# Validated Probing with Linear Relaxations

R. Baker Kearfott (rbk@louisiana.edu)

**Abstract.** During branch and bound search in deterministic global optimization, adaptive subdivision is used to produce subregions $\boldsymbol{x}$, which are then eliminated, shown to contain an optimal point, reduced in size, or further subdivided. The various techniques used to reduce or eliminate a subregion $\boldsymbol{x}$ determine the efficiency and practicality of the algorithm. Ryoo and Sahinidis have proposed a "probing" technique, involving the dual variables of a linear relaxation, to reduce the size of subregions $\boldsymbol{x}$. This technique, combined with others, has been successful in the BARON global optimization software.

Here, we show how the Ryoo and Sahinidis technique can be used in a validated context, without significant performance penalty. Our validation technique involves a mathematically rigorous regularization process and use of an interval Newton method on the Kuhn-Tucker conditions (complementarity conditions). This allows us to obtain rigorous bounds on dual variables.

We compare the process, implemented within our GlobSol environment, to an algorithm using LP relaxations but no "probing," on a standard test set. The results indicate that use of the "probing" technique does not significantly benefit the overall branch and bound process, although there is evidence that GlobSol's performance can depend crucially on the problem formulation and on values of heuristically set algorithm parameters. In any case, the regularization process we propose here, a relatively simple technique that results in a rigorous relaxation, is potentially of wider use in validated computations, where validated bounds on selected dual variables are desired.

**Keywords:** nonconvex optimization, global optimization, linear relaxation, probing, validated computing, interval analysis

## 1. Introduction

Our general global optimization problem can be stated as

$$\boxed{\begin{array}{l}\text{minimize } \varphi(x) \\ \text{subject to } c_i(x) = 0,\ i = 1, \ldots, m_1, \\ \qquad\qquad g_i(x) \le 0,\ i = 1, \ldots, m_2, \\ \text{where } \varphi : \boldsymbol{x} \to \mathbb{R} \text{ and } c_i, g_i : \boldsymbol{x} \to \mathbb{R}, \text{ and where} \\ \boldsymbol{x} \subset \mathbb{R}^n \text{ is the hyperrectangle (box) defined by} \\ \qquad\qquad \underline{x}_i \le x_i \le \overline{x}_i,\ 1 \le i \le n, \\ \text{where the } \underline{x}_i \text{ and } \overline{x}_i \text{ are constant bounds.}\end{array}} \tag{1}$$

We will call this problem a *general nonlinear programming problem*, abbreviated "general NLP" or "NLP".

Perhaps beginning with Falk and Soland (Falk and Soland, 1969; Soland, 1971), deterministic branch and bound methods for global optimization have been studied extensively over the last several decades. For an introduction and further references, see, for example, (Tawarmalani and Sahinidis, 2002) or (Hansen, 1992). In these methods, an initial region $\boldsymbol{x}^{(0)}$ is adaptively subdivided into subregions $\boldsymbol{x}$ of the form in (1), while various techniques are used to reduce, eliminate, or subdivide $\boldsymbol{x}$. For a relatively early explanation of this common search scheme, see (Pardalos and Rosen, 1987). For more recent explanations in which convex underestimators are employed, see for example (Floudas, 2000), (Tawarmalani and Sahinidis, 2002). For explanations focusing on validation but restricted to traditional interval arithmetic-based techniques, see (Hansen, 1992) or (Kearfott, 1996, Ch. 5).

Sahinidis et al have implemented a particularly successful branch and bound method in their BARON software (Tawarmalani and Sahinidis, 2002). This software has compared favorably to other branch and bound schemes in recent benchmarking tests (Neumaier et al., 2004). One possible reason is the use of linear relaxations (Tawarmalani and Sahinidis, 2002), while another is "probing."

If exact arithmetic were used in the algorithms employed in the BARON package and if the solutions to the linear relaxations produced during the computation were exact, then BARON would give the exact global optimum. However, floating point arithmetic is used, and state-of-the-art LP solvers, although good, are fallible, so BARON can fail to give correct global optimum or global optimizers in certain cases. Nonetheless, it is our contention that the crucial algorithms that give BARON its efficiency and practicality can be modified to be validated (that is, to rigorously take account of roundoff errors and algorithmic errors in the approximate problems), thus enabling software that is competitive with BARON in its present form but gives results that are correct with mathematical rigor. In (Kearfott, 2004), we studied a rigorous implementation of linear relaxations. In this paper, we study how probing can be implemented in a validated way, as well as present empirical results with validated probing.

### 1.1. Linear Relaxations

Convex relaxations go back to McCormick (McCormick, 1976) or before, while Tawarmalani and Sahinidis (tawarmalani and Sahinidis, 2004) introduced linear relaxations.

DEFINITION 1. *A linear relaxation of problem (1) is a linear program of the form*

$$\begin{array}{l}\text{minimize } a_{\varphi}^T x + b_{\varphi} \\ \text{subject to } a_{\underline{c}_i}^T x \le b_{\underline{c}_i},\ i = 1, \ldots, m_1, \\ \qquad\qquad a_{\overline{c}_i}^T x \le b_{\overline{c}_i},\ i = 1, \ldots, m_1, \\ \qquad\qquad a_{g_i}^T x \le b_{g_i},\ i = 1, \ldots, m_2, \\ \qquad\qquad \underline{x}_i \le x_i \le \overline{x}_i,\ 1 \le i \le n, \\ \text{where } a_{\varphi}^T x + b_{\varphi} \le \varphi(x),\ a_{\underline{c}_i}^T x - b_{\underline{c}_i} \le c_i(x), \\ \qquad a_{\overline{c}_i}^T x - b_{\overline{c}_i} \le -c_i(x), \text{ and } a_{g_i}^T x - b_{g_i} \le g_i(x), \\ \text{for } x \in \boldsymbol{x}.\end{array} \tag{2}$$

Thus, any optimum of a linear relaxation (2) of (1) is less than or equal to an optimum of the original nonlinear program (1). Furthermore, through various subdivision processes, and provided the widths of particular components of $\boldsymbol{x}$ are sufficiently small, linear relaxations can be constructed that approximate the original problem (1) arbitrarily closely. For our view of this process, see (Kearfott and Hongthong, 2003).

