Journal of Global Optimization manuscript No.
(will be inserted by the editor)

Rigorous verification of feasibility

Ferenc Domes · Arnold Neumaier

2013.09.10

Abstract This paper considers the problem of finding and verifying feasible points for constraint satisfaction problems, including those with uncertain coefficients. The main part is devoted to the problem of finding a narrow box around an approximately feasible solution for which it can be rigorously and automatically proved that it contains a feasible solution. Some examples demonstrate difficulties when attempting verification.
We review some existing methods and present a new method for verifying the existence of feasible points of constraint satisfaction problems in an automatically determined narrow box. Numerical tests within GloptLab, a solver developed by the authors, show how the different methods perform.
Also discussed are the search for approximately feasible points and the use of approximately feasible points within a branch and bound scheme for constraint satisfaction.

Keywords rigorous feasibility verification · constraint satisfaction · global optimization · verified computing · interval analysis

1 Introduction

There are many search techniques for finding solutions of a constraint satisfaction problem (CSP) in the real domain. However, in floating point arithmetic of fixed accuracy, local search routines typically find only approximately feasible points, even if the problem itself is feasible.

As a consequence, local or global solvers for CSPs are often content with finding approximately feasible points. For example, the price-winning global solver Baron ([22]) often considered as the state of the art for branch-and-bound methods, quits once a point is found satisfies some δ-feasibility condition – irrespective of whether

F. Domes
University of Vienna, Faculty of Mathematics, Vienna, Austria, E-mail: Ferenc.Domes@univie.ac.at

A. Neumaier
University of Vienna, Faculty of Mathematics, Vienna, Austria, E-mail: Arnold.Neumaier@univie.ac.at

or not there is a nearby feasible point. This may happen even when there are feasible points far from the boundary of the feasible set, so that *every* nearby floating point vector is feasible, too. The user gets less than he could hope for from a global solver.

In a branch and bound context, one should therefore continue the search until one can either verify the feasibility of a point, or at least guarantee the existence of a feasible point close to the point returned. The former is possible of course only if the problem has a feasible point representable in the floating-point format used; thus the second alternative will usually be the case when the problem contains equality constraints, or more generally when the feasible domain has an empty interior. Indeed, it is quite likely under these circumstances that no feasible point is representable exactly in the floating-point format used.

Thus we need methods for verifying the feasibility of a point, or if that fails, for finding close to a given point a narrow region that is guaranteed to contain a feasible point. Such verification methods are usually based on outward rounding interval arithmetic. They can be seamlessly integrated into branch and bound codes as these make use of interval techniques also in other respects, e.g., in constraint propagation.

For linear and nonlinear systems of equations, the verification of feasibility is usually based on fixed point theorems, and has a long history; see, e.g., [1, 12, 15, 16, 19–21]. General constraint satisfaction problems in the real domain usually contain also inequalities, and the methods for equations need appropriate adaptation. In the past, these were exclusively discussed in the context of global optimization methods, where they are indispensable for getting verified upper bounds on the objective. In particular, Hansen [8, Section 12] and Kearfott [9–11] discuss methods for finding a narrow box $\mathbf{z}$ centered at a given approximately feasible point, such that it can be verified that it contains a feasible point. Of course, there may be no closeby feasible point, in which case these methods will return without a result.

In this paper, we review some existing methods and present a new method for verifying the existence of feasible points of constraint satisfaction problems in an automatically determined narrow box. Numerical tests within GloptLab [2], a solver developed by the authors, show how the different methods perform. Also discussed are the search for approximately feasible points and the use of approximately feasible points within a branch and bound scheme for constraint satisfaction.

While traditionally the coefficients of a constraint in a constraint satisfaction problem are taken to be exactly known, we allow them to vary in (narrow) intervals, to be able to rigorously account for uncertainties due to one of the following sources. This defines our problem class as uncertain constraint satisfaction problems as a generalization of traditional constraint satisfaction problems.

As already alluded to above, an important application of the techniques discussed in this paper is to rigorous constrained global optimization. Here truly feasible points are necessary to obtain valid upper bounds on the optimal objective function value. Using such an upper bound from a feasible (and ideally nearly optimal) point eliminates most of the search space, leaving a CSP with a tiny feasible region only. Therefore these verification techniques usually save a large amount of time by speeding up the branch and bound process.

