{2,2} = 0, u_{2,3} = 0, u_{3,1} = 0, u_{3,2} = 0, u_{3,3} = 0, u_{3,4} = 0, u_{3,5} = 0, u_{3,6} = 0, u_{3,7} = 0, u_{3,8} = 0, u_{3,9} = 0, u_{4,1} = 0, u_{4,2} = 0, u_{4,3} = 0, u_{4,4} = 0, u_{4,5} = 0, r_1 = 0, r_2 = 0, r_3 = 0, r_4 = 0, r_5 = 0)\). The corresponding point \(x(w^-)\) : \((x_1 = 1, x_2 = 0.625, x_3 = 47.5, x_4 = 90, \pi = 3.14159)\) in the original variables happens to be polynomial feasible; the original objective function evaluated at this point has value \(f(x_1 = 1, x_2 = 0.625, x_3 = 47.5, x_4 = 90, \pi = 3.14159)\) = 6395.51. Thus the true global minimum solution to \(P P_2\) lies in the interval \([6395.51, 6395.51]\). This is a better solution than computed in Li and Chang [7]. More importantly, the method presented here assures that this is in fact the global minimum as there can exist no better solution than the lower bound 6395.51.

**Example 9** We consider the problem \(P P_3\) which is Problem 338 in Schittkowski [10] and Example 3 in [7]:

\[
\begin{align*}
\text{minimize} & \quad -x_1^2 - x_2^2 - x_3^2 \\
\text{subject to} & \quad \frac{1}{4} x_1 + x_2 + x_3 = 1 \\
& \quad x_1^2 + \frac{1}{3} x_2^2 + \frac{1}{4} x_3^2 = 4 \\
& \quad -2 \leq x_1 \leq 2 \\
& \quad -2.45 \leq x_2 \leq 2.45 \\
& \quad -4 \leq x_3 \leq 4 
\end{align*}
\]

(The bounds on the variables were not specified in the original formulation but they are implied by the equality constraints and made explicit here.) The program is reformulated using \((\sigma_1, \sigma_2, \sigma_3) = (7, 7, 7)\) and correspondingly \((\kappa_1, \kappa_2, \kappa_3) = (0.03125, 0.0382813, 0.0625)\). The solution to the loose reformulation \(L P_3^{\sigma_1, \sigma_2, \sigma_3}(7, 7, 7)\) is \(||f||(w^-) = -10.9965\) which occurs at \(x(w^-)\) : \((x_1 = -0.375, x_2 = -1.65897, x_3 = 2.84647)\). However this point \(x(w^-)\) is not polynomial feasible; it violates the second constraint in \(P P_3\). It happens that the tight reformulation \(L P_3^{\sigma_1, \sigma_2, \sigma_3}(7, 7, 7)\) is infeasible, as is often the case for polynomial programs with nonlinear equality (as opposed to inequality) constraints. As an alternate means of finding an upper bound on the true global minimum to \(P P_3\) (and a polynomial-feasible point at which that upper bound occurs) we can generate a focused problem that uses narrower ranges of values for the variables \(x_1, x_2,\) and \(x_3\) concentrated near the polynomial-infeasible point \(x(w^-)\) where the solution to \(L P_3^{\sigma_1, \sigma_2, \sigma_3}(7, 7, 7)\) occurs. If we choose a range focused on \(x(w^-) \pm \kappa_1\) for each variable then the resulting polynomial program is the following (let us call it \(P P_3.2\)):

\[
\begin{align*}
\text{minimize} & \quad -x_1^2 - x_2^2 - x_3^2 \\
\text{subject to} & \quad \frac{1}{4} x_1 + x_2 + x_3 = 1 \\
& \quad x_1^2 + \frac{1}{3} x_2^2 + \frac{1}{4} x_3^2 = 4 \\
& \quad -0.40625 \leq x_1 \leq -0.34375 \\
& \quad -1.69725 \leq x_2 \leq -1.62069 \\
& \quad 2.78397 \leq x_3 \leq 2.90897 
\end{align*}
\]

