Reduction Constraints for the Global Optimization of NLPs

Leo Liberti

DEI, Politecnico di Milano, P.zza L. Da Vinci, 20133 Milano, Italy

(liberti@elet.polimi.it)

July 2002

Abstract

Convergence of Branch-and-Bound algorithms for the solution of NLPs is obtained by finding ever nearer lower and upper bounds to the objective function. The lower bound is calculated by constructing a convex relaxation of the NLP. Reduction constraints are new linear problem constraints which are (a) linearly independent w.r.t. the exisisting constraints; (b) redundant w.r.t. the original NLP formulation; (c) not redundant w.r.t. its convex relaxation. Thus, they can be successfully employed to reduce the feasible region of the convex relaxation without cutting the feasible region of the original NLP.

Keywords: global optimization, valid cut, NLP, branch-and-bound

1 Introduction

Global nonlinear optimization has witnessed a remarkable theoretical development in the last decade (Adjiman et al. [1998b], Pardalos et al. [2000], Floudas [2001]). A lot of new algorithms have been proposed, either geared towards a specific problem or class of problems (Hirafuji and Hagan [2000], Hägglöf et al. [1995]), or more general (Smith and Pantelides [1999], Adjiman et al. [1998a], Adjiman et al. [1998c], Ryoo and Sahinidis [1995], Kesavan and Barton [2000], Vaidyanathan and El-Halwagi [1996]). Software implementations of these algorithms, however, are scarce, not easily available, and more importantly, not really ready for practical use. One could draw a parallel with the development of LP solvers, where the straight implementation of an algorithm is usually not enough to give birth to a good piece of code; all sorts of "implementation tricks" are necessary to this end. We feel that at the present state, global optimization solvers for NLPs are in their basic form, with the algorithm in place but devoid of other speeding-up devices which are crucial for practical usability.

In this article we describe the theory and implementation of one such speeding-up device, named "method of reduction constraints", to be used in a Branch-and-Bound solution framework. The basic idea is as follows: new linear constraints are created in the NLP, which allow for smaller feasible regions during the solution of the lower bounding problem.

Before attempting a description of this method, we need some context. The type of NLP this article is concerned with is completely general:

$$\left.\begin{array}{rcl} \min_x & f(x) & \\ \alpha & \leq g(x) \leq & \beta \\ a & \leq x \leq & b \end{array}\right\} \qquad (1)$$

where $x \in \mathbb{R}^n$ are the problem variables, α, β are the lower and upper bounds of the constraints, a, b are the lower and upper bounds of the variables, the function $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function and $g : \mathbb{R}^n \to \mathbb{R}^m$ are the constraints of the problem. Objective function and constraints may be nonlinear.

2 Global Optimization with Symbolic Reformulation

This method has been devised as a speeding-up device for the Symbolic Reformulation Spatial Branch-and-Bound Algorithm described in Smith and Pantelides [1999]. The most important novelty in Smith's Branch-and-Bound algorithm is the technique used to formulate the lower bounding problem. First, the original NLP is reformulated symbolically into a *standard form*; from this it is easy to derive a convex relaxation.

In the standard form, all the nonlinear parts in the objective function and constraints of the original NLP are isolated in equality constraints of either one of the forms:

$$\begin{aligned} w_i &= w_j \otimes w_k \\ w_i &= h(w_j), \end{aligned}$$

where $\otimes$ indicates a nonlinear binary operator and h a nonlinear unary operator. The nonlinearities are then substituted by the newly created added variables w_i. We then obtain a problem of the form:

$$\left.\begin{array}{lcl} \min w_1 & & \\ l \leq & Aw & \leq u \\ w_{i_1} & = & w_{j_1} \otimes_1 w_{k_1} \\ \ldots & & \\ w_{i_{\nu+1}} & = & h_{\nu+1}(w_{j_{\nu+1}}) \\ \ldots & & \\ w^L \leq & w & \leq w^U, \end{array}\right\} \quad (2)$$

where w is the vector of variables, A is the matrix of the linear constraints, l, u, w^L, w^U are real vectors, and the nonlinear parts have been isolated in simple equality constraints, as described above.

There are two important points about the standard form.

- The problem in standard form is completely equivalent to the original NLP. Although the standard problem has more problem variables, the projection of its feasible region on the space of the original variables is the same as the feasible region of the original problem; furthermore, it has the same same globally optimal solution.
- Because the nonlinear parts have been isolated in simple equality constraints, it is very easy to devise symbolic algorithms for the manipulation of the equations. In particular, it is easy to obtain the convex relaxation of the problem automatically.