The paper is organized as follows. In Section 2 we define the problem class treated, the uncertain constraint satisfaction problems. In Section 3 we discuss finding and using approximately feasible points. In particular, finding approximately feasible points (Subsection 3.1), re-using approximately feasible points in a new subbox (Subsection 3.2) and feasible point search in branch and bound (Subsection 3.3). In Section 4

some existing verification techniques are reviewed for nonsingular systems of nonlinear equations (Subsection 4.1), for systems of nonlinear equations (Subsection 4.2), for systems of nonlinear inequalities (Subsection 4.3) and for general constraint satisfaction problems (Subsection 4.4). Our new verification method is presented in Section 5. We conclude our paper with Section 6, a comparison of the different verification techniques.

Notation. The n-dimensional identity matrix is denoted by I_n and the n-dimensional zero matrix is denoted by 0_n. The ith row vector of a matrix A is denoted by $A_{i:}$ and the jth column vector by $A_{:j}$. The number of nonzero entries of a matrix A is denoted by $\mathrm{nnz}(A)$. The set $\neg N$ denotes the complement of a set N. The number of elements of a set N is denoted by $|N|$. Let $I \subseteq \{1,\ldots,m\}$ and $J \subseteq \{1,\ldots,n\}$ be index sets and let $n_I := |I|$, $n_J := |J|$. Let x be an n-dimensional vector, then x_J denotes the n_J-dimensional vector built from the components of x selected by the index set J. For an $m \times n$ matrix A the expression $A_{I:}$ denotes the $n_I \times n$ matrix built from the rows of A selected by the index sets I. Similarly, $A_{:J}$ denotes the $m \times n_J$ matrix built from the columns of A selected by the index sets J. Instead of using the index sets I and J we also write $A_{i:k,j:l}$ for some $i \leq k \leq n$ and $j \leq l \leq m$ denoting that $I = \{i,\ldots,k\}$ and $J = \{j,\ldots,l\}$. $(A^T)^{-1}$ is denoted by A^{-T}. For vectors and matrices the comparison operators $=$, $\neq$, $<$, $>$, $\leq$, $\geq$ and the absolute value $|A|$ of a matrix A are interpreted component-wise. For an $n \times n$ matrix A, $\mathrm{diag}(A)$ denotes the n-dimensional vector with $\mathrm{diag}(A)_i = A_{ii}$.

For the notation of the most used quantities and operators in interval analysis we refer to [13]. In addition to this

$$\mu(\mathbf{a}) := \begin{cases} \underline{a} \text{ if } \underline{a} > 0, \\ \overline{a} \text{ if } \overline{a} < 0, \\ 0 \text{ otherwise,} \end{cases} \tag{1}$$

denotes the **minimal point**,

$$\langle \mathbf{a} \rangle := |\mu(\mathbf{a})|$$

denotes the **mignitude** and

$$|\,\mathbf{a}| := \max(-\underline{a}, \overline{a}),$$

denotes the **magnitude** of an interval $\mathbf{a}$. An interval is called **thin** or **degenerate** if its width is zero.

Let S be a set then $\square S := [\,\inf(S), \sup(S)]$ is called the **interval hull** of S.

A **box** (interval vector) $\mathbf{x} = [\underline{x}, \overline{x}]$ or is the Cartesian product of the closed real intervals $\mathbf{x}_i := [\underline{x}_i, \overline{x}_i]$, representing a (bounded or unbounded) axiparallel box in $\mathbb{R}^n$. The values $-\infty$ and ∞ are allowed as lower and upper bounds, respectively, to take care of one-sided bounds on variables. The condition $x \in \mathbf{x}$ is equivalent to the collection of simple bounds

$$\underline{x}_i \leq x_i \leq \overline{x}_i \quad (i = 1,\ldots,n),$$

or, with inequalities on vectors and matrices interpreted component-wise, to the two-sided vector inequality $\underline{x} \leq x \leq \overline{x}$. Apart from two-sided constraints, this includes with $\mathbf{x}_i = [a,a]$ variables x_i fixed at a particular value $x_i = a$, with $\mathbf{x}_i = [a,\infty]$ lower bounds $x_i \geq a$, with $\mathbf{x}_i = [-\infty,a]$ upper bounds $x_i \leq a$, and with $\mathbf{x}_i = [-\infty,\infty]$

free variables. $\overline{\mathbb{IR}}^n$ denotes the set of all n-dimensional boxes. Operations defined for intervals (like width, midpoint, distance, mignitude and magnitude) are interpreted component-wise when applied to boxes.