This program \(P P_3.2\) can be reformulated using the same numbers of unit variables \((\sigma_1, \sigma_2, \sigma_3) = (7, 7, 7)\) used for \(P P_3\) above, which now produce smaller error limits \((\kappa_1, \kappa_2, \kappa_3) = (0.000488281, 0.000598125, 0.000976562)\) due to the narrowed bounds on each variable. The pessimistic reformulated focused program \(P P_3.2^+\) has solution \(-10.9928\) which is achieved at the polynomial-feasible point \(x(w^+)\) : \((x_1 = -0.366211, x_2 = -1.6622, x_3 = 2.84531)\). Combining the results of these reformulations \(L P_3^{\sigma_1, \sigma_2, \sigma_3}(7, 7, 7)\) and \(P P_3.2\) shows that the global solution to the original problem \(P P_3\) must lie within the interval \([-10.9965, -10.9928]\). Again, in contrast to the approximation method presented in Li and Chang [7], the bounding approach presented here guarantees that there cannot exist a better minimum than \(-10.9965\).
3.4 Alternative Formulation: Allowed Constraint Violation

Another way to use the reformulation technique described above would be to compute a single mixed binary linear program LP from the original program PP, using the linearized version $[g]$ of each constraint $g$ and the linearized version $[f]$ of the objective $f$. The error bounds on each $[g]$ could then be used to calculate the possible constraint violation $\tau_i$ for each constraint $g_i \leq 0$, and from these the maximum possible constraint violation $\tau$ across all constraints could be computed. The interval $[z^-, z^+]$ would then contain the global optimum $z^\ast$ for the variant of the problem $PP$ in which each constraint is nearly satisfied (within the feasibility tolerance $\tau$). The user could adjust the $\sigma_i$ parameters in the pre-solution phase in order to achieve the desired feasibility tolerance. Note that in this alternative version, feasibility or infeasibility of the reformulated optimization problem does not guarantee feasibility or infeasibility of the original polynomial problem.

4 DISCUSSION

4.1 Problem Size

Let us now consider the number of variables and constraints that must be added to a polynomial optimization problem $PP$ in the course of reformulation and linearization as described above. We assume that we have a list $m = \{m_1, m_2, \ldots, m_t\}$ of all the terms used in monomials in the polynomial program $PP$ and we use $d$ to denote the greatest degree of any term in $m$. We use $\sigma = \sup_i \sigma_i$ to denote the largest number of unit variables required to reformulate any variable $x_i$.

The original program $PP$ shown in Equation 1 contains:

- $n$ bounded real variables $x_1, x_2, \ldots, x_n$
- $q$ polynomial constraints $g_1 \leq 0, g_2 \leq 0, \ldots, g_q \leq 0$.
- One polynomial objective function $f$

In the reformulated program $LP$ there are:

- Exactly $n$ remainder variables $r_1, r_2, \ldots, r_n$ which are real variables
- Exactly $\sigma_1 + \sigma_2 + \cdots + \sigma_n$ unit variables which are binary 0-1 variables:

\[
\begin{array}{cccc}
  u_{1,1} & u_{1,2} & \cdots & u_{1,\sigma_1} \\
  u_{2,1} & u_{2,2} & \cdots & u_{2,\sigma_2} \\
  \vdots & \vdots & \ddots & \vdots \\
  u_{n,1} & u_{n,2} & \cdots & u_{n,\sigma_n}
\end{array}
\]

Let us use $\phi$ to represent the number of binary variables in a reformulated mixed binary linear program $LP$. The text above shows that $\phi = \sum_{i=1}^{n} \sigma_i$ which implies $\phi \leq n\sigma$.

- Not more than $td(\sigma + 2)\sigma$ unit-product variables which are real variables. For each monomial term $m_k \in M$:

\[
\begin{array}{cccc}
  y_{k,1,1} & y_{k,1,2} & \cdots & y_{k,1,n_k,1} \\
  y_{k,2,1} & y_{k,2,2} & \cdots & y_{k,2,n_k,2} \\
  \vdots & \vdots & \ddots & \vdots \\
  y_{k,s_k,1} & y_{k,s_k,2} & \cdots & y_{k,s_k,n_k,s_k}
\end{array}
\]

where each $s_k$ is the number of terms $z_{k,1}, z_{k,2}, \ldots, z_{k,s_k}$ needed to express the $k$-th monomial term $m_k$ according to Equation 18 and $n_{k,\ell}$ is the number of remainder variables $n_r$ in the $\ell$-th term $z_{k,\ell}$.
or one if $n_r = 0$. Note that $n_r$ is limited by the maximum degree $d$ of polynomial expressions in PP: $n_r \leq d$. Note also that the number of terms $s_k$ for each monomial $m_k$ is limited by the degree $d_k$ of $m_k$ and the maximum number $\sigma$ of binary variables used to represent any original variable: $s_k \leq (\sigma + 2)^{d_k}$, using the product in Equation 19. Let us use $\psi$ to represent the number of unit-product variables. As shown here $\psi \leq td(\sigma + 2)^d$.

