Noname manuscript No.
(will be inserted by the editor)

On Smooth Reformulations and Direct Non-Smooth Computations for Minimax Problems

**Ralph Baker Kearfott, Sowmya Muniswamy,
Yi Wang, Xinyu Li and Qian Wang**

Received: date / Accepted: date

Abstract Minimax problems can be approached by reformulating them into smooth problems with constraints or by dealing with the non-smooth objective directly. We focus on verified enclosures of all globally optimal points of such problems. In smooth problems in branch and bound algorithms, interval Newton methods can be used to verify existence and uniqueness of solutions, to be used in eliminating regions containing such solutions, and point Newton methods can be used to obtain approximate solutions for good upper bounds on the global optimum. We analyze smooth reformulation approaches, show weaknesses in them, and compare reformulation to solving the non-smooth problem directly. In addition to analysis and illustrative problems, we exhibit the results of numerical computations on various test problems.

Keywords minimax, verified computations, Fritz John equations

1 Introduction

The nonlinear discrete minimax problem can be stated as

$$\min_x \max_{1\le i\le m} |f_i(x)|, \quad x \in \mathbb{R}^n, \quad m \ge n, \tag{1}$$

or, more generally, as

$$\min_x \max_{1\le i\le m} f_i(x), \quad x \in \mathbb{R}^n, \quad m \ge n, \tag{2}$$

Ralph Baker Kearfott
Department of Mathematics, University of Louisiana at Lafayette, U.L. Box 4-1010, Lafayette, LA 70504-1010 USA
Tel.: 337-482-5270
Fax: 337-482-5346
E-mail: rbk@louisiana.edu

Sowmya Muniswamye
Department of Mathematics, University of Louisiana at Lafayette

Xinyu Li
Department of Mathematics, University of Louisiana at Lafayette

Qian Wang
Department of Mathematics, University of Louisiana at Lafayette

where the functions f_i are, in general, smooth but nonlinear. This problem is of interest in data fitting, providing an ℓ_∞ fit, as an alternative to least squares (ℓ_2 fits) or least absolute value (ℓ_1 fits). Also important, the problem has been well-studied in an operations research setting, as a model to minimize the worst possible outcome, in the presence of uncertainty in uncontrollable aspects of the problem.

Various numerical methods have been proposed for finding points $\check{x}$ that approximate solutions x^* to this problem. For a review of these in a broader context see [20].

As with most traditional algorithms in nonlinear optimization and for the solution of nonlinear systems, there is no guarantee that the point $\check{x}$ that such algorithms output is close to an actual solution x^*. This leads to the following problem, whose solution can be useful in branch and bound algorithms for global optimization.

> Given $\check{x} \in \mathbb{R}^n$, *rigorously* verify:
>
> There is a small box
>
> $$\boldsymbol{x} = ([\underline{x}_1, \overline{x}_1], [\underline{x}_2, \overline{x}_2], \ldots, [\underline{x}_n, \overline{x}_n]), \tag{3}$$
>
> $\check{x} \in \boldsymbol{x}$, such that there is a unique solution $x^* \in \boldsymbol{x}$ to the minimax problem (1).

Interval Newton methods can be used for verification if the problem can be posed as a nonlinear system

$$\text{Find } x \in \mathbb{R}^n \text{ such that } G(x) = 0, \text{ where } x \in \mathbb{R}^n, \tag{4}$$

where G has continuous first derivatives, or at least has first derivatives with bounded interval extensions; see [16, Theorem 1.5.7] and [7, §1.5.2 and §6.2.2] or [3]. In particular, verification of solutions to the Fritz John equations can be used to verify existence of critical points of nonlinear, possibly constrained optimization problems within given bounds ([7, pp. 195–197] or [3, §10.2]). However, the objective in the minimax problem (1) is non-smooth, and the second derivatives, entering as first derivatives in the Fritz John equations, are therefore unbounded. Nonetheless, as Lemaréchal [13] and others have proposed, and as is now common practice, the minimax problem (1) can be reposed as one of the following two constrained optimization problems:

$$\begin{aligned} &\min\nolimits_{(x,v)\in\mathbb{R}^n\times\mathbb{R}} v \\ &\text{such that } \left\{ \begin{array}{r} f_i(x) \le v \\ -f_i(x) \le v \end{array} \right\}, \quad 1 \le i \le m. \end{aligned} \tag{5}$$

or

$$\begin{aligned} &\min\nolimits_{(x,v)\in\mathbb{R}^n\times\mathbb{R}} v^2 \\ &\text{such that} \quad f_i^2(x) \le v^2, \quad 1 \le i \le m. \end{aligned} \tag{6}$$

For the more general case (2), the reformulation can be simply

$$\begin{aligned} &\min\nolimits_{(x,v)\in\mathbb{R}^n\times\mathbb{R}} v \\ &\text{such that} \quad f_i(x) \le v, \quad 1 \le i \le m. \end{aligned} \tag{7}$$

In §2, we analyze the structure of the Fritz John equations (and also the Kuhn–Tucker equations) in (5) and formulation (6), and we point out that the corresponding

Jacobian matrices are singular at solutions for which the minimum is a perfect fit (i.e. for which the minimum is 0). In these cases interval Newton technology cannot be used in a straightforward way to provide rigorously verified bounds on solutions x, although appropriate variants, as well as point Newton methods, might be used in branch and bound algorithms acceleration devices. An alternative is direct reformulation, which can be, in theory, accelerated with the aid of linear relaxations. We present some techniques for this in §3. We also point out a very simple and effective alternative for constructing feasible points in that section. We present numerical results affected by the phenomena we brought to light in previous sections in §4. We summarize in §5.

Some previous work in using branch and bound algorithms or constraint propagation specifically to obtain mathematically rigorous bounds on solutions to minimax problems includes (but is not limited to) [17], [5].

2 Structure of the Fritz John Equations

We first work with the re-formulation (5). We express the Fritz John equations in terms of inequality constraints of the form

$$g(x, v) \leq 0.$$

To describe (5) in that form, we adopt the notation

$$g_i^- = -f_i(x) - v, \quad g_i^+ = f_i(x) - v, \qquad 1 \leq i \leq m,$$

with corresponding multipliers u_i^- and u_i^+. With that, the Fritz John system of equations becomes

$$\begin{aligned} &G(x, v, u) = \\ &\left(\begin{array}{c} u_0(0, \ldots, 0, 1)^T + \sum_{i=1}^{m} \left\{ u_i^- \nabla g_i^-(x, v) + u_i^+ \nabla g_i^+(x, v) \right\} \\ u_1^- g_1^-(x, v) \\ \vdots \\ u_m^- g_m^-(x, v) \\ u_1^+ g_1^+(x, v) \\ \vdots \\ u_m^+ g_m^+(x, v) \\ N(u) \end{array} \right) \\ &= 0, \end{aligned} \tag{8}$$

where $N(u) = 0$ is a normalization equation for the multipliers u_0, u_i^-, and u_i^+, $1 \leq i \leq m$.