We also consider an expression $p(x)$ in $x = (x_1, \ldots, x_n)^T$ such that the evaluation at any $x \in \mathbf{x}$ is a real number. The box $p(\mathbf{x})$ is called an **interval enclosure** of $p(x)$ in the box $\mathbf{x}$ if $p(x) \in p(\mathbf{x})$ holds for all $x \in \mathbf{x}$. There are a number of methods for defining $p(\mathbf{x})$, for example interval evaluation or centered forms (for details, see, e.g., [16]).

To account for inaccuracies in computed entries of a matrix, we use interval matrices, standing for uncertain real matrices whose coefficients are between given lower and upper bounds. The expression $\mathbf{A} := [\underline{A}, \overline{A}] \in \overline{\mathbb{IR}}^{m\times n}$ denotes an $m \times n$ interval matrix with lower bound $\underline{A}$ and upper bound $\overline{A}$. $\mathbf{A} \in \overline{\mathbb{IR}}^{n\times n}$ is symmetric if $\mathbf{A}_{ik} = \mathbf{A}_{ki}$ for all $i, k \in \{1, \ldots, n\}$. The comparison matrix $\langle \mathbf{A} \rangle$ of a square interval matrix $\mathbf{A}$ is defined by

$$\langle \mathbf{A} \rangle_{ij} := \begin{cases} -|\,\mathbf{A}_{ij}| & \text{for } i \neq j, \\ \langle \mathbf{A}_{ij} \rangle & \text{for } i = j. \end{cases}$$

A real matrix A is identified with the thin interval matrix with $\underline{A} = \overline{A} = A$. The width and the radius of an interval matrix $\mathbf{A}$ are the noninterval matrices defined by

$$\operatorname{wid}(\mathbf{A}) := \overline{A} - \underline{A}, \text{ and } \operatorname{rad}(\mathbf{A}) := \operatorname{wid}(\mathbf{A})/2,$$

respectively.

2 Uncertain constraint satisfaction problems

The traditional continuous **constraint satisfaction problem** (CSP) with equality and inequality constraints may be written in the compact interval form

$$F(x) \in \mathbf{F}, \quad x \in \mathbf{x}, \tag{2}$$

with $F : \mathbb{R}^n \to \mathbb{R}^m$, $\mathbf{F} \in \overline{\mathbb{IR}}^m$, and $\mathbf{x} \in \overline{\mathbb{IR}}^n$. Here components of $\mathbf{F}$ and $\mathbf{x}$ are thin for equality constraints, unbounded for inequality constraints, and bounded but thick for two-sided constraints. A point $z \in \mathbf{x}$ is called **feasible** or a **solution** of (2) if $F(z) \in \mathbf{F}$ is satisfied. The exact constraint satisfaction problem is called **feasible** if it has at least one solution otherwise it is called **infeasible**.

While traditionally the coefficients of a constraint in a constraint satisfaction problem are taken to be exactly known, we allow them to vary in (narrow) intervals, to be able to rigorously account for uncertainties due to one of the following sources:

- measurements of limited accuracy,
- conversion errors from an original representation to our normal form,
- rounding errors when creating new constraints by relaxation techniques.

Such uncertain constants can be naturally expressed if we formulate the constraints in the form of the following **uncertain constraint satisfaction problem** (UCSP)

$$BF(x) \in \mathbf{b}, \quad x \in \mathbf{x}, \quad B \in \mathbf{B}, \tag{3}$$

where $F : \mathbb{R}^n \to \mathbb{R}^w$, $\mathbf{b} \in \mathbb{IR}^m$, $\mathbf{x} \in \mathbb{IR}^n$, and $\mathbf{B} \in \mathbb{IR}^{m\times w}$. The entries of B are *not* variables but uncertain constants whose precise values within the bounds $B \in \mathbf{B}$ are not known. Thus whether a particular vector x is a solution of the UCSP may depend on which $B \in \mathbf{B}$ is the true value. This ambiguity makes working with uncertain constraints nontrivial. It requires great care in the derivation of methods to ensure the validity of an enclosure no matter which value $B \in \mathbf{B}$ is the true value.