- Not more than $td^2(d + 1)(\sigma + 2)^d$ product constraints, as for each product variable $y_{k,\ell,h}$ added it is necessary to add $n_r(n_u + 1)$ constraints to satisfy Proposition 3 and both the numbers $n_r$ of remainder variables (or one if there are none) and $n_u$ of unit variables in any element $z_{k,\ell}$ are limited by the maximum degree $d$ of polynomials. Let us use $\rho$ to denote the number of additional constraints.

- Linearized versions of the original $q$ constraints.

- The linearized objective function $[f]$.

To summarize this using $O$-notation, the reformulated linear program LP will have a number of variables that is $O(td^2\sigma^d)$ including $O(n\sigma)$ binary 0-1 variables; LP will also have a number of additional constraints (besides the original $q$ in PP) that is $O(td^3\sigma^d)$. Note that all of the added constraints involve binary variables.

An important consideration is the time required to solve each mixed binary linear program. For a program with $\phi$ binary variables, in the worst case a branch-and-bound algorithm would require enumeration of the $2^\phi$ distinct combinations of 0 and 1 for each variable, and solving a continuous linear program for each case. As $\phi$ is $O(n\sigma)$ the time required to solve each linear reformulation is $O(2^{n\sigma})$ multiplied by the time required to solve a standard linear program with $n + \phi + \psi$ variables and $\rho \sim O(td^3\sigma^d)$ constraints.

### 4.2 Conclusion

This paper presented a reformulation and linearization technique to generate an approximate solution to a polynomial optimization problem. The approximate solution takes the form of interval bounds on the true global optimum. In the reformulation step each variable in the original polynomial problem is replaced by a sum of binary and continuous variables, the number of which depends on the error limit specified by the user for each original variable. In the linearization step products of continuous variables are replaced by sums of those variables, and constraints are added to the problem to limit the differences between those sums and products. Bounds on the error introduced by linearization are computed, and with these bounds a pair of mixed binary linear programs is created whose solutions bound the solution to the original polynomial program. The tightness of the generated interval bounds depends on the error limits specified by the user, which also determine the size of each reformulated program and consequently the time required to solve it.

### REFERENCES

[1] A. Charnes and W. W. Cooper. Programming with linear fractional functionals. *Naval Research Logistics Quarterly*, 9:181–186, 1962.

[2] C. A. Floudas and V. Visweswaran. A global optimization algorithm (GOP) for certain classes of nonconvex NLPs–I. Theory. *Computers and Chemical Engineering*, 14(2):1397–1417, 1990.

[3] C. A. Floudas and V. Visweswaran. A global optimization algorithm (GOP) for certain classes of nonconvex NLPs–II. Application of theory and test problems. *Computers and Chemical Engineering*, 14(2):1419–1434, 1990.

[4] Reiner Horst and Hoang Tuy. *Global Optimization: Deterministic Approaches*. Springer, third, revised and enlarged edition, 2006.
[5] Jean B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001.

[6] Jean B. Lasserre. Semidefinite programming vs. LP relaxations for polynomial programming. *Mathematics of Operations Research*, 27(2):347–360, 2002.

[7] Han-Lin Li and Ching-Ter Chang. An approximate approach of global optimization for polynomial programming problems. *European Journal of Operational Research*, 107:625–632, 1998.

[8] Joseph W. Norman. *Analysis of Imprecise Decision Models*. PhD thesis, Stanford University, 2002.

[9] Joseph W. Norman. A tutorial introduction to the logic of parametric probability. Preprint at arXiv:1201.3142 [math.LO], May 2012.

[10] K. Schittkowski. *More Test Examples for Nonlinear Programming Codes*. Number 282 in Lecture notes in Economics and Mathematical Systems. Springer-Verlag, Heidelberg, 1987.

[11] Hanif D. Sherali and Cihan H. Tuncbilek. A global optimization algorithm for polynomial programming problems using a reformulation-linearization technique. *Journal of Global Optimization*, 2:101–112, 1992.

*Implementation note: pqlsh-8.1.1+cplex+openmp 2012-05-29*