In the following examples and analysis, we will have

$$F(x) = (f_1(x), \cdots, f_m(x))^T = Ax - b, \quad A \in \mathbb{R}^{m \times n}, \quad b \in \mathbb{R}^n, \tag{9}$$

we will be looking at minimizing $\|F\|_\infty$, we will denote the minimum by v^*, and we will denote an arbitrary minimizing point by x^*. This avoids the extra notation for second-order terms but still gives insight into many nonlinear problems. In this context, the Jacobian matrix in (8) has a structure as in Figure 1, to within a permutation of its rows. In many cases when $m \geq n$, the Jacobian matrix in Figure 1 is non-singular.

	x $\underbrace{\quad}_{n}$	v $\underbrace{\quad}_{1}$	u_0 $\underbrace{\quad}_{1}$	u_i^- $\underbrace{\quad}_{m}$	u_i^+ $\underbrace{\quad}_{m}$
$G_{1:n}$ n	$\begin{pmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$	$-(\nabla f_1, \cdots, \nabla f_m)$	$(\nabla f_1, \cdots, \nabla f_m)$
G_{n+1} 1	$(0, \cdots, 0)$	0	1	$(-1, \cdots, -1)$	$(-1, \cdots, -1)$
$u_i^- g_i^-$ m	$\begin{pmatrix} -u_1^-(\nabla f_1)^T \\ \vdots \\ -u_m^-(\nabla f_m)^T \end{pmatrix}$	$\begin{pmatrix} -u_1^- \\ \vdots \\ -u_m^- \end{pmatrix}$	$\begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$	$\mathrm{diag}(g_1^-, \cdots, g_m^-)$	$\begin{pmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{pmatrix}$
$u_i^+ g_i^+$ m	$\begin{pmatrix} u_1^+(\nabla f_1)^T \\ \vdots \\ u_m^+(\nabla f_m)^T \end{pmatrix}$	$\begin{pmatrix} -u_1^+ \\ \vdots \\ -u_m^+ \end{pmatrix}$	$\begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{pmatrix}$	$\mathrm{diag}(g_1^+, \cdots g_m^+)$
$N(u)$ 1	$(0, \cdots, 0)$	0	$*$	$(*, \cdots, *)$	$(*, \cdots, *)$

Fig. 1 The structure of the Jacobian matrix for the Fritz John system (8).

Example 1 Suppose

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ -1 & 1 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}.$$

Then the solution is

$$x^* = [0, 3]^T \quad \text{with} \quad Ax^* - b = [-1, 1, 0, -1]^T.$$

Thus, since the optimal v is $v^* = 1$, the complementarity conditions $u_i^- g_i^- = 0$, $u_i^+ g_i^+ = 0$, $1 \le i \le 4$ indicate the only possibly non-zero multipliers are u_1^-, u_4^-, and u_2^+, and the remaining 5 multipliers necessarily must equal zero. Solving with our GlobSol system [9], we obtain $u_0 = \frac{1}{2}$ and $u_1^- = u_4^- = u_2^+ = \frac{1}{6}$, if we use the normalization condition

$$N(u) = u_0 + \sum_{i=1}^{m} u_i^- + \sum_{i=1}^{m} u_i^+ - 1 = 0.$$

The Jacobian matrix of the Fritz John system corresponding to (5) at the solution $x^* = [0, 3]^T$ is thus of the form

$$G' = \left(\begin{array}{cc|c|c|cccc|cccc} x_1 & x_2 & v & u_0 & u_1^- & u_2^- & u_3^- & u_4^- & u_1^+ & u_2^+ & u_3^+ & u_4^+ \\ \hline 0 & 0 & 0 & 0 & -1 & 0 & -1 & 1 & 1 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ \hline -\frac{1}{6} & 0 & -\frac{1}{6} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{6} & -\frac{1}{6} & -\frac{1}{6} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 \\ 0 & \frac{1}{6} & -\frac{1}{6} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 \\ \hline 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right).$$

In fact, G' is non-singular, with ℓ_2 condition number slightly larger than 25.

The Fritz John system corresponding to (6) is

$$G(x, v, u) = \begin{pmatrix} u_0(0, \ldots, 0, 2v)^T + \sum_{i=1}^{m} \{u_i \nabla g_i(x, v)\} \\ u_1 g_1(x, v) \\ \vdots \\ u_m g_m(x, v) \\ N(u) \end{pmatrix} = 0, \tag{10}$$

where $g_i(x, v) = f_i^2(x) - v^2$, and its Jacobian matrix is as in Figure 2.

	x $\overbrace{\quad}^{n}$	v $\overbrace{\quad}^{1}$	u_0 $\overbrace{\quad}^{1}$	u_i $\overbrace{\quad}^{m}$
$G_{1:n}$ $\quad n$	$\sum_{i=1}^m 2u_i \nabla f_i (\nabla f_i)^T$	$\begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$	$2\,(f_1 \nabla f_1, \cdots, f_m \nabla f_m)$
G_{n+1} $\quad 1$	$(0, \cdots, 0)$	$2u_0 - 2\sum_{i=1}^m u_i$	$2v$	$-2v(1, \cdots, 1)$
$u_i g_i$ $\quad m$	$\begin{pmatrix} 2u_1 f_1 (\nabla f_1)^T \\ \vdots \\ 2u_m f_m (\nabla f_m)^T \end{pmatrix}$	$\begin{pmatrix} -2u_1 v \\ \vdots \\ -2u_m v \end{pmatrix}$	$\begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$	$\mathrm{diag}(g_1, \cdots, g_m)$
$N(u)$ $\quad 1$	$(0, \cdots, 0)$	0	$*$	$(*, \cdots, *)$

Fig. 2 The structure for the Jacobian matrix for the Fritz John system corresponding to reformulation (6).

Example 2 If the system is as in Example 1, then the solution $x^* = (0,3)^T$ with $v^* = \pm 1$ has $u_0 = \frac{1}{2}$, $u_3 = 0$, and $u_1 = u_2 = u_4 = \frac{1}{6}$; the Jacobian matrix of the Fritz John system corresponding to (6) at this solution is

$$G'(x) = \left(\begin{array}{cc|c|cccc} x_1 & x_2 & v & u_0 & u_1 & u_2 & u_3 & u_4 \\ \hline \frac{2}{3} & -\frac{1}{3} & 0 & 0 & -2 & 0 & 0 & 2 \\ -\frac{1}{3} & \frac{2}{3} & 0 & 0 & 0 & 2 & 0 & -2 \\ 0 & 0 & 0 & 2 & -2 & -2 & -2 & -2 \\ \hline -\frac{1}{3} & 0 & -\frac{1}{3} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & -\frac{1}{3} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ \frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{array}\right).$$

This matrix is non-singular, with condition number slightly greater than 14.

For both (1) and (2), the Jacobian matrix of the Fritz John system must be singular if the solution set is not isolated, as the following example of (2) illustrates.

Example 3 Suppose

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} 1 \\ 2 \end{pmatrix}.$$

In this example, the minimax problem reduces to solution of an underdetermined linear system, and the set of solutions is described by $x_1 = 1$, $x_2 = 2$, and x_3 and x_4 free. Since the solution set is not isolated, it is clear that the corresponding Fritz John matrices must be singular at such solutions. Indeed, the third and fourth rows of the Jacobian matrix in Figure 1 and the third and fourth rows of the Jacobian matrix in Figure 2 consist entirely of zeros.