## 1.2. On Probing

Ryoo and Sahinidis introduced probing in (Ryoo and Sahinidis, 1995), with further explanation in (Tawarmalani and Sahinidis, 2002). The following theorem summarizes Test 1 and Test 2 in (Ryoo and Sahinidis, 1995):

THEOREM 1. *(a simple consequence of theorems in (*Ryoo and Sahinidis, 1995*)) Suppose $L$ is any underestimate for the global optimum of the original nonlinear programming problem (1) (for example, $L$ can be a rigorous lower bound on the solution to the relaxation (2)), suppose $U$ is a known upper bound for a solution to the original problem (1). Both of the following are true:*

1. *Assume that, for some $i$, the solution of the relaxed linear problem (2) corresponds to an active lower bound constraint $\underline{x}_i - x_i \le 0$, suppose the corresponding Lagrange multiplier at the solution is $\underline{y}_i > 0$, and define*

$$x_i^{(\ell)} = \overline{x}_i - (U - L)/\underline{y}_i.$$

*If $\underline{x}_i < x_i^{(\ell)}$, then all global optimizers in the original nonlinear problem (1) are also optimizers of the problem obtained by replacing $[\underline{x}_i, \overline{x}_i]$ by $[x_i^{(\ell)}, \overline{x}_i]$.*

2. *Assume that, for some $i$, the solution of the relaxed linear problem (2) corresponds to an active upper bound constraint $x_i - \overline{x}_i \leq 0$, suppose the corresponding Lagrange multiplier at the solution is $\overline{y}_i < 0$, and define*

$$x_i^{(u)} = \underline{x}_i - (U - L)/\overline{y}_i.$$

*If $\overline{x}_i > x_i^{(u)}$, then all global optimizers in the original nonlinear problem (1) are also optimizers of the problem obtained by replacing $[\underline{x}_i, \overline{x}_i]$ by $[\underline{x}_i, x_i^{(u)}]$.*

Sahinidis et al have implemented the tests in Theorem 1 with success in the BARON software package, using CPLEX (ILOG, 2003) to solve the linear programming problems and find the dual variables $\underline{y}_i$ and $\overline{y}_i$. However, these results are not at present validated in BARON. Neumaier and Shcherbina (Neumaier and Shcherbina, 2004), as well as Jansson (Jansson, 2004) have shown how to validate the optimum to the relaxation (2), given approximate values of the dual variables, but validation of the domain narrowing processes in Theorem 1 requires validated values of the dual variable themselves. We show how to do that in this work.

Ryoo and Sahinidis have two additional related procedures, Test 3 and Test 4 of (Ryoo and Sahinidis, 1995), to use the dual variables for cases when bound constraints are not active at the solution. This is what Ryoo and Sahinidis called "probing," since those procedures involve artificially setting variables on their bounds and computing corresponding dual variables. Although these procedures are related to the procedure we explain in Theorem 1, we have so far only constructed validated procedures for the case when bound constraints are naturally active at their solutions. Abusing the original terminology somewhat, we nonetheless refer to validated procedures based on Theorem 1 as *validated probing*.

### 1.3. Interval Newton Methods

Theory of interval Newton methods appears in various places, including (Neumaier, 1990) and in our work (Kearfott, 1996, §1.5 and §6.2.2). Suppose $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$. Then an interval Newton operator is of the form

$$\tilde{\boldsymbol{u}} = \boldsymbol{N}(f; \boldsymbol{u}, \check{u}) = \check{u} + \boldsymbol{v}, \tag{3}$$

where $\boldsymbol{v}$ is an interval vector that contains all solutions $v$ to point systems $Jv = -f(\check{u})$, for $J \in \boldsymbol{f'}(\boldsymbol{u})$, where $\boldsymbol{f'}(\boldsymbol{u})$ is an interval extension

to the Jacobi matrix[1] of $f$ over $\boldsymbol{u}$. Then $\boldsymbol{N}(f; \boldsymbol{u}, \check{u}) \subset \boldsymbol{u}$ implies there exists a unique solution to $f(u) = 0$ within $\boldsymbol{u}$ that must lie within $\tilde{\boldsymbol{u}}$.

DEFINITION 2. *If an interval Newton method as above uses a Lipschitz set as defined in (*Neumaier, 1990*) for the interval derivative matrix[2], then we say the interval Newton method is* existence and uniqueness validating*, or simply a "validating interval Newton method." If we apply an existence and uniqueness validating interval Newton method, and $\tilde{\boldsymbol{u}} \subset \boldsymbol{u}$, we say that the interval Newton method has validated existence and uniqueness for $f$ in $\boldsymbol{u}$.*

In fact, examining basic properties of interval arithmetic, we see that it is not necessary for each component $\tilde{\boldsymbol{u}}_i \subset \boldsymbol{u}_i$ if we are only interested in bounds on some components, as we clarify here:

THEOREM 2. *Suppose we partition the coordinate indices $\{i\}_{i=1}^n$ into two sets $\mathcal{I}$ and $\mathcal{I}_\neg$, and suppose an existence and uniqueness validating interval Newton method is applied, with the result that $\tilde{\boldsymbol{u}}_i \subset \boldsymbol{u}_i$ for each $i \in \mathcal{I}$. Then, for each set of coordinates $u_j \in \boldsymbol{u}_j$, $j \in \mathcal{I}_\neg$, there is a unique set of coordinates $u_i \in \tilde{\boldsymbol{u}}_i$, $i \in \mathcal{I}$ such that $f(u) = 0$.*

Theorem 2 can be useful when the validating interval Newton method is an interval Gauss–Seidel method, in which $\tilde{\boldsymbol{u}}$ is computed componentwise, and in our context, where $\mathcal{I}$ will contain indices corresponding only to those dual variables $y_i$ belonging to bound constraints thought to be active at optimality. We clarify this in §2 below.