In order to obtain the convex relaxation of a problem in standard form, one just cycles over the simple nonlinear constraints and replaces each one with its convex relaxation. For example, the convex relaxation for a bilinear term $z = xy$ is as follows (McCormick [1976]):

$$\left.\begin{array}{lcl} z & \geq & x^L y + y^L x - x^L y^L \\ z & \geq & x^U y + y^U x - x^U y^U \\ z & \leq & x^U y + y^L x - x^U y^L \\ z & \leq & x^L y + y^U x - x^L y^U. \end{array}\right\} \quad (3)$$

The above relaxation has been proved to be the convex envelope for the nonconvex surface $z = xy$ (Al-Khayyal and Falk [1983]). The inequalities in (3) can be deduced from the fact that the values $x - x^L$, $x^U - x$, $y - y^L$, $y^U - y$ are all nonnegative, so any product between them will also be nonnegative. Therefore the following are valid problem constraints:

$$\begin{aligned} (x - x^L)(y - y^L) &\geq 0 \\ (x - x^L)(y^U - y) &\geq 0 \\ (x^U - x)(y - y^L) &\geq 0 \\ (x^U - x)(y^U - y) &\geq 0. \end{aligned}$$

Expanding these products and substituting z for xy gives the linear envelope (3). Refer to Smith and Pantelides [1999], Liberti and Pantelides [To appear], to see other examples of convex relaxations.

It appears evident that the construction of the convex relaxation, as described above, produces convex NLPs (or LPs, depending on the relaxations used) having larger feasible regions than the original NLP. This causes the lower bounding solution to be likely to be located outside the feasible region of the *original* NLP. Because of the way the Branch-and-Bound algorithm is structured, this in turn causes the upper bounding solution to be calculated by locally solving an NLP (a very computationally expensive step). By contrast, if the lower bounding solution (call it x^*) is found in the feasible region of the original NLP, it suffices to compute $f(x^*)$ to identify a valid upper bound to the objective function. Hence a lot can be gained by restricting (cutting) the feasible regions of the relaxed convex problems.

A *reduction constraint* is a new problem constraint which is

1. redundant with respect to the original NLP, i.e. it does not restrict its feasible region;
2. not redundant (i.e. a valid cut) with respect to the convex relaxation.

3 Theory of Reduction Constraints

The idea stems from the work of Sherali and Alameddine (cf. Sherali and Alameddine [1992], Sherali and Wang [2001]); Smith mentioned it as a "trick" that was necessary to solve one of the test problems in his Ph.D. thesis (Smith [1996]). However, we think it is possible to apply these concepts to any problem in the general form (1), more precisely after the conversion to standard form (2).

For example, consider a non-convex NLP that involves:

- bilinear products of the form $x_i y$ for all $i \in I$, where I is a variable index set;
- a normalization constraint of the form

$$\sum_{i \in I} x_i = 1. \tag{4}$$

Such situations are very common in process engineering models where, for example, x_i represent mass or molar functions of a set of components i whole y is an extensive variable (e.g. flowrate).

The conversion of the above problem to standard form will result in the introduction of new variables w_i defined as $w_i \equiv x_i y$ for all $i \in I$. During the convexification process, the constraints $w_i = x_i y$ will be replaced by the convex envelopes (3).

A linear reduction constraint of the form

$$\sum_{i \in I} w_i = y \tag{5}$$

can be obtained by multiplying both sides of eqn. (4) by y. This constraint is not redundant with respect to the convex relaxation of the original NLP. In fact, Smith reported (Smith [1996]) that the use of such a constraint in the optimization of distillation column models was crucial in actually being able to obtain the global solution within reasonable amounts of computation time and resources.

Define an NLP as a set P containing an objective function f_P, a set of constraints C_P and a set of variable ranges R_P. Let m_P be the number of problem constraints and n_P the number of problem variables.