If $\mathbf{B}$ contains only a single matrix, (3) becomes the **exact constraint satisfaction problem** (ECSP)

$$BF(x) \in \mathbf{b} \quad x \in \mathbf{x}. \tag{4}$$

The traditional constraint satisfaction problem (2) is obtained from (4) if we take $w = m$, $B = I_m$ and $\mathbf{F} = \mathbf{b}$. From the point of view of solvability, (4) and (2) are equivalent, as one can redefine $\widetilde{F}(x) := BF(x)$. However, from a computational point of view, the form (4) has advantages that typically lead to improved linear relaxations once B is not the identity matrix.

Any CSP with uncertain coefficients can be brought into the above form of an UCSP, by introducing new variables for every subexpression composed of a product with an uncertain coefficient or a linear combination in which a coefficient is uncertain. As an example we consider the nonlinear, exact constraint satisfaction problem

$$x_1 + e^{0.1x_1+0.2x_2^2} \le 1,\ x_1 \in [-1,1],\ x_2 \in [-2,0]. \tag{5}$$

Since the decimal numbers are not exactly representable as floating-point numbers, it must be represented internally as an inexact CSP. By introducing the intermediate variable $x_3 = 0.1x_1 + 0.2x_2^2$ we obtain

$$0.1x_1+0.2x_2^2-x_3 = 0,\ \ x_1+e^{x_3} \le 1,\ \ x_1 \in [-1,1],\ \ x_2 \in [-2,0],\ \ x_3 \in [-\infty,\infty], \tag{6}$$

and end up in

$$BF(x) \in [\infty,1],\ F(x) := \begin{pmatrix} x_1 \\ x_2^2 \\ x_3 \\ e^{x_3} \end{pmatrix},\ x \in \mathbf{x} := \begin{pmatrix} [-1,1] \\ [-2,0] \\ [-\infty,\infty] \end{pmatrix},\ B := \begin{pmatrix} 0.1 & 0.2 & -1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}. \tag{7}$$

In floating point arithmetic the coefficient 0.1 cannot represented, therefore the exact problem (7) becomes an uncertain problem with

$$\begin{gathered} BF(x) \in [\infty,1],\quad F(x) := \begin{pmatrix} x_1 \\ x_2^2 \\ x_3 \\ e^{x_3} \end{pmatrix},\ x \in \mathbf{x} := \begin{pmatrix} [-1,1] \\ [-2,0] \\ [-\infty,\infty] \end{pmatrix}, \\ B \in \mathbf{B} := \begin{pmatrix} [\nabla 0.1, \Delta 0.1] & [\nabla 0.2, \Delta 0.2] & [-1,-1] & [0,0] \\ [1,1] & [0,0] & [0,0] & [1,1] \end{pmatrix}, \end{gathered} \tag{8}$$

The definition of feasibility for the traditional CSP given above does not make sense for the uncertain constraint satisfaction problem (3). For example, in the UCSP

$$ax_1 + x_2 = 1,\quad x_1 \in [-1,1],\quad x_2 \in [-2,0],\quad a \in [0.79,0.81], \tag{9}$$

no single point can be feasible since it cannot satisfy (9) for all $B \in \mathbf{B}$. But the problem should not be classified as infeasible since, e.g., $x_1 = 1 - a$, $x_2 = 1$ should be considered as a coefficient-dependent solution. Since a is uncertain, this "solution" comprises the set $\{x \in \mathbb{R}^2 \mid x_1 \in [0.19, 0.21],\ x_2 = 1\}$. Therefore we must generalize the definition:

Definition 1 A set $Z \subseteq \mathbf{x}$ is called **feasible** for the uncertain constraint satisfaction problem (3) if, for all $B \in \mathbf{B}$, there is an $x \in Z$ with $BF(x) \in \mathbf{F}$, **infeasible** if $BF(x) \notin \mathbf{F}$ for all $B \in B$ and $x \in Z$, and **partially feasible** otherwise. The problem (2) is called **feasible** (**infeasible**) if $\mathbf{x}$ is feasible (infeasible). The **solution set** of (3) is the set

$$\widehat{Z} := \{x \in \mathbf{x} \mid \exists B \in \mathbf{B} : BF(x) \in \mathbf{b}\}$$

of all feasible or partially feasible points of (3).