A more "general" (although not necessarily more common in applications) situation, assuming the f_i are linear, is $m \geq n$, and the gradients of the f_i form a Haar system (that is, every set of n such gradients is linearly independent). In that situation, the minimax solution must be unique, so there it is reasonable to expect that the Jacobian matrices for the Fritz John system corresponding to (5) and (6) will be non-singular. However, this is often not the case: a surprising situation is when $m \geq n$ and the gradients form a Haar system, but the fit is very good, that is $\|F(x^*)\|\infty = 0$. In that case, the Jacobian matrices corresponding to both (5) and (6) must be singular at x^*. The following theorem formalizes this fact.

Theorem 1 *Suppose $m > n$, and $\min_x \|F(x)\|_\infty$ with F as in (9) has solution x^* with $\|F(x^*)\|_\infty = 0$. Then the Jacobian matrices corresponding to both* (5) *and* (6) *must be singular at x^*.*

Proof We will first consider the Jacobian matrix for (5), as in Figure 1. Observe that $v = 0$, so each of g_i^- and g_i^+ must equal zero, $1 \leq i \leq m$. Now we will analyze the determinant of the matrix in Figure 1 by adding and subtracting a column from other columns and adding and subtracting a row from other rows, then expanding by minors. First add the u_0 column to the last $2m$ columns to obtain a 0 in the G_{n+1}

row corresponding to those columns, so the only nonzero in the G_{n+1} row is in the u_0 column. Now add the u_i^- column to the u_i^+ column, $1 \le i \le m$ to obtain zeros in the last m columns and the G_i rows, $1 \le i \le m$. At this point the only non-zeros in the last m columns are in the $N(u)$ row. Since $m > n$, there are at least two such columns; if one of them indeed has a nonzero in the $N(u)$ row, we can subtract an appropriate multiple of that column from the other columns, leaving $m - 1 \ge 1$ columns entirely of zeros. Thus, the determinant must equal zero, and the matrix must be singular.

Now we consider the Jacobian matrix for (6). Recalling that $v = 0$ and each f_i is zero for $1 \le i \le m$, each g_i must also equal zero, so the m rows corresponding to the $u_i g_i$ must consist entirely of zeros, thus proving that the Jacobian matrix is singular.

Note that this argument is valid even if the f_i are nonlinear, since the second-order derivatives of the f_i only appear in the upper n by n block of the Jacobian matrix corresponding to (5) (Figure 1) or (10) (Figure 2), and this block does not play a role in the singularity analysis. Also note that the theorem is also true for the more general problem (2) in the case that all of the f_i are equal to zero at a solution.

Example 4 We generate the following exact tabular data from $y(t) = 2t + 1$.

i	t_i	y_i
1	0	1
2	1	3
3	2	5
4	3	7

Thus, the problem is $\min_x \|F(x)\|_\infty$, where $F(x) = Ax - b$, where

$$A = \begin{pmatrix} t_1 & 1 \\ t_2 & 1 \\ t_3 & 1 \\ t_4 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} 1 \\ 3 \\ 5 \\ 7 \end{pmatrix}.$$

The unique solution is, of course, $x^* = (2, 1)^T$, with $v = \|F(x^*)\|_\infty = 0$. Using the normalization

$$N(u) = \sum_{i=0}^{m} u_i - 1,$$

the Jacobian matrix corresponding to (5) and Figure 1 is thus of the form

$$G' = \left(\begin{array}{c|c|c|c|c|c|c|c|c|c|c|c} x_1 & x_2 & v & u_0 & u_1^- & u_2^- & u_3^- & u_4^- & u_1^+ & u_2^+ & u_3^+ & u_4^+ \\ \hline 0 & 0 & 0 & 0 & 0 & -1 & -2 & -3 & 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 0 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ \hline * & * & * & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ * & * & * & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ * & * & * & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ * & * & * & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline * & * & * & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ * & * & * & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ * & * & * & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ * & * & * & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right),$$

where the "$*$'s" depend on the multipliers; this matrix is clearly singular, as can be seen by adding the fourth column to each of the columns to its right, then adding u_i^- column to the u_i^+ column, $i = 1, 2, 3, 4$. Similarly, using the same normalization, the Jacobian matrix corresponding to (6) and Figure 2 is of the form

$$G' = \left(\begin{array}{c|c|c|c|c|c|c|c} x_1 & x_2 & v & u_0 & u_1 & u_2 & u_3 & u_4 \\ \hline * & * & 0 & 0 & 0 & 0 & 0 & 0 \\ * & * & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & * & * & * & * & * \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{array}\right).$$

The Jacobian matrices corresponding to the Kuhn–Tucker conditions should have similar properties: the column corresponding to u_0 and the row corresponding to $N(u)$ are merely absent, and the proofs of singularity are similar. Furthermore, one would expect the matrices to be ill-conditioned when $\|F(x^*)\|_\infty$ is small, but not exactly singular, and one would expect the rows of the m by n Jacobian matrix of F to be approximately linearly dependent (thus leading to approximate singularity of the Fritz John Jacobian matrices) when fitting simple models to data with a large number of data points (that is, with n relatively small and $m \gg n$).

3 Alternatives

The fact that the Jacobian matrices of the Fritz John or Kuhn Tucker systems are singular or ill-conditioned at solutions of problems of interest suggests that alternatives

to Newton-like methods, and, in particular, to interval Newton methods, in global optimization and other contexts may be worthwhile. In our experience, state-of-the-art algorithms such as IPOPT [18] for finding approximate local optima and optimizers can handle the reformulations well. However, interval-Newton-based verification of feasibility cannot proceed in the presence of general singularities. On the other hand, such verification of feasibility is important in finding a good but mathematically rigorous upper bound on the global optimum in constrained optimization problems. We suggest two alternatives here.

3.1 Approach the Non-Smoothness Directly

A possibility is to try to handle the non-smoothness directly, without reformulation. In principle, Newton-like methods could be used in conjunction with generalized gradients, or interval slopes. (The two concepts are related; see [12]). Computational existence and uniqueness results can sometimes even be obtained with interval Newton methods, but convergence in iterative processes is generally only linear; see [7, Chapter 6].)

Another possibility, within the context of branch and bound algorithms for global optimization, is to provide linear relaxations for $\max \cdot$ and $|\cdot|$. The general framework for our version of linear relaxations (along with further references) appears in [8], while empirical results with that scheme appear in [1]; we briefly discuss our techniques in the context of other relaxation techniques in [10, §3.1]. Essentially, we parse the expressions comprising the objective and constraints into sequences of individual operations, and an inequality or equality constraint is associated with each individual operation to result in an optimization problem equivalent to the original one. Each such simple constraint is then replaced by a set of linear constraints that together form a relaxation of the original constraint. Basic techniques are used to form relaxations for max, min, and $|\cdot|$. For example, suppose the constraint corresponding to an individual operation is

$$|x| \le v,$$

where x is assumed to lie in the interval $[\underline{x}, \overline{x}]$. The constraint is then simply replaced by the two linear constraints

$$x \le v \quad \text{and} \quad -x \le v.$$

If the constraint is

$$|x| \ge v,$$

we use the following scheme.

if $\underline{x} \ge 0$ **then**
    Replace $|x| \ge v$ by $x \ge v$.
else if $\overline{x} \le 0$ **then**
    Replace $|x| \ge v$ by $-x \ge v$.
else `/* Use a secant line between the ends of the graph. */`
    Replace $|x| \ge v$ by $\dfrac{\overline{x}+\underline{x}}{\overline{x}-\underline{x}}x - \left[\dfrac{\overline{x}+\underline{x}}{\overline{x}-\underline{x}}\underline{x} + 1\right]\underline{x}$.