Let P be the original NLP, $\bar{P}$ the NLP in standard form and $\breve{P}$ its convex relaxation. Let J be a non-empty index set such that

$$c_i \equiv \sum_{j\in J} a_{ij}w_j - b_i = 0 \tag{6}$$

is a linear constraint of the standard problem $\bar{P}$ (where $a_{ij} \neq 0$ for all $j \in J$), and let w_l be a variable in $\bar{P}$. If for all $j \in J$ there is an index $\kappa(j)$ such that $w_{\kappa(j)} = w_j w_l$ is a constraint in $\bar{P}$ (this is equivalent to requesting that all products $w_j w_l$ appear in the original problem for all $j \in J$), then multiplying c_i by w_l produces a new linear constraint

$$r_{il} \equiv \sum_{j\in J} a_{ij}w_{\kappa(j)} - b_i w_l = 0, \tag{7}$$

which can be added to the problem in standard form $\bar{P}$. Because this constraint has been derived by information already existing in $\bar{P}$, it is redundant; however, because it is linear, it will not be changed by the relaxation procedure, so it will not be redundant in the convex relaxation $\breve{P}$.

Let c'_i be the constraint $\sum_{j\in J} a_{ij}w_j = 0$ (derived from c_i by setting the RHS to 0).

3.1 Proposition
Constraints c'_i and r_{il} are linearly independent.

Proof. First note that $\kappa : J \to \{1, \ldots, n_{\bar{P}}\}$ is injective: for if there existed $j_1, j_2 \in J$ such that $\kappa(j_1) = \kappa(j_2) = h$, we would then have the constraint $w_{j_1}w_l = w_h = w_{j_2}w_l$, i.e. $w_{j_1}w_l = w_{j_2}w_l$, which implies $w_{j_1} = w_{j_2}$ and thus that w_{j_1} and w_{j_2} are the same problem variable, that is, $j_1 = j_2$. Next, we shall show that $\kappa(J) \neq J$, i.e. κ is *not* a permutation of J. Suppose, to get a contradiction, that κ permutes J, and let t be the order of the permutation κ, so that κ^t is the identity; since $w_{\kappa(j)} = w_j w_l$ for all $j \in J$, we have $w_{\kappa^q(j)} = w_j w_l^q$ for all integers q such that $1 \leq q \leq t$. In particular we have $w_j = w_{\kappa^t(j)} = w_j w_l^t$, which implies that w_l has the constant value 1. This is false as in general w_l is a problem variable and not a constant. Now, since κ is injective and not a permutation of J, it follows that there exist $j \in J$ such that $\kappa(j) \notin J$. Thus constraint r_{il} depends on $w_{\kappa(j)}$ whereas c'_i does not. $\square$

One drawback of this theory is that it is based on the fact that bilinear products $w_j w_l$ should appear in formulation of the problem for all $j \in J$. This requirement can be partially relaxed by introducing new variables as well as new constraints; it suffices to require that the number of new constraints is higher than the number of new variables.

Consider a pair of linearly independent linear constraints in the standard form problem $\bar{P}$:

$$\sum_{j\in J} a_j w_j + a_h w_h - b = 0 \tag{8}$$

$$\sum_{j\in J} a'_j w_j + a'_h w_h - b' = 0. \tag{9}$$

where J is an index set such that $h \notin J$. Consider multiplying (8) and (9) by problem variable w_l; if constraints $w_{\kappa(j)} = w_j w_l$ already exist in the problem for all $j \in J$, but the product $w_h w_l$ does *not* exist in $\bar{P}$, we can nonetheless multiply (8), (9) by w_l to get:

- 1 new variable $w_{\kappa(h)}$ with associated constraint $w_{\kappa(h)} = w_h w_l$;
- 2 new reduction constraints:

$$\begin{aligned}\sum_{j\in J} a_j w_{\kappa(j)} + a_h w_{\kappa(h)} - b w_l &= 0\\ \sum_{j\in J} a'_j w_{\kappa(j)} + a'_h w_{\kappa(h)} - b' w_l &= 0.\end{aligned}$$

Note also that (8) and (9) are equivalent to:

$$\sum_{j\in J}(a_j a'_h - a'_j a_h)w_{\kappa(j)} - (ba'_h - b'a_h)w_l = 0,$$

which implies that the new constraints are linearly independent, for otherwise we would get $\frac{a_h}{a'_h} = \frac{b}{b'} = \frac{a_j}{a'_j}$ for all j, which would contradict the linear independence of (8) and (9).

By using induction, this reasoning can be applied to sets of constraints of any cardinality. Effective tightening occurs when the number of reduction constraints created is higher than the number of new variables.

Notice that we never accounted explicitely for the case where $h = l$ in $w_{\kappa(h)} = w_h w_l$. However, there is no particular issue about this instance: just define $w_{\kappa(h)} = w_h^2$.