### 1.4. Expanded Systems

In practice, we apply linear relaxations to an equivalent derived system in which each constraint and the objective contain at most one arithmetic operation or standard function evaluation[3]. We obtain this system through a common process, such as we illustrate in (Kearfott, 1996, Ch. 7). We give a further example of this process in §3 below.

In addition to using a decomposition into elementary operations, we simplify further into an "equivalent relaxed expanded NLP," as we have explained in (Kearfott and Hongthong, 2003).

Conceptually, the validation processes we describe in this work would apply equally well to the original system (1) and to the equivalent

[1] In some contexts, we can use an interval slope matrix. Here, however, we use an interval Jacobi matrix for simplicity.

[2] A common component-wise interval extension of the Jacobi matrix for the system provides such a Lipschitz set.

[3] The derived system and the processes we use to obtain it in not unique to our own work, not even in this context. For example, many of the developments in (Tawarmalani and Sahinidis, 2002) are based on such a derived system.

relaxed expanded NLP. However, in practice (and in the illustrative example in §3), we apply these processes to the equivalent relaxed expanded system. In any case, the equivalent relaxed expanded NLP has an extremely sparse constraint matrix (with at most three nonzero entries per row), and this sparsity should be exploited in any implementation.

## 2. Validating the Dual Variables

We observe first that, in part 1 of Theorem 1, to avoid losing global optimizers, we may underestimate $x_i^{(\ell)}$. Also, since $\underline{y}_i > 0$ and $-(U - L) < 0$, underestimating $\underline{y}_i$ decreases $x_i^{(\ell)}$, so, in the context of part 1 of Theorem 1, we need to find validated lower bounds for such dual variables $\underline{y}_i$. Similarly, in the context of part 2 of Theorem 1, we need to find validated upper bounds on dual variables $\overline{y}_i$ to ensure that no optima are lost due to roundoff.

To see how we can bound the dual variables $\underline{y}_i$ or $\overline{y}_i$, we rewrite the relaxed problem (2) as

$$\boxed{\begin{array}{l}\text{minimize } d^T x \\ \text{subject to } Ax \le b,\end{array}} \tag{4}$$

where $A$ is an $m$ by $n$ matrix, with $m \ge n$. The dual problem to (4) is now

$$\boxed{\begin{array}{l}\text{maximize } b^T y \\ \text{subject to } A^T y = d, \\ \qquad\qquad y \le 0.\end{array}} \tag{5}$$

With this notation, validated use of Theorem 1 requires lower and upper bounds on the dual variables $y$. We will use the complementarity conditions relating the primal (4) and dual (5) to provide validated lower and upper bounds on the system. In particular, forming the Kuhn-Tucker (Lagrange multiplier) system corresponding to the primal (4), we obtain the system

$$\begin{array}{rcll} A^T y - d & = & 0 & \\ y_i \left(A_{i,:} x - b_i\right) & = & 0, & 1 \le i \le m, \end{array} \tag{6}$$

whose Jacobi matrix is of the form

$$H(x,y) = \left(\begin{array}{c|cc} 0 & \multicolumn{2}{c}{A^T} \\ \hline y_1 A_{1,:} & A_{1,:}x - b_1 & 0 \\ \vdots & & \ddots \\ y_m A_{m,:} & 0 & A_{m,:}x - b_m \end{array}\right). \tag{7}$$

The function $f : \mathbb{R}^{m+n} \to \mathbb{R}^{m+n}$ to be used in the interval Newton method for validation will have its first $n$ components equal to the $n$ components of $A^T y - d$ and its next $m$ components equal to $y_i\,(A_{i,:}x - b_i)$, $1 \leq i \leq m$. However, with the methods we use to form the relaxation (2) of (1), the Jacobi matrix $H$ is typically singular, precluding usual application of an interval Newton method. We now explain how we modify the system (6) to obtain a suitable non-singular system.

It is well-known (and can be seen from the second set of $m$ equations in (6)), the Lagrange multipliers $y_i$ and the residuals obey a complementarity condition; in particular, if the $i$-th constraint $A_{i,:}x \leq b_i$ is inactive at a solution, (i.e. if strict inequality holds at the optimum of (4)) then the corresponding dual variable (Lagrange multiplier) $y_i$ is equal to zero. Examining the Jacobi matrix (7) reveals, however, that, if there are redundant constraints, such as when we replace an equality constraint by two inequality constraints, there may be $i$ for which both $y_i = 0$ and $A_{i,:}x - b_i = 0$, in which case $H$ will be singular, and the interval Newton method will fail to validate. Also, in such cases, the matrix $A^T$ will not be of full rank, ensuring that $H$ will be singular and usual validation with interval Newton methods will not be possible.

Indeed, typically, $m > n$, and $m - n$ of the constraints are inactive at the solution, while only the dual variables $\hat{y}$ corresponding to the $n$ active constraints are nonzero. If we could correctly identify these from approximately computed dual values, then we could simply form an $n \times n$ system of equations $\hat{A}^T\hat{y} = d$ by omitting those rows of $A$ corresponding to inactive constraints, and use the system $\hat{A}^T\hat{y} = d$ directly in a validating interval Newton method. However, we cannot be certain that constraints we identify as inactive from dual variables we identify as only being approximately equal to zero are actually zero.