For constrains $|x| = v$, it is sufficient to replace $|x| = v$ by $-x = v$ if $\overline{x} \le 0$, by $x = v$ if $\underline{x} \ge 0$, and by the two constraints $x \le v$ and $-x \le v$ in all other cases.

Constraints $\max\{x, y\} \le v$, $\max\{x, y\} \ge v$, and $\max\{x, y\} = v$ are derived similarly, and have been programmed in GlobSol.

We incorporate these techniques in our numerical experiments in §4.

3.2 A Simple "Peg to Feasibility" Approach

Local floating point optimization software will often find approximately feasible points $\check{x}$. However, for general constrained optimization problems, verifying that such points are near an exactly feasible point (or, more precisely, constructing a small box about $\check{x}$ in which a feasible point can be proven to exist) is in general problematical. Interval Newton methods cannot be used if the Fritz–John or Kuhn–Tucker system is singular, and the technique proposed in [7, 5.2.4] for equality and active inequality constraints often fails. This problem is not present with unconstrained problems, because all points are then feasible, without verification.

On the other hand, examining reformulations (5) and (6), suggests the following procedure for unconstrained minimax problems.

1. Use a local approximate optimizer to find an approximately feasible solution $(\check{x}, \check{v})$ to either reformulation (5) or reformulation (6).
2. Replace $\check{v}$ by $\tilde{v} = \max_{1 \le i \le m} |f_i(\check{x})|$.

Note that $(\check{x}, \check{v})$ must be feasible for both (5) and (6), and a mathematically rigorous feasible point can be found by evaluating the f_i at $\check{x}$ with interval arithmetic. Furthermore, if $(\check{x}, \check{v})$ is near a globally optimal point, $\tilde{v}$, representing either an upper bound on the global optimum or, in the case of (6), a bound on the square root of the global optimum, is within the accuracy of the approximate solver of the actual global optimum. Thus, this technique is essentially as good as evaluation at a point in smooth unconstrained optimization. In fact, it is equivalent (to within roundoff errors) to evaluating the original unconstrained non-smooth objective at $\check{x}$ to compute upper bounds on the optimum, but using the reformulation elsewhere, such as in constraint propagation.

Also note that, even in cases where the Jacobian matrix is singular at a solution, traditional point Newton methods may still converge, but perhaps somewhat more slowly. Also, although interval Newton methods cannot be naively used to verify existence of a solution when the Jacobian matrix at the solution is singular, they may still be effective as acceleration procedures in branch and bound methods, especially if preconditioners such as described in [7, §3.2] or [4] are used.

4 Computational Results

We illustrate the behavior of our GlobSol [9] software on the simple examples we have presented, then study the behavior on the minimax test problem collection of [14]. The source code for the version of GlobSol we are using is available by request from the first author. It includes a branch and bound algorithm as indicated in [9], which processes a box first by constraint propagation, then by computing linear relaxations, then by attempting an interval Newton method. We implemented linear relaxations of `MAX` and `ABS`, as well as generalized derivative enclosures. In addition to the linear relaxations, a local optimizer is used to obtain good upper bounds on the global optimum and to

refine optimizing points obtained through the linear relaxations. If the local solver fails for constrained problems, a simple steepest descent and a generalized Newton method are used to find feasible points; if the local solver fails on an unconstrained problem (e.g. when handling the problem directly as a non-smooth problem), a combination of steepest descent and the classical Minpack 1 routine `HYBRJ1` [15] are used to try to find a local minimum. IPOPT [18] was used as the local optimizer, and C-LP [2] was the linear program solver used to solve the relaxations.

The experiments were all done on a laptop running Ubuntu 11.10 with a Core 2 Duo T9400 processor (2 cores at 2.5GHz), and 4GB memory; the gnu compiler suite was used with gfortran 4.6.1 and gcc/g++ 4.6.1 to compile IPOPT and C-LP. (The Fortran 2003 standard intrinsic module `ISO C BINDING` was used to interface the Fortran and C/C++ codes.) Optimization level 3 was used in the compilations. Note that the CPU times may not be definitive, since the processors use speed-stepping technology, and the room's environmental temperature was not strictly controlled.

Although the execution times will vary on different machines and possibly under different conditions, the total number of boxes processed should usually be the same, given the same tolerances and same version of GlobSol. However, the total number of boxes could be different, even with standard arithmetic, when bisection occurs near boundaries, due to differences in word length of various levels of cache.

The output in all of these experiments consists of a set of boxes in which all possible global optimizers must lie, as well as mathematically rigorous bounds on any possible global optimum. We do not attempt to rigorously prove existence or uniqueness of optimizing points within particular boxes. A natural way of proving uniqueness for constrained problems is to use the Kuhn–Tucker conditions, and a natural way for smooth unconstrained problems is to set the gradient equal to zero. To prove uniqueness, the Kuhn–Tucker Jacobian matrix applied directly in an interval Newton method must be non-singular, or, more generally, sufficiently well-conditioned, but Theorem 1 gives negative results concerning non-singularity of these matrices. Although interval Newton methods can sometimes be used directly to non-smooth objectives to prove existence and uniqueness, they are effective in such cases only if certain non-trivial conditions hold; see [7, Chapter 6].

4.1 Illustrative Examples

We try

1. "peg to feasibility" in conjunction with reformulation (5),
2. "peg to feasibility" in conjunction with reformulation (6), and
3. approaching the non-smoothness directly.

4.1.1 Results for Example 1

For each scheme, we used

$$(\boldsymbol{x}, \boldsymbol{v}) = ([-10, 10], [-10, 10], [0, 10])$$

as initial search region. (Note that 0 is a natural lower bound on v.) For this problem, for both Reformulation (5), and Reformulation (6), the local solver obtained the

unique optimum immediately, GlobSol removed a box around this approximate optimum through the complementation process of [7, §4.3.1], and constraint propagation immediately eliminated the five complementary boxes so generated. Handling the problem directly as a non-smooth problem, the local solver failed, but Minpack's `HYBRJ1` succeeded. Furthermore, the same point was also found with the linear relaxation.

These results are sumarized in the following table. In this table (and in the other tables in this section), the columns are as follows:

1. "Scheme" identifies Reformulation (5), Reformulation (6), or direct handling as an unconstrained non-smooth problem ("Direct").
2. "CPU" denotes the CPU time in seconds,
3. "Boxes" denotes the total number of nodes processed in the branch and bound process,
4. "IN" denotes the number of times the interval Newton method succeeded in narrowing a coordinate bound or rejecting a box,
5. "LP" denotes the number of times a new upper bound on the optimum was obtained or a node was rejected due to the linear relaxation,
6. "CP" denotes the number of times constraint propagation succeeded in narrowing a coordinate bound or rejecting a box, and
7. "AS" denotes the number of times the local (approximate) solver succeeded in finding a feasible point. This includes both successes from the local optimizer (IPOPT), and successes from the generalized Newton method, used to project onto the feasible set in cases IPOPT failed.