4 Extensions and Generalizations

One possible generalization of reduction constraints would be to extend this reasoning to other nonlinear terms apart from bilinear products.

Start from a set of linear constraints $D = \{\sum a_{ij}x_j = b_i\}$; choose a nonlinear term $f(x)$ to multiply all the constraints in D. We get $\{\sum a_{ij}w_{\kappa(j)} - b_i w_l\}$ where $w_{\kappa(j)} \equiv f(x)x_j$ and $w_l = f(x)$. If $|\{w_{\kappa(j)}\}| < |D|$, we generate more new constraints than new variables. However, because the term $w_l \equiv f(x)$ is already present in the standard form problem (as the nonlinear term $f(x)$ is replaced by w_l during the standardization process), this procedure can be seen as an application of the usual reduction constraints creation procedure by means of bilinear terms: all we are doing is effectively using w_l as a candidate multiplier variable.

However, we can also approach the problem from a different point of view. Start with a nonlinear constraint:

$$\sum_{i\in I} f_i(x) = b, \tag{10}$$

and multiply it by a candidate multiplier variable x_l to get

$$\sum_{i\in I} x_l f_i(x) = bx_l.$$

If the terms $x_l f_i(x)$ are already in the formulation of the problem, we get a new linear reduction constraint

$$\sum_{i\in I} w_{\kappa(i)} - bx_l = 0,$$

where $w_{\kappa(i)} \equiv x_l f_i(x)$ for all $i \in I$.

There are two problems with this approach:

- in general, it is quite uncommon for all terms $f_i(x)$ and $x_l f_i(x)$ to exist in the formulation of the problem, so this technique might be applied to only a small class of problems;
- there are no guarantees that the newly created reduction constraint will be linearly independent of the existing linear constraints.

There are, however, certain well-structured problems which would benefit from such an application of reduction constraints creation. If, for example, the problem under consideration contains a lot of fractional terms like $\frac{h(x)}{x_l}$, it would be advantageous to carry out a procedure as described below.

Suppose we have an index set I and q constraints $c_1, \ldots, c_q$ of the form:

$$c_i \quad \equiv \quad \sum_{j \in I} \frac{f_{ij}(x)}{x_l} + \sum_{j \notin I} g_j(x) = b_i.$$

By multiplying these constraints by x_l we get

$$\sum_{j \in I} f_{ij}(x) + \sum_{j \notin I} x_l g_j(x) = b_i x_l.$$

Now, because variables $w_j \equiv f_{ij}(x)$ must have already been created by the standard form procedure, we need to create $n_P - |I|$ new variables $w_j \equiv x_l g_j(x)$ (where n_P is the total number of problem variables) to get q new linear reduction constraints $c'_1, \ldots, c'_q$ of the form:

$$c'_i \quad \equiv \quad \sum_{j \in I} w_j + \sum_{j \notin I} w_j - b_i x_l = 0.$$

Because we started the construction from general nonlinear constraints, there is no guarantee that the set $R = \{c'_i\}_{i=1,\ldots,q}$ will be linearly independent. Let R' be a linearly independent maximal subset of R. If $|R'| > n_P - |I|$ then the procedure is creating more new constraints than variables, so it is convenient.

Another generalization is deriving reduction constraints from linear *inequality* constraints of the type

$$\sum_{j=1}^{n} a_j x_j \quad \leq \quad b. \tag{11}$$

Provided $y \geq 0$, and $w_j = x_j y$ for all $j \leq n$, we can multiply constraint (11) by y to obtain the reduction constraint

$$\sum_{j=1}^{n} a_j w_j - by \quad \leq 0.$$

However, this ceases to hold if y^L, the lower bound on y, is negative. In this case we translate y so that it becomes nonnegative before multiplying the constraint by it. We then obtain

$$\sum_{j=1}^{n} a_j x_j (y - y^L) \quad \leq \quad b(y - y^L)$$

and hence the modified reduction inequality constraint

$$\sum_{j=1}^{n} a_j w_j - \sum_{j=1}^{n} a_j y^L x_j - by \quad \leq \quad by^L. \tag{12}$$

5 Numerical Results

We have chosen a selection of bilinear test problems from the oil industry literature to test the practical efficiency of the method of redundant constraints. Our main source for these problems has been Adhya et al. [1999]. Citing from their abstract,

> Pooling and blending problems occur frequently in the petrochemical industry where crude oils, procured from various sources, are mixed together to manufacture several end-products. Finding optimal solutions to pooling problems requires the solution of nonlinear optimization problems with multiple local minima.