The definition implies that if the set Z is feasible then all sets $Z' \subseteq \mathbf{x}$ containing Z are also feasible. In particular, the definition applies to boxes $\mathbf{z}$, and a feasible set exists iff the box $\mathbf{x}$ is feasible, i.e., iff the problem itself is feasible. The solution set is nonempty iff $\mathbf{x}$ is feasible or partially feasible.

Regarding the example (9), by Definition 1 the box $\mathbf{z}_1 := ([1.2, 1.27]\ [0, 0])^T$ is feasible, the box $\mathbf{z}_2 := ([0.9, 0.95]\ [0, 0])^T$ is infeasible and the box $\mathbf{z}_3 := ([1.25, 1.25]\ [0, 0])^T$ is partially feasible. The problem (9) is feasible since $\mathbf{z}_1$ is feasible, $\mathbf{z}_1 \subset \mathbf{x}$ and thus the box $\mathbf{x}$ is feasible.

3 Finding and using approximately feasible points

In this section we discuss several aspects of finding and using approximately feasible points. In Subsection 3.1 we define δ-feasibility of approximate solution and discuss methods for finding them. In Subsection 3.2 we give a method for choosing of starting point in a new subbox, while in Subsection 3.3 we discuss the δ-feasible solution search by branch and bound.

The result of this section will be useful in Section 4 and 5, where we need δ-feasible solutions in order to to find a (narrow) box close to them which provably contains a feasible point.

3.1 Finding approximately feasible solutions

Given the uncertain constraint satisfaction problem (3), we use the minimal point (1) to define the vector valued **feasibility measure**

$$\Delta_{\mathbf{B}}(x) := \mu(\mathbf{B}F(x) - \mathbf{b}) \tag{10}$$

of a point $x \in \mathbb{R}^n$. For a given positive definite, diagonal scaling matrix D, the number

$$d(x) := \|D\Delta_{\mathbf{B}}(x)\|_2 \tag{11}$$

is called the **feasibility distance** of x for the constraint satisfaction problem (3).

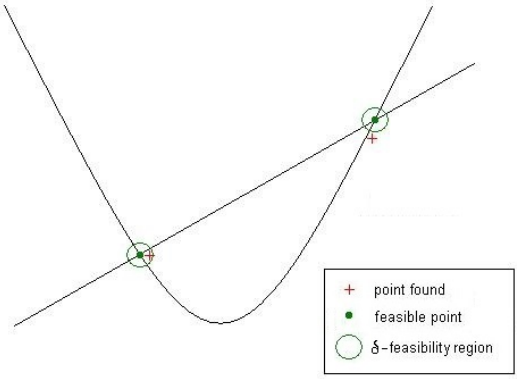

Fig. 1 Finding approximately feasible points of a problem consisting two equality constraints: one of the point found is δ-feasible (but not feasible) the other is infeasible (since it is outside of the δ-feasibility region).

A point x is called **δ-feasible** if $x \in \mathbf{x}$ and $d(x) \leq \delta$, where $\delta > 0$ is a **feasibility tolerance**. In particular, feasible points are δ-feasible for every $\delta > 0$. In the example of Figure 1, the set of δ-feasible points consists of the union of the two circles shown, classifying just one of the two marked points as δ-feasible.

Since in floating point arithmetic computations not exact, the minimum of the feasibility distance function found by the local search may be greater than zero even if the problem is feasible. As shown in Figure 1 the solutions (feasible points) of two equality constraints are missed by the points found by minimizing the feasibility distance function. Intuitively if a point found $\widetilde{x}$ is near enough to a feasible point we could say that it is δ-feasible. Even when the computations are exact, not each real number can be represented as a floating point number. For example, if the CSP consists of two linear constraints in $\mathbb{R}^2$ intersecting in $z = (0.1, 0.1)$, only a δ-feasible point $\widehat{z}$ near z can be found since the number 0.1 has no exact representation in finite, binary arithmetic.