Scheme	CPU	Boxes	IN	LP	CP	AS
Re. (5)	0.008	5	0	0	0	1
Re. (6)	0.02	5	0	0	0	1
Direct	0.14	9	0	1	4	1

4.1.2 Results for Example 4

Recall that this is a very simple problem in which the Jacobian matrix is singular at the solution because the fit is good. Results from GlobSol are in the following table.

Scheme	CPU	Boxes	IN	LP	CP	AS
Re. (5)	0.012	4	0	0	4	1
Re. (6)	0.032	4	0	0	4	1
Direct	0.18	50	0	1	0	1

For the "Direct" method, a good approximation to the global optimum is obtained through the linear relaxation, but the only mechanism for fathoming boxes or reducing coordinate widths after that appears to be through large lower bounds on the global optimum.

4.2 Results from a Test Problem Set in the Literature

The collection we have found is in the technical report [14]. This set consists of an interesting collection of data fitting problems, discretization of continuous minimax problems, and practical problems from the literature, many from [19]. The posted

material includes a technical report and a Fortran code `T06.FOR` for setting initial guesses and for computing values for the f_i. The 25 problems include both ℓ_∞ problems (i.e. of the form (1)) and the general problem (2). After separating the problem set into these two types, we elected, rather than modifying `TI06.FOR`, to reprogram the individual functions for GlobSol, directly from the descriptions in the technical report. Doing so, we found several inconsistencies between the function evaluation instructions in `TI06.FOR` and the descriptions in the technical report, including actual differences in the definition of the f_i and differences between the reported optimum value given in Table 2.1 of [14] and the actual global optimum. With the exception of the Watson problem, where we took the definition from `TI06.FOR`, we used the definitions from the technical report. We also eliminated the function GAMMS (problem 2.13), since we have not yet implemented the gamma function in GlobSol.

GlobSol also allows input of an initial point guess for the local optimizer. We used the point guesses from [14].

A complication is that one problem (Watson) as in `TI06.FOR` has a 2-dimensional parabola-shaped solution set, and GlobSol seeks to enclose all globally optimizing solutions in bounding boxes in a mathematically rigorous way. Similarly, the "Bard" problem has a one-dimensional line of solutions in x_2-x_3-space. (To our knowledge, such properties of these problems have not been recognized before in the literature.) Although such solution sets present convergence problems separate from the issues with non-smooth minimax problems, we elected to keep these problem in the set. Another complication is the effectiveness of certain implementation details. For example, efficient set arithmetic, such as explained in [5], has not been used, and linear relaxations of division and the trigonometric functions are not yet optimal in GlobSol. We nonetheless elected to keep those problems involving these functions in our test set.

Our resulting test problem set is as follows.

4.2.1 Test Problems Corresponding to (1)

Bard (problem 2.8 in [14]): The initial search region was $\boldsymbol{x} = [10^{-4}, 10^4]^{\times 3}$, $\boldsymbol{v} = [0, 10^4]$, where $[10^{-4}, 10^4]^{\times 3}$ is interpreted to mean $[10^{-4}, 10^4], [10^{-4}, 10^4], [10^{-4}, 10^4]$). The smallest scaled box diameter ε_{d} for GlobSol was set to 10^{-6}, and singular expansion factor $\sigma = 5$. (Boxes that are fathomed due to their scaled diameter are expanded in each coordinate about their center points so their scaled diameters are $\sigma\sqrt{\varepsilon_{\mathrm{d}}}$, and the resulting regions are removed from the search region through the complementation process of [7, §4.3.1]. This limits unnecessary computation due to the clustering phenomenon of [11].) The optimum is about 5×10^{-2}; the results are as follows.

Scheme	CPU	Boxes	IN	LP	CP	AS
Re. (5)	72.62	11750	90702	3	255	1252
Re. (6)	190.1	32812	485216	3	2338	3641
Direct	148.6	21966	0	1	20566	213

Figure 3 shows the enclosures GlobSol gives for the one-dimensional set of global optimizing points.

Davidon 2 (problem 2.10 in [14]): The initial search region was $\boldsymbol{x} = [-10^4, 10^4]^{\times 4}$, $\boldsymbol{v} = [0, 10^6]$, $\varepsilon_{\mathrm{d}} = 10^{-4}$, and $\sigma = 5$. GlobSol found the solution reported in [19] to

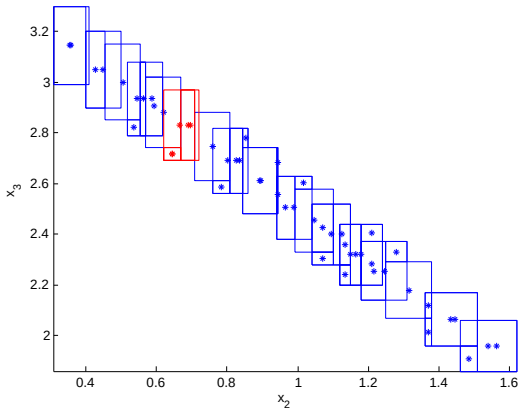

Fig. 3 GlobSol output boxes for the Bard problem, scheme (5)

be unique within the search region. The optimum is about 1.16×10^2; the results are as follows.

Scheme	CPU	Boxes	IN	LP	CP	AS
Re. (5)	0.304	23	166	0	22	7
Re. (6)	1.084	34	292	1	33	9
Direct	5.492	147	13	1	60	5

EVD61 (problem 2.16 in [14]): Observe that, if x_4 spans more than π, this generates multiple solutions with x_1; however, starting with an initial guess $x_4 = 0$ leads our local optimizer to a negative x_4, so we include negative values of x_4. We obtained some, but not totally satisfactory resolution of the global minimum with $\boldsymbol{x} = ([0,10],[0,10],[3.14,9.4],[-3.15,0.01],[-10,10],[0,5])$, $\boldsymbol{v} = [0,5]$, $\varepsilon_{\mathrm{d}} = 2 \times 10^{-4}$, and $\sigma = 3$. The lower bound obtained on the global optimum was about 10% larger than the reported optimum of 0.035 in [14]. This problem was difficult for GlobSol. Because of this, we do not report results here.

Filter (problem 2.18 in [14]): Satisfactory results were not obtained for this problem.

Kowalik-Osborne (problem 2.9 in [14]): The initial search region was $\boldsymbol{x} = [0,10]^{\times 4}$, $\boldsymbol{v} = [0,0.2]$, $\check{v} = 0.1$, with $\sigma = 2$, $\varepsilon_{\mathrm{d}} = 10^{-4}$. The optimim is roughly 8×10^{-3}; the results are as follows.