The choice of blending problems was due to the fact that they provided a validation benchmark for the code ("small" blending problems, like Haverly's pooling problems) as well as a proper testbed ("large" blending problems, like the multi-quality problems proposed in the Adhya et al. [1999] article: examples 1, 2, 3 and 4).

Table 1 reports the number of main iterations (i.e. the total number of regions examined) taken by the spatial Branch-and-Bound code to find the global minimum. The first column describes the performance of the sBB code with reduction constraints added to the problem formulation, whereas the results in the second column have been obtained without reduction constraints. It is easy to see that reduction constraints make a huge performance difference.

N. of Iterations Problem Name	$oo\mathcal{OPS}$ **(RC)**	$oo\mathcal{OPS}$ (no RC)
Haverly 1	**1**	31
Haverly 2	**7**	43
Haverly 3	**7**	39
Foulds 2	**7**	131
Foulds 3	**1**	> 20,000
Foulds 4	**1**	> 20,000
Foulds 5	**1**	> 20,000
Ben-Tal 4	**1**	101
Ben-Tal 5	**1**	> 200,000
example 1	**5445**	11245
example 2	**11049**	83051
example 3	**7565**	> 200,000
example 4	**1467**	2887

Table 1: Numerical results.

6 Conclusion

We have introduced the idea of reduction constraints, the context where they can be used (i.e. Branch-and-Bound algorithms, in particular Smith's reformulation-based sBB algorithm). We have investigated some important points behind the concept of reduction constraints, and described the motivation for their usage within the field of Global Nonconvex Optimization. We have proposed some possible extensions to the concept of reduction constraint. Finally, our numerical results show that reduction constraints make a huge difference in terms of computational efficiency of the sBB algorithm.

References

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

C. Adjiman, S. Dallwig, C. Floudas, and A. Neumaier. A global optimization method, αBB, for general twice-differentiable constrained NLPs: I. Theoretical advances. *Computers & Chemical Engineering*, 22(9):1137–1158, 1998a.

C. Adjiman, C. Schweiger, and C. Floudas. Mixed-integer nonlinear optimization in process synthesis. In D.-Z. Du and P. E. Pardalos, editors, *Handbook of Combinatorial Optimization, vol. I*, volume 1, pages 1–76, Dordrecht, 1998b. Kluwer Academic Publishers.

C. S. Adjiman, I. P. Androulakis, and C. A. Floudas. A global optimization method, αBB, for general twice-differentiable constrained NLPs: II. Implementation and computational results. *Computers & Chemical Engineering*, 22(9):1159–1179, 1998c.

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

C. Floudas. Global optimization in design and control of chemical process systems. *Journal of Process Control*, 10:125–134, 2001.

K. Hägglöf, P. Lindberg, and L. Svensson. Computing global minima to polynomial optimization problems using Gröbner bases. *Journal of Global Optimization*, 7(2):115:125, 1995.

M. Hirafuji and S. Hagan. A global optimization algorithm based on the process of evolution in complex biological systems. *Computers and Electronics in Agriculture*, 29:125–134, 2000.

P. Kesavan and P. Barton. Generalized branch-and-cut framework for mixed-integer nonlinear optimization problems. *Computers and Chemical Engineering*, 24:1361–1366, 2000.

L. Liberti and C. Pantelides. Tightest convex envelopes of monomials of odd degree. *Journal of Global Optimization*, To appear.

G. McCormick. Computability of global solutions to factorable nonconvex programs: Part I — Convex underestimating problems. *Mathematical Programming*, 10:146–175, 1976.

P. Pardalos, H. Edwin Romeijin, and H. Tuy. Recent development and trends in global optimization. *Journal of Computational and Applied Mathematics*, 124:209–228, 2000.

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

H. Sherali. Tight relaxations for nonconvex optimization problems using the reformulation-linearization/convexification technique (RLT). 2:1–63, 2002.

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

H. Sherali and H. Wang. Global optimization of nonconvex factorable programming problems. *Mathematical Programming*, A89:459–478, 2001.

E. Smith. *On the Optimal Design of Continuous Processes.* PhD thesis, Imperial College of Science, Technology and Medicine, University of London, Oct. 1996.

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

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