However, if we form a related optimization problem by omitting *any* $m - n$ constraints, the resulting problem is also a relaxation (since the feasible set of the derived problem contains the feasible set of the original problem). Thus, we may omit any $m - n$ rows of $A$ in the Kuhn-Tucker system (6), then use the resulting reduced system in a validating interval Newton method to obtain lower and upper bounds $\underline{y}_i$ and $\overline{y}_i$ on dual variables corresponding to active bound constraints; we may then use the bounds $\underline{y}_i$ and $\overline{y}_i$, so obtained in Theorem 1, to rigorously narrow the uncertainty interval on components $x_i$ to the solution of our original nonlinear program (1), regardless of whether we have correctly identified the inactive constraints from the approximate solution.

Although use of the reduced system leads to rigorous results even if we have not correctly identified the inactive constraints, if we have

correctly identified our inactive constraints from the approximate solution, then we can expect the validating interval Newton method to succeed in providing reasonable lower and upper bounds on the dual variables. First, the reduced linear relaxation will then be equivalent to the original linear relaxation. Second, examining (7) in that case, we see, if we construct bounds $\boldsymbol{x}$ and $\boldsymbol{y}$ about an approximate solution to the reduced problem, the corresponding interval extension of the Jacobi matrix is a $2n$ by $2n$ matrix that has approximately the form

$$\boldsymbol{H}(\boldsymbol{x},\boldsymbol{y}) \approx \left( \begin{array}{c|c} 0 & A^T \\ \hline \boldsymbol{y}_1 A_{1,:} & \\ \vdots & 0 \\ \boldsymbol{y}_n A_{n,:} & \end{array} \right).$$

If $A \in \mathbb{R}^{n \times n}$ is nonsingular (i.e. if the active constraints are independent) and the $\boldsymbol{y}_i$ are sufficiently narrow and nonzero, then $\boldsymbol{H}(\boldsymbol{x},\boldsymbol{y})$ will be non-singular.

DEFINITION 3. *Take any linear relaxation of the form (4), suppose we have computed approximate dual variables* $\{y_i\}_{i=1}^m$*, and, using any criterion, we have identified* $m-n$ *of these* $y_i$ *to correspond to inactive constraints*[4]*. Suppose we modify the original problem by omitting* $m-n$ *such constraints, and also omitting any variables that occur only in those constraints we have omitted*[5]*. We will refer to any problem so obtained as a* regularized linear relaxation*.*

The regularized linear relaxation is unbounded if there is a variable that appears with a non-zero coefficient that does not occur in any constraints (explicit or bound constraints).

## 3. An Illustrative Example

We exhibit:

1. the decomposition into elementary operations,
2. the equivalent relaxed expanded NLP,

[4] Two ways of identifying inactive constraints are to judge $|y_i|$ to be small or to judge $A_{i,:}x - b_i$ to be strictly negative.

[5] If, after omitting constraints which are inactive, the resulting problem has a variable that occurs only in the objective, and not in the remaining constraints, this is an indication that the original problem is unbounded. In such cases, our validation process for the dual variables is not expected to succeed.

3. the original linear relaxation,

4. approximate primal and dual variables,

5. a regularized linear relaxation,

6. the result of applying an interval Newton method to the regularized linear relaxation,

7. and the result of applying the technique in Theorem 1,

for the following.

EXAMPLE 1.

$$\begin{array}{l} \textit{minimize } x_1 + x_2 \\ \textit{subject to } x_1 - x_2 - 1 \leq 0, \\ \qquad\qquad 2x_1 - 2x_2 - 2 \leq 0, \\ \qquad\qquad x_1 \geq 0. \end{array} \tag{8}$$

Decomposing[6] into elementary operations using techniques with the "code list," a particular compiler gives the following (non-unique) decomposition on the left, then techniques from (Kearfott and Hongthong, 2003) give the following equivalent relaxed expanded NLP on the right:

$$\begin{array}{|l|} \hline \text{minimize } x_3 \\ \hline \text{subject to } x_3 = x_1 + x_2 \\ \qquad\qquad x_4 = x_1 - x_2 \\ \qquad\qquad x_5 = x_4 - 1 \\ \qquad\qquad x_6 = 2x_1 \\ \qquad\qquad x_7 = 2x_2 \\ \qquad\qquad x_8 = x_6 - x_7 \\ \qquad\qquad x_9 = x_8 - 2 \\ \hline \qquad\qquad x_5 \leq 0 \\ \qquad\qquad x_9 \leq 0 \\ \hline \qquad\qquad x_1 \geq 0 \\ \hline \end{array} \qquad \begin{array}{|l|} \hline \text{minimize } x_3 \\ \hline \text{subject to } x_3 \leq x_1 + x_2 \\ \qquad\qquad x_4 \leq x_1 - x_2 \\ \qquad\qquad x_5 \leq x_4 - 1 \\ \qquad\qquad x_6 \leq 2x_1 \\ \qquad\qquad x_7 \geq 2x_2 \\ \qquad\qquad x_8 \leq x_6 - x_7 \\ \qquad\qquad x_9 \leq x_8 - 2 \\ \hline \qquad\qquad x_5 \leq 0 \\ \qquad\qquad x_9 \leq 0 \\ \hline \qquad\qquad x_1 \geq 0 \\ \hline \end{array} \tag{9}$$

Since the original problem was linear, equivalent relaxed expanded NLP[7] is the already a linear relaxation. When we place this linear

[6] a process we review in (Kearfott, 1996), and we further explain in this context in (Kearfott and Hongthong, 2003).