Scheme	CPU	Boxes	IN	LP	CP	AS
Re. (5)	794.3	100146	619486	2	1721	9
Re. (6)	1866.	144164	517773	17	21451	3118
Direct	75.25	19971	0	8	12715	17

OET5 (problem 2.11 in [14]): We used $\boldsymbol{x} = [-10,10]^{\times 3}$, $\boldsymbol{v} = [0,0.5]$, $\varepsilon_{\mathrm{d}} = 4 \times 10^{-3}$, and $\sigma = 3$. Note that the minimizer (x_1, x_2, x_3, x_4) is paired with a second minimizer $(-x_1, -x_2, -x_3, x_4)$. Furthermore, although the optimal objective value is 0.00263597 (to six digits), there is a parabola-shaped region of approximate solutions (with objective value at most 0.0733467), as illustrated in Figure 4. The

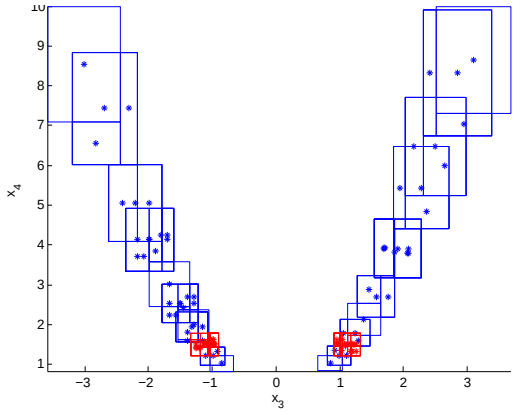

Fig. 4 GlobSol output boxes (x_4 against x_3) for the OET5 problem, scheme (5)

optimum is roughly 2.6×10^{-3}; we obtained the following results.

Scheme	CPU	Boxes	IN	LP	CP	AS
Re. (5)	1386	57419	986	7	38234	1
Re. (6)	4725	178173	0	12	32011	34217
Direct	≥ 7134	> 400000	≥ 2853	≥ 1	≥ 0	≥ 14

Here, the algorithm using the unconstrained non-smooth objective ("direct") could not finish in 75600 seconds, and the statistics appearing represent the quantities at the point the algorithm detected the time had been exceeded; at that time, there were still 124,161 boxes remaining to be processed. Also note that, in the "Direct" case, the best attained upper bound on the global optimum was 0.00283071 (rounded), contrasting with the more accurate bound 0.00263597 (rounded); an inability to obtain a good upper bound significantly affects performance.

OET6 (problem 2.12 in [14]): We used $\boldsymbol{x} = [-10, 10]^{\times 4}$, $\boldsymbol{v} = [0, 10^6]$, $\varepsilon_{\mathrm{d}} = 10^{-4}$, $\sigma = 10$. The optimum reported in [14] was roughly 2×10^{-3}, but GlobSol computed a rigorous upper bound on the optimum of about 2.6×10^{-12}, so something may be wrong in the statement of or implementation of this problem. We obtained the following results with the problem as stated in [14].

Scheme	CPU	Boxes	IN	LP	CP	AS
Re. (5)	> 57600	≥ 48558	≥ 0	≥ 0	≥ 20558	≥ 25410
Re. (6)	16980	246869	0	12	38070	87765
Direct	≥ 12490	> 400000	≥ 15	≥ 33	≥ 0	≥ 10

Osborne2 (problem 2.25 in [14]): Although the approximate optimizer successfully completes, properties of this problem pose difficulties to the GlobSol algorithm unrelated to the issue of singularity in minimax reformulations. In particular, the symmetries between (x_2, x_6, x_9), (x_3, x_7, x_{10}) and (x_4, x_8, x_{12}) reveal many solutions, and this combines with the large nonlinearities in the exponentials. Such symmetries and nonlinearities would be best treated by appending symmetry-breaking

constraints and reformulating the exponentials in terms of other variables. We thus do not report results for this problem here.

PBC1 (problem 2.15 in [14]): We used $\boldsymbol{x} = [-100, 100]^{\times 4}$, $\boldsymbol{v} = [0, 10^6]$, $\varepsilon_{\mathrm{d}} = 10^{-4}$, $\sigma = 10$, with initial guess as in [14] and initial v equal to 1. (The approximate solver failed to converge with initial v equal to 0.) The optimum is roughly 2×10^{-2}, and we obtained the following results:

Scheme	CPU	Boxes	IN	LP	CP	AS
Re. (5)	4691.9	115204	1348396	0	11513	253
Re. (6)	10765.3	205263	1233096	0	1450	14184
Direct	4047.0	196322	170	0	30754	18

4.2.2 An Overall Comment

An initial computation in GlobSol is to run the approximate optimizer (in our case, IPOPT) before starting the branch and bound process; if the approximate optimizer is successful, GlobSol attempts to construct a small box about the point the approximate optimizer returns, within which it is rigorously verified that a feasible point exists. In turn, the objective is evaluated with interval arithmetic over this small box to obtain an upper bound on the global optimum. (In the case of unconstrained problems, including unconstrained minimax, all that need be done is evaluate the objective, using interval arithmetic, at the point the approximate optimizer returns.) This process often gives a very good upper bound on the global optimum, that the branch and bound process uses effectively to complete its exhaustive search for optimizers quickly. When the problem is formulated in GlobSol directly as a non-smooth problem, IPOPT generally cannot handle the non-smoothness, and returned in failure mode for all of the examples reported in in this work. In contrast, IPOPT failed in this initial computation only with "Filter," and it failed both with reformulation (5) and reformulation (6).

A more sophisticated implementation might be to use a reformulation in the approximate optimizer, but to use "direct" computations in the interval arithmetic-based part of the branch and bound algorithm.

4.3 An Additional Application from the Literature

For comparison of the overall algorithm, we tried GlobSol on the application in [5, Section 5], using reformulation (5), the most successful of the reformulations, overall, from the test problem set. GlobSol completed successfully in a reasonable amount of time. However, GlobSol's total processing time and number of boxes considered were each larger than the results reported in [5]. Nonetheless, although the output reported in [5] was with 44 boxes, GlobSol returns only two boxes, each with very narrow bounds, as well as the narrow bounds of $[0.06561, 0.06566]$ on the global optimum. Abridged GlobSol output is as follows.

Table 1 Number of boxes versus optimum value

Problem	Optimum	Boxes (5)	Boxes (6)	Boxes, direct
Bard	0.05	11750	32812	21966
Davidon2	115.7	23	34	147
Kowalik-Osborne	0.008	100146	144164	19971
OET5	0.0026	97087	178173	> 400000
PBC1	0.02	115204	205263	196322
correlation coefficients (boxes versus optimum value)		-0.666	-0.690	-0.415

```
Output from FIND_GLOBAL_MIN on  09/23/2012  at  10:57:06.
Initial box:
[   -60.00    ,    60.00     ], [   -1.000    ,    0.000     ]
[   -60.00    ,    60.00     ], [   -1.000    ,    0.000     ]
 -------------------------------------------------
[    0.000    ,    120.0     ]
LIST OF BOXES CONTAINING VERIFIED FEASIBLE POINTS:
Box no.:           1
Box coordinates:
[    21.08    ,    21.12     ], [  -0.7775    ,  -0.7755     ]
[   -10.94    ,   -10.92     ], [  -0.2119    ,  -0.2099     ]
[   0.6466E-01,   0.6666E-01 ]
Box contains the following approximate root:
  21.10    ,  -0.7765    ,   -10.93    ,  -0.2109    ,   0.6566E-01
OBJECTIVE ENCLOSURE AT APPROXIMATE ROOT: [   0.6566E-01,   0.6566E-01 ]
-------------------------------------------------
Box no.:           2
Box coordinates:
[   -10.94    ,   -10.92     ], [  -0.2119    ,  -0.2099     ]
[    21.08    ,    21.12     ], [  -0.7775    ,  -0.7755     ]
[   0.6466E-01,   0.6666E-01 ]
Box contains the following approximate root:
 -10.93    ,  -0.2109    ,    21.10    ,  -0.7765    ,   0.6566E-01
OBJECTIVE ENCLOSURE AT APPROXIMATE ROOT: [   0.6566E-01,   0.6566E-01 ]
-------------------------------------------------
BEST_ESTIMATE:   0.6566E-01
BEST_LOWER_BOUND:   0.6561E-01
Number of bisections:          501
Total number of boxes processed in loop:          736
Overall CPU time:    10.88
```

Improvements can probably be made by combining the techniques in [5] with those in this work and in GlobSol.

4.3.1 Relationship to Theorem 1

Here, we examine the correlation between the amount of work required to complete the branch and bound process and how good the ℓ_∞ fit is (that is, how close the optimum value is to 0). Table 1 compares the total number of boxes produced during the branch and bound process to the optimum value, while Table 2 compares the total execution time to the optimum value.

We see through the negative correlations that near-singularity due to a good fit does affect the overall branch and bound processes. The less pronounced correlation with CPU time than with number of boxes may be due to the algorithm favoring the less computationally intensive constraint propagation to the interval Newton method in cases where the optimum is near zero.

Table 2 CPU time versus optimum value

Problem	Optimum	CPU (5)	CPU (6)	CPU, direct
Bard	0.05	72.62	190.1	148.6
Davidon2	115.7	0.30	1.08	5.1
Kowalik-Osborne	0.008	794.3	4725.	75.25
OET5	0.0026	4943.	16980.	$\geq$7134.
PBC1	0.02	1386.	10765.3	4047.
correlation coefficients (CPU versus optimum value)		-0.395	-0.500	-0.396

4.4 Test Problems Corresponding to (2)

For comparison, we examined the test problems from [14] corresponding to (2): In general, unless these problems consist of explicit reformulations corresponding to (1), difficulties with singularities when the fit is good may not appear so frequently.

CB2 (problem 2.1 in [14]): We used $\boldsymbol{x} = [-10, 10]^{\times 2}$, $\varepsilon_{\mathrm{d}} = 10^{-8}$, $\sigma = 10$, with the following results. (The optimum value is about 1.95.)

Scheme	CPU	Boxes	IN	LP	CP	AS
Re. (7)	0.028	5	8	0	5	3
Direct	27.802	10035	11	1	10035	42

EVD52 (problem 2.4 in [14]): We used $\boldsymbol{x} = [-10, 10]^{\times 2}$, $\varepsilon_{\mathrm{d}} = 10^{-8}$, $\sigma = 10$, with the following results. (The optimum value is about 3.60.)

Scheme	CPU	Boxes	IN	LP	CP	AS
Re. (7)	0.064	7	222	0	7	5
Direct	77.061	29988	0	1	25976	148

EXP (problem 2.14 in [14]): We eliminated this problem, since it is unbounded as stated in [14]: To see this, take $x_2 = x_3 = x_4 = x_5 = 0$, and let x_1 be negative with $|x_1|$ arbitrarily large.

Polak2 (problem 2.22 in [14]): We used $\boldsymbol{x} = [-10, 10]^{\times 10}$, $\boldsymbol{v} = [-10^6, 10^6]$, $\varepsilon_{\mathrm{d}} = 10^{-8}$, $\sigma = 10$, with the following results. (The optimum is about 54.5.)

Scheme	CPU	Boxes	IN	LP	CP	AS
Re. (7)	0.184	21	0	0	21	1
Direct	1.648	18	0	0	18	1

Polak3 (problem 2.23 in [14]): The optimum of 261.08 reported in [14] does not correspond to the actual optimum of problem 2.23 given in [14]. To see this, note that setting each x_i equal to zero gives that each f_i is bounded below by 0 and above by

$$\sum_{j=0}^{10} \frac{1}{1+j} e^1 \leq 3.02e < 8.21.$$

For this reason and other difficulties, we have not reported results on this problem.

Polak6 (problem 2.6 in [14]): We used $\boldsymbol{x} = [-10, 10]^{\times 4}$, $\boldsymbol{v} = [-10^6, 10^6]$, $\varepsilon_{\mathrm{d}} = 10^{-4}$, $\sigma = 10$, $\boldsymbol{v} = [1, 11]$, with the midpoints of these coordinates used for the first initial guess for the approximate solver, with the following results. (The optimum is about -44.)

Scheme	CPU	Boxes	IN	LP	CP	AS
Re. (7)	570.5	8218	29812	1	7091	3500
Direct	50089.9	992091	90312	0	315858	23

Rosen–Suzuki (problem 2.5 in [14]): We used $\boldsymbol{x} = [-10, 10]^{\times 4}$, $\boldsymbol{v} = [-10^6, 10^6]$, $\varepsilon_{\mathrm{d}} = 10^{-4}$, $\sigma = 10$, with $x = (0, 0, 0, 0)$ and $v = 1$ used for the first initial guess for the approximate solver. (The optimum is about -44.)

Scheme	CPU	Boxes	IN	LP	CP	AS
Re. (7)	10.72	810	5487	1	595	426
Direct	> 57600.40	≥ 1052029	≥ 71965	≥ 0	≥ 0	≥ 27

(A time limit of 57600 seconds was exceeded.)

SPIRAL (problem 2.3 in [14]): We used $\boldsymbol{x} = [-1000, 1000]^{\times 2}$, $\boldsymbol{v} = [-10^6, 10^6]$, $\varepsilon_{\mathrm{d}} = 10^{-8}$, $\sigma = 10$, with initial initial guess as in [14], that is,
$x = (1.41831, -4.79462)$, and $v = 1$ used for the first initial guess for the approximate solver. (The optimum is near 0.)

Scheme	CPU	Boxes	IN	LP	CP	AS
Re. (7)	0.04	4	0	0	0	1
Direct	0.05	1	0	0	0	1

WF (problem 2.2 in [14]): We used $\boldsymbol{x} = [0, 10]^{\times 2}$, $\boldsymbol{v} = [-10^6, 10^6]$, $\varepsilon_{\mathrm{d}} = 10^{-8}$, $\sigma = 10$, with initial initial guess as in [14], that is, $x = (3, 1)$, and $v = 1$ used for the first initial guess for the approximate solver. (The optimum is near 0.)