[7] formed as explained in (Tawarmalani and Sahinidis, 2002) and in our works (Hongthong and Kearfott, 2004) and (Kearfott and Hongthong, 2003)

relaxation into the form (4), we obtain

$$d = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}^T,$$

$$A = \begin{pmatrix} 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{10}$$

$$b = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \end{pmatrix}^T.$$

We used the SLATEC routine DSPLP to find approximate primal and dual solutions to (4) and (5) with $A$, $b$, and $d$ as in (10), obtaining approximate primal and dual values:

$$x \approx \begin{pmatrix} 0 \\ -0.1 \times 10^{1} \\ -0.1 \times 10^{1} \\ 0.1 \times 10^{1} \\ 0 \\ -0.2 \times 10^{-307} \\ -0.2 \times 10^{1} \\ 0.2 \times 10^{1} \\ -0.2 \times 10^{-13} \end{pmatrix}, \quad y \approx \begin{pmatrix} -0.1 \times 10^{1} \\ -0.1 \times 10^{1} \\ -0.1 \times 10^{1} \\ 0 \\ 0 \\ 0 \\ 0 \\ -0.1 \times 10^{1} \\ 0 \\ -0.2 \times 10^{1} \end{pmatrix}, \quad Ax - b \approx \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

In this unusual singular problem, all of the constraints appear to be active at the approximate solution. However, examining the approximate dual variables $y$, we surmise that constraints 1, 2, 3, and 8 are active, as well as constraint 10 (corresponding to the bound constraint on $x_1$), while constraints 4, 5, 6, 7, and 9 are can be considered to be inactive. Using this criterion, we eliminate constraints 4, 5, 6, 7, and 9, and observe that, now, only variables 1, 2, 3, 4, and 5 occur. We thus obtain the regularized linear relaxation

$$\boxed{\begin{array}{l} \text{minimize } \tilde{d}^T x \\ \text{subject to } \tilde{A}x \leq \tilde{b}, \end{array}} \tag{11}$$

where

$$\tilde{d} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \end{pmatrix}^T,$$

$$\tilde{A} = \begin{pmatrix} 1 & 1 & -1 & 0 & 0 \\ 1 & -1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad \text{and} \tag{12}$$

$$\tilde{b} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \end{pmatrix}^T.$$

If we are lucky, then this regularized linear relaxation will be equivalent to the original linear relaxation (4). However, regardless of how we choose which constraints to delete (and hence, regardless of how accurate the approximate solution to the linear program (4) is), the regularized relaxed problem (11) is a relaxation of the original problem, and a validated solution to it will provide rigorous lower and upper bounds on the Lagrange multipliers for Theorem 1. Furthermore, if the constraints that remain correspond to non-zero dual variables, it is likely that the interval Newton method will succeed in validating upper and lower bounds for them.

Actually, when we use DSPLP to approximately solve the regularized problem with objective and constraints given by (12), we obtain approximate values $x_{\text{regularized}}$ for the primal variables and $y_{\text{regularized}}$ for the dual variables of:

$$x_{\text{regularized}} \approx \begin{pmatrix} 0 \\ -0.1 \times 10^1 \\ -0.1 \times 10^1 \\ 0.1 \times 10^1 \\ 0 \end{pmatrix}, \quad y_{\text{regularized}} \approx \begin{pmatrix} -0.1 \times 10^1 \\ -0.1 \times 10^1 \\ -0.1 \times 10^1 \\ -0.1 \times 10^1 \\ -0.2 \times 10^1 \end{pmatrix}. \tag{13}$$

Although it is not crucial for the validation step, in this case the approximate solution to the regularized relaxed problem corresponds to the approximate solution to the original linear relaxation, and the regularized relaxed problem is probably equivalent to the original linear relaxation.

Plugging in the computed approximations $x_{\text{regularized}}$ and $y_{\text{regularized}}$, the Jacobi matrix $H(x, y)$ as in (7) for the regularized system is ap-

proximately:

$$H_{\text{regularized}} \approx \left(\begin{array}{ccccc|ccccc} & & & & & 1 & 1 & 0 & 0 & -1 \\ & & & & & 1 & -1 & 0 & 0 & 0 \\ & & 0 & & & -1 & 0 & 0 & 0 & 0 \\ & & & & & 0 & -1 & 1 & 0 & 0 \\ & & & & & 0 & 0 & -1 & 1 & 0 \\ \hline -1 & -1 & 1 & 0 & 0 & & & & & \\ -1 & 1 & 0 & 1 & 0 & & & & & \\ 0 & 0 & 0 & -1 & 1 & & \text{diag}(\approx 0) & & & \\ 0 & 0 & 0 & 0 & -1 & & & & & \\ -2 & 0 & 0 & 0 & 0 & & & & & \end{array}\right), \quad (14)$$

and the condition number $\kappa(H_{\text{regularized}}) \approx 9.6$, a favorable value when applying an interval Newton method.

In this example, although small, one sees the advisability of implementing an interval Newton method that exploits sparsity. We implemented an existence and uniqueness validating interval Newton method based on computing and storing the inverse midpoint preconditioner one row at a time[8], and applying each preconditioner row as it is computed in a step of an interval Gauss–Seidel method.

In this example, the only coordinate of interest is the last one, corresponding to the dual variable $\tilde{y}_5 = u_{10}$ for the bound constraint $x_1 > 0$ (and $x_1$ is the only component for which this particular regularized relaxed system can be applied). Thus, for Theorem 2, we may have $\mathcal{I} = 10$, and the only condition that needs to be satisfied to obtain valid lower and upper bounds on the dual variable for $x_1$ is $\tilde{\boldsymbol{u}}_{10} \subset \boldsymbol{u}_{10}$.