Scheme	CPU	Boxes	IN	LP	CP	AS
Re. (7)	0.10	38	337	0	11	0
Direct	11.61	36	0	0	17	2

Wong1 (problem 2.19 in [14]): We used $\boldsymbol{x} = [-10, 10]^{\times 7}$, $\boldsymbol{v} = [-10^6, 10^6]$, $\varepsilon_{\mathrm{d}} = 10^{-8}$, $\sigma = 10$, with initial initial guess as in [14] , that is, $x = (1, 2, 0, 4, 0, 1, 1)$, and $v = 1$ used for the first initial guess for the approximate solver. Note: The best lower bound on the global optimum obtained (about 691.7) was about 1.6% larger than the optimum reported in [14] (about 680.6).

Scheme	CPU	Boxes	IN	LP	CP	AS
Re. (7)	99.40	2388	10193	0	2196	1364
Direct	> 57600	≥ 980455	≥ 192042	≥ 0	≥ 42196	≥ 449350

Wong2 (problem 2.20 in [14]): We used $\boldsymbol{x} = [-100, 100]^{\times 10}$, $\boldsymbol{v} = [-10^6, 10^6]$, $\varepsilon_{\mathrm{d}} = 10^{-8}$, $\sigma = 10$, with initial initial guess as in [14], that is,
$x = (2, 3, 5, 5, 1, 2, 7, 3, 6, 10)$, and $v = 1$ used for the first initial guess for the approximate solver. Note: The best lower bound on the global optimum obtained was about 33.5, whereas the optimum reported in [14] was about 24.3.

Scheme	CPU	Boxes	IN	LP	CP	AS
Re. (7)	290.68	13129	118122	5	68894	12958
Direct	> 57600	≥ 958437	≥ 2849	≥ 0	≥ 78561	≥ 2087

4.5 Advice

Based on both the theory and our practical experience, we formulate the following advice.

1. Reformulation (5) seems to usually be better in practice, despite the fact it involves twice as many constraints.
2. In choosing an initial guess for a general point iterative method for constrained optimization problems, it is better to choose $v \gg 0$ rather than $v = 0$.
3. Although, in theory, use of intervals instead of gradients (theoretically equivalent to using the set of subgradients) allows solution of minimax problems without continuous reformulations, in practice this is rarely advisable. This is likely due to the approximate solver not able to obtain optimizing points when the problem is non-smooth. Nontheless, the technique may still have its place in specific parts of a branch and bound algorithm.
4. In general, time to complete the algorithm varies substantially from problem to problem, depending on problem characteristics. For particular problems, search boxes $\boldsymbol{x}$ and tolerances ε_{d} and σ may need to be adjusted to solve the problem in a practical way.
5. An important feature of complete search algorithms is the "end game," that is, how the leaves of the search tree are handled. GlobSol uses the expansion process first described in [6] to expand such small boxes and eliminate from consideration adjacent small boxes, without losing the verified enclosure property. Controlled by σ and ε_{d}, exactly how this process is carried out, and how it interacts with non-isolated sets of optimizers, greatly affects the overall performance of the algorithm.

5 Conclusion and Discussion

Even though the Jacobian matrix for the Fritz John system is singular, that does not completely rule out use of the reformulations (5) and (6) in global optimization algorithms; it merely rules out use of traditional interval Newton methods based on the Fritz John system for verification, or it indicates that traditional iterative methods based on the Fritz John system may not work well. In fact, we discovered the singularity while experimenting with the reformulations (6) with problems from [14] within our GlobSol system [9]. Even so, we found that properly preconditioned interval Newton methods could make progress in reducing coordinate widths of the search region, even though they could not be used with the Fritz John system to verify existence and uniqueness of critical points[1].

[1] Most of the time the interval Newton made progress in this context, the preconditioner was a permutation of the identity matrix. This may be due to unbounded entries in the Jacobian matrix.

It is to be expected that, in data fitting with a large amount of data, adjacent rows in the matrix A (or corresponding Jacobian matrix, in the nonlinear case) are approximately linearly dependent, leading to approximate singularity of the Fritz John system.

Although we have discussed the Fritz John systems, similar ideas are valid for the Kuhn Tucker system.

In addition to revealing information about branch and bound algorithms for minimax problems, this study has also revealed weaknesses in the implementation of linear relaxations in GlobSol. Future improvements in the sharpness of the linear relaxations may change the results reported here slightly, but probably will not change the overall conclusions.

Acknowledgments

The authors wish to thank the referees for their careful reading and helpful suggestions.

References

1. Ralph Baker Kearfott. Discussion and empirical comparisons of linear relaxations and alternate techniques in validated deterministic global optimization. *Optimization Methods and Software*, 21:715–731, October 2006.
2. Julian Hall. Homepage of C-LP, 2002. `https://projects.coin-or.org/Clp`.
3. E. R. Hansen. *Global Optimization Using Interval Analysis*. Marcel Dekker, Inc., New York, 1992.
4. C. Hu. *Optimal Preconditioners for the Interval Newton Method*. PhD thesis, University of Southwestern Louisiana, 1990.
5. L. Jaulin. Reliable minimax parameter estimation. *Reliable Computing*, 7(3):231–246, 2001.
6. Baker Kearfott. A proof of convergence and an error bound for the method of bisection in $\mathbf{R}^n$. *Math. Comp.*, 32(144):1147–1153, October 1978.
7. R. B. Kearfott. *Rigorous Global Search: Continuous Problems*. Kluwer, Dordrecht, Netherlands, 1996.
8. R. Baker Kearfott and Siriporn Hongthong. Validated linear relaxations and preprocessing: Some experiments. *SIAM J. on Optimization*, 16(2):418–433, 2005.
9. Ralph Baker Kearfott. GlobSol user guide. *Optimization Methods and Software*, 24(4-5):687–708, August 2009.
10. Ralph Baker Kearfott. Interval computations, rigour and non-rigour in deterministic continuous global optimization. *Optim. Methods Softw.*, 26(2):259–279, April 2011.
11. Ralph Baker Kearfott and Kaisheng Du. The cluster problem in multivariate global optimization. *Journal of Global Optimization*, 5:253–265, 1994.
12. R.B. Kearfott and Humberto Muñoz. Slope interval, generalized gradient, semigradient, and slant derivative. *Reliable Computing*, 10(3):163–193, June 2004.
13. C. Lemaréchal. Nondifferentiable optimization. In *Nonlinear Optimization 1981*, pages 85–89, New York, 1982. Academic Press.
14. L. Lukšan and J. Vlček. Test problems for nonsmooth unconstrained and linearly constrained optimization. Technical Report 798, Institute of Computer Science, 2000.
15. J. J. Moré, B. S. Garbow, and K. E. Hillstrom. User guide for MINPACK-1. Technical Report ANL-80-74, Argonne National Laboratories, 1980.
16. A. Neumaier. *Interval Methods for Systems of Equations*. Cambridge UniverCsity Press, Cambridge, England, 1990.
17. Z. Shen, A. Neumaier, and M. C. Eiermann. Solving minimax problems by interval methods. *BIT*, 30:742–751, 1990.
18. A. Wächter. Homepage of IPOPT, 2002. `https://projects.coin-or.org/Ipopt`.
19. G. A. Watson. The minimax solution of an overdetermined system of non-linear equations. *Journal of the Institute of Mathematics and its Applications*, 23(2):167–180, 1979.
20. G. A. Watson. Approximation in normed linear spaces (a historical survey of numerical methods). *Journal of Comput. Appl. Math.*, 121:1–36, 2000.