To apply our interval Newton method to our example, we construct a box $\boldsymbol{u}$ with center at $\check{u} = ({x^T}_{\text{regularized}}, {y^T}_{\text{regularized}})^T$ and whose $i$-th coordinate width is set (heuristically) to $(1 + |\check{u}_i|)\sqrt{\epsilon_d}$, where $\epsilon_d = 10^{-10}$ is the tolerance to which we expect $\check{u}$ to be accurate. Using the regularized system, we do up to two sweeps of an interval Gauss–Seidel method, quitting if $\tilde{\boldsymbol{u}}_i \subset \boldsymbol{u}_i$ for every $i \in \mathcal{I}$. After a single sweep, we obtained a rigorous enclosure for the dual variable corresponding to the bound constraint[9]:

$$y_5 \in [\ -2.00000000000000267,\ -1.99999999999999733].$$

[8] We used the Harwell Fortran subroutines MA28AD and MA28CD for this purpose.

[9] and indeed, we obtained rigorous enclosures for all of the primal and dual variables of the regularized problem.

At this point, if these computations were embedded in an actual validated optimization code[10], we would use part 1 of Theorem 1, evaluating $\overline{x}_1 - (U - L)/\underline{y}_5$ with interval arithmetic (or else with upward directed rounding) to rigorously take account of roundoff error in this final computation.

## 4. On the Implementation

Care needs to be taken in the implementation to be assured of correctness. For example, in Theorem 1, in part 1, the Lagrange multiplier $y_i$ should be negative. It may so happen, however, that there is rank-deficiency in the system, and the exact dual variable $y_i$ for the solution to the regularized linear relaxation is zero. In that case, $\underline{y}_i < 0$, and $x_i^{(\ell)} > \overline{x}_i$ is an incorrect conclusion. Various such pitfalls are to be avoided in validated code.

If we apply the process to each sub-box $\tilde{\boldsymbol{x}}$ in the branch and bound search tree, then Theorem 1 dictates that $L$ and $U$ be lower and upper bounds over the sub-box $\tilde{\boldsymbol{x}}$, rather than the original box. The lower bound can be computed as a validated solution to the linear relaxation, while the upper bound can be obtained with a straightforward interval evaluation.

## 5. Tests Within a Validated Global Optimization Code

We implemented the validated probing process as described above within our GlobSol validated global optimization system. In particular, we used an experimental version of GlobSol, as we describe in (Kearfott, 2004), in which we have implemented rigorous linear relaxations. As in (Kearfott, 2004), we used the "tiny" problems from the Library 1 set in (Neumaier et al., 2004) as test problems. In contrast to (Kearfott, 2004), we compiled our experimental version of GlobSol with the NAG Fortran 95 compiler, release 5.0, without optimization, on a dual 2.8 GHz AMD Opteron processor[11] running Linux (SuSe distribution 9.1), with 4 gigabytes of memory. We applied the "probing" process described above to each box processed in the global search process, after applying constraint propagation and the linear relaxations (as

[10] For this simple example, the original optimization problem is linear, and $\underline{x}_1$ and $\overline{x}_1$ are already known fairly accurately.

[11] The actual computations were not done in parallel, but the system load was such that, at all times. the GlobSol program had total resources of at least one processor.

described in (Kearfott, 2004), but before applying the interval Newton method.

In an initial experiment, we tried example `ex14_1_1` from (Neumaier et al., 2004), an example for which linear relaxations had a decisive effect on performance. In this problem, which could complete with 59 boxes total processed, the probing procedure was called 106 times; in 85 of those calls, the probing procedure encountered no active bound constraints in the reduced problem (and hence no relevant Lagrange multipliers could be computed), there were no explicit constraints in one problem[12], and the process failed to reduce any coordinate widths in any of the other times the probing process was applied.

The procedure could be of more utility in other problems, in which bound constraints tend to be active more often. To see this, we applied the technique to all of the "tiny" problems from the Library 1 test set for which we had implemented validated linear relaxations all necessary operations[13]. We allowed a maximum of 100,000 boxes or 7200 seconds of CPU time for completion in all cases, and we marked cases that could not complete in that time as "not OK". The results are in Table I.

In no case did the probing process make a difference in whether or not the process successfully completed within the given resource limits. Thus, we list only one column, labeled "OK" to show whether or not the branch and bound process completed. The remaining columns are as follows:

*Column 3, labeled "#B-w,"* gives the number of boxes processed with the probing process.

*Column 4, labeled "#B-wo,"* gives the number of boxes processed without the probing process.

*Column 5, labeled "CPU-w,"* gives the total processor time when the probing process is used.

*Column 6, labeled "CPU-wo,"* gives the total processor time when the probing process is not used.

*Column 7, labeled "CPU wo/w,"* gives the ratio of processor times without to with probing.

*Column 8, labeled "B wo/w,"* gives the ratio of total number of boxes processed without the probing process to with the probing process.

[12] The SLATEC routine DSPLP, the linear programming solver we used, could only handle cases with explicit constraints

[13] We had not yet implemented linear relaxations for the trigonometric functions and quotients.

Table I. Results with and without probing

| Problem | OK? | #B-w | #B-wo | CPU-w | CPU-wo | CPU wo/w | B wo/w | #succ. |
|---|---|---|---|---|---|---|---|---|
| dispatch | Yes | 13 | 13 | 0.51 | 0.51 | 1.0 | 1.0 | 0 |
| ex14_1_1 | Yes | 1775 | 1791 | 100.98 | 102.75 | 1.0 | 1.0 | 6 |
| ex14_1_2 | No | 67172 | 68743 | 7200.56 | 7200.59 | 1.0 | 1.0 | 0 |
| ex14_1_3 | Yes | 564 | 564 | 5.98 | 4.76 | 0.8 | 1.0 | 0 |
| ex14_1_5 | Yes | 100 | 100 | 3.23 | 3.1 | 1.0 | 1.0 | 0 |
| ex14_1_9 | Yes | 102 | 102 | 0.92 | 0.87 | 0.9 | 1.0 | 0 |
| ex14_2_1 | No | 54005 | 54899 | 7200.37 | 7200.45 | 1.0 | 1.0 | 0 |
| ex14_2_2 | Yes | 2220 | 2220 | 121.67 | 117.02 | 1.0 | 1.0 | 0 |
| ex14_2_3 | No | 44121 | 45516 | 7201.15 | 7201.29 | 1.0 | 1.0 | 0 |
| ex14_2_5 | Yes | 1890 | 1890 | 107.74 | 103.61 | 1.0 | 1.0 | 0 |
| ex2_1_1 | Yes | 234 | 234 | 1.25 | 1.21 | 1.0 | 1.0 | 0 |
| ex2_1_2 | Yes | 173 | 173 | 0.8 | 0.77 | 1.0 | 1.0 | 0 |
| ex2_1_4 | Yes | 222 | 222 | 2.93 | 2.75 | 0.9 | 1.0 | 0 |
| ex3_1_1 | No | 94039 | 100000 | 7204.15 | 6829.19 | 0.9 | 1.1 | 1 |
| ex3_1_2 | Yes | 78 | 78 | 0.52 | 0.51 | 1.0 | 1.0 | 0 |
| ex3_1_3 | Yes | 253 | 253 | 0.84 | 0.82 | 1.0 | 1.0 | 0 |
| ex3_1_4 | Yes | 37 | 37 | 0.56 | 0.49 | 0.9 | 1.0 | 0 |
| ex4_1_2 | Yes | 6 | 6 | 0.43 | 0.35 | 0.8 | 1.0 | 0 |
| ex4_1_4 | Yes | 7 | 7 | 0.01 | 0.01 | 1.0 | 1.0 | 0 |
| ex4_1_5 | Yes | 39 | 39 | 0.11 | 0.1 | 0.9 | 1.0 | 0 |
| ex4_1_6 | Yes | 5 | 5 | 0.02 | 0.02 | 1.0 | 1.0 | 0 |
| ex4_1_7 | Yes | 4 | 4 | 0.01 | 0.01 | 1.0 | 1.0 | 0 |
| ex4_1_8 | Yes | 5 | 5 | 0.01 | 0.01 | 1.0 | 1.0 | 0 |
| ex4_1_9 | Yes | 38 | 38 | 0.16 | 0.14 | 0.9 | 1.0 | 0 |
| ex5_4_2 | Yes | 511 | 511 | 25.8 | 23.46 | 0.9 | 1.0 | 1 |
| ex6_1_2 | Yes | 122 | 122 | 2.76 | 2.7 | 1.0 | 1.0 | 0 |
| ex7_2_1 | No | 14264 | 14507 | 7201.44 | 7201.15 | 1.0 | 1.0 | 0 |
| ex7_2_5 | Yes | 153 | 153 | 3.47 | 3.45 | 1.0 | 1.0 | 0 |
| ex7_2_6 | Yes | 36 | 36 | 0.21 | 0.17 | 0.8 | 1.0 | 1 |
| ex7_3_1 | Yes | 97 | 97 | 7.73 | 7.59 | 1.0 | 1.0 | 0 |
| ex7_3_3 | Yes | 55 | 55 | 1.8 | 1.78 | 1.0 | 1.0 | 0 |
| ex8_1_3 | No | 100000 | 100000 | 2719.08 | 2464.62 | 0.9 | 1.0 | 0 |
| ex8_1_4 | Yes | 28 | 28 | 0.11 | 0.11 | 1.0 | 1.0 | 0 |
| ex8_1_5 | Yes | 131 | 131 | 0.93 | 0.89 | 1.0 | 1.0 | 0 |
| ex8_1_6 | Yes | 36 | 36 | 0.44 | 0.37 | 0.8 | 1.0 | 0 |
| ex8_1_7 | No | 100000 | 100000 | 4394.23 | 3951.33 | 0.9 | 1.0 | 0 |
| ex9_2_4 | No | 56105 | 100000 | 7284.87 | 4680.5 | 0.6 | 1.8 | 2011 |
| ex9_2_5 | No | 79322 | 80311 | 7212.43 | 7212.31 | 1.0 | 1.0 | 0 |
| ex9_2_8 | Yes | 8 | 8 | 0.02 | 0.02 | 1.0 | 1.0 | 0 |
| house | No | 59257 | 59273 | 7204.24 | 7204.24 | 1.0 | 1.0 | 0 |
| least | Yes | 1440 | 1440 | 69.46 | 66.39 | 1.0 | 1.0 | 0 |
| mhw4d | Yes | 240 | 240 | 4.81 | 4.77 | 1.0 | 1.0 | 0 |
| nemhaus | Yes | 0 | 0 | 0 | 0 | – | – | 0 |
| rbrock | Yes | 4 | 4 | 0 | 0 | – | 1.0 | 0 |
| sample | Yes | 27734 | 43 | 535.77 | 3.79 | 0.0 | 0.0 | 0 |
| wall | Yes | 117 | 117 | 4.55 | 4.51 | 1.0 | 1.0 | 0 |

*Column 9, labelled "#succ.,"* gives the total number of times a coordinate bound was reduced in the probing process.

As can be seen from Table I, the actual probing process had very little effect on the overall algorithm, and indeed, only changed coordinate bounds in problems `ex3_1_1`, `ex5_4_2`, `ex7_2_6`, and `ex9_2_4`.

In `ex9_2_4`, the branch and bound process did not complete without probing without processing more than 100,000 boxes, whereas, with the

process, the branch and bound could not complete within 7200 seconds of processor time. That example is:

$$\begin{array}{l}\min\{(0.5x_5-1)(x_5-2)+(0.5x_6-1)(x_6-2)\}\\ \text{subject to:}\\ \quad -x_4+x_7+1 = 0, \quad -x_3+x_5+x_7 = 0, \quad x_2-x_6 = 0,\\ \quad x_1-x_5 = 0, \quad x_5+x_6-x_8 = 0, \quad x_2x_4 = 0,\\ \quad x_1x_3 = 0.\end{array}$$

Unverified symbolic computation (with Mathematica) reveals a unique isolated global minimum of 0 at $x_5 = 1$, $x_6 = 2$, but GlobSol failed to validate a single feasible point, and thus failed to obtain a validated upper bound for a global optimum. We traced the failure for this problem to a heuristic parameter $\alpha$ in the global search that determined when computation of an approximate feasible point or optimizer was attempted[14]. Changing this parameter so an approximate optimizer was sought in every box processed in the branch and bound algorithm and changing the problem to not include bound constraints with GlobSol's "peeling" process[15] , we obtained this unique solution as follows:

| | # boxes | CPU | # probe succ. |
|---|---|---|---|
| without "probing" | 45 | 2.1 | — |
| with "probing" | 45 | 2.0 | 0 |

Thus, "probing" as we have implemented it does not even appear to contribute to the success of the overall algorithm, even for `ex9_2_4`. However, examination of the results of this test set provides evidence that the validated "probing" process as we implemented it is rigorous.

One possible reason "probing" was not effective is that we applied it after using the original linear relaxation directly[16], replacing the objective by $x_i$ and $-x_i$ to compute new lower and upper bounds on each variable $x_i$. This direct recomputation uses the same model of the original problem as probing, and may be related mathematically to probing.

[14] In this test set, this difficulty was unique to problem `ex9_2_4`.

[15] We obtained similar results, with 229 boxes processed rather than 45, when the initial coordinate bounds were handled as bound constraints with "peeling".

[16] as explained in (Tawarmalani and Sahinidis, 2002), etc.

## 6. Summary and Future Work

We have described a simple technique to further relax a linear relaxation of a nonlinear program, in such a way that validated bounds on dual variables of the second relaxation can often be computed. We illustrated this technique with a simple example. We implemented the technique within the branch and bound algorithm in our GlobSol software, and tried the algorithm on a standard test set. The results revealed a sensitivity of GlobSol's ability to complete efficiently, in at least one problem, to the setting of a heuristic algorithm parameter, but also revealed no advantage to doing the "probing" within our environment.

Future work may involve trying the "true" probing procedures as described in Test 3 and Test 4 of (Ryoo and Sahinidis, 1995), although we first need to understand how Test 3 and Test 4 work in a validated context. We may also wish to further clarify the mathematical relationship between the "probing" process and computing new lower and upper bounds on the variables directly from the original linear relaxation. Finally, further study of heuristic parameter settings, for particular problems, within GlobSol is probably useful.

## References

Falk, J. E. and R. M. Soland: 1969, 'An Algorithm for Separable Nonconvex Programming Problems'. *Manage. Sci.* **11**, 287–311.

Floudas, C. A.: 2000, *Deterministic Global Optimization: Theory, Algorithms and Applications*. Dordrecht, Netherlands: Kluwer.

Hansen, E. R.: 1992, *Global Optimization Using Interval Analysis*. New York: Marcel Dekker, Inc.

Hongthong, S. and R. B. Kearfott: 2004, 'Rigorous Linear Overestimators and Underestimators'. preprint, `http://interval.louisiana.edu/preprints/estimates_of_powers.pdf`.

ILOG: 2003, 'Cplex 9.0 User Guide'. `http://www.gams.com/solvers/cplex.pdf`.

Jansson, C.: 2004, 'A Rigorous Lower Bound for the Optimal Value of Convex Optimization Problems'. *J. Global Optim.* **28**(1), 121–137.

Kearfott, R. B.: 1996, *Rigorous Global Search: Continuous Problems*. Dordrecht, Netherlands: Kluwer.

Kearfott, R. B.: 2004, 'Empirical Comparisons of Linear Relaxations and Alternate Techniques in Validated Deterministic Global Optimization'. preprint, `http://interval.louisiana.edu/preprints/validated_global_optimization_search_comparisons.pdf`.

Kearfott, R. B. and S. Hongthong: 2003, 'A Preprocessing Heuristic for Determining the Difficulty of and Selecting a Solution Strategy for Nonconvex Optimization'. preprint, `http://interval.louisiana.edu/preprints/2003_symbolic_analysis_of_GO.pdf`.

McCormick, G. P.: 1976, 'Computability of Global Solutions to Factorable Nonconvex Programs'. *Math. Prog.* **10**(2), 147–175.

Neumaier, A.: 1990, *Interval Methods for Systems of Equations.* Cambridge, England: Cambridge University Press.

Neumaier, A. and O. Shcherbina: 2004, 'Safe Bounds in Linear and Mixed-Integer Programming'. *Math. Prog.* **99**(2), 283–296.
`http://www.mat.univie.ac.at/~neum/ms/mip.pdf`.

Neumaier, A., O. Shcherbina, W. Huyer, and T. Vink'o: 2004, 'A Comparison of Complete Global Optimization Solvers'. preprint,.

Pardalos, P. M. and J. B. Rosen: 1987, *Constrained Global Optimization: Algorithms and Applications*, Lecture Notes in Computer Science no. 268. New York: Springer-Verlag.

Ryoo, H. S. and N. V. Sahinidis: 1995, 'Global Optimization of Nonconvex NLPs and MINLPs with Applications in Process Design'. *Computers and Chemical Engineering* **19**(5), 551–566.

Soland, R. M.: 1971, 'An Algorithm for Separable Nonconvex Programming Problems II: Nonconvex Constraints'. *Manage. Sci.* **17**, 759–773.

Tawarmalani, M. and N. V. Sahinidis: 2002, *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, and Applications.* Dordrecht, Netherlands: Kluwer.

tawarmalani, M. and N. V. Sahinidis: 2004, 'Global Optimization of mMixed-Integer Nonlinear Programs: A Theoretical and Computational Study'. *Math. Prog.* **99**(3), 563–591.