In order to find a feasible point of (3) inside a given box $\mathbf{x}_0$ one can minmize the feasibility distance function by solving the bound constrained optimization problem

$$\begin{array}{l} \min\ f(x) := \frac{1}{2}\|D\Delta_B(x)\|_2^2 \\ \text{s.t.}\ \ x \in \mathbf{x}_0 \subseteq \mathbf{x} \end{array} \tag{12}$$

for some fixed $B \approx \text{mid}(\mathbf{B})$ in place of $\mathbf{B}$. This ensures that the objective function is continuously differentiable if F is, with gradient

$$\nabla f(x) = (BF)'(x)^T D^2 \Delta_B(x). \tag{13}$$

Suitable methods for choosing a starting point $x_0 \in \mathbf{x}_0$ are discussed in Subsection 3.2. Then local optimization techniques (see, e.g., [17]) applied to (12) yield a point $\widetilde{x} \in \mathbf{x}_0$ that approximately has the minimal feasibility distance in a neighborhood.

In case we only have an unconstrained solver to our disposal we cannot solve (12) directly, but we can still solve the corresponding unconstrained problem

$$\min \quad f^u(x) := \frac{1}{2}\Big(\|D\Delta_B(x)\|_2^2 + \|D'\mu(\mathbf{x}_0 - x)\|_2^2\Big), \tag{14}$$

where D' is another positive definite diagonal scaling matrix. Again, $f^u(x)$ is continuously differentiable if F is, with gradient

$$\nabla f^u(x) = (BF)'(x)^T D^2 \Delta_B(x) + (D')^2 \mu(\mathbf{x}_0 - x) \tag{15}$$

Since the computed solution $\widetilde{x}'$ of (14) need not satisfy the condition $x \in \mathbf{x}$, we project it to the box $\mathbf{x}$, resulting in the point $\widetilde{x}$ with

$$\widetilde{x}_i := \min(\overline{x}_i, \max(\underline{x}_i, \widetilde{x}_i)) \quad \text{for } i = 1, \ldots, n.$$

Finally, no matter how $\widetilde{x}$ was obtained, we compute $d(\widetilde{x})$ (using $\mathbf{B}$, not B) to check if the point $\widetilde{x}$ is δ-feasible. If this is not the case, the feasibility search failed, which may (or may not) be an indication that the current subbox $\mathbf{x}_0$ is infeasible. This suggests that the information contained in the point might be used to prune the box $\mathbf{x}_0$. A technique for doing this will be discussed in [7].

Let $\widetilde{x} \in \mathbf{x}_0$ be a δ-feasible point of (3) ($d(\widetilde{x}) \leq \delta$) and let

$$I := \{i \mid \underline{F}_i \neq \overline{F}_i\}$$

with $n_s := |I|$ be the index set of the non-equality constraints of (3). If we introduce additional n_s slack variables x^s addition to the n original variables x, then the problem (3) in the equality form can be posed as

$$EF(x, x^s) = 0, \quad x \in \mathbf{x}, \quad x^s \in \mathbf{F}_I, \quad E \in \mathbf{E}, \tag{16}$$

where the mapping F is extended by linear terms for the new slack variables, formally

$$F(x, x^s) := (F(x)^T \; x_1^s \; \ldots \; x_{n_s}^s)^T.$$

In this case the point found $\widetilde{x}$ must be extended by the value of the slack variables at $\widetilde{x}$ to match the equality form (16) resulting in

$$z := \begin{pmatrix} \widetilde{x} \\ B_{I:}F(\widetilde{x}) \end{pmatrix}. \tag{17}$$

The equality form (16) and the corresponding δ-feasible point z is needed for verifying feasible points of constraint satisfaction problems presented in Sections 4 and 5. If the point $\widetilde{x}$ is not δ-feasible it can be still useful as discussed in the following subsection.

3.2 Choice of starting point in a new subbox

The points found by the local search – even if they are not sufficiently feasible – hold valuable information. In this subsection we again assume that we have found an unsatisfactory $\widetilde{x}$, but we use the results of the local solver in a different manner. Considering that we have some other methods to reduce or split $\mathbf{x}$ (e.g., a branch and bound scheme) it might be useful to find another, more feasible point in a reduced or splitted box but minimize the efforts needed for the new search.

The following table summarizes the possible statuses obtained from the local search resulting in the point $\widetilde{x}$ and indicates if it should be re-used as a starting

point or not. The statuses depend on the results of the local solver, the feasibility distance and the possible projection performed:

status of the point found	code	reuse
the point found is a δ-feasible solution	`f`	yes
the solver reached maximum number of iterations	`m`	yes
the point is random point	`r`	yes
the point has been projected	`p`	yes
the solver found an infeasible local minimum	`l`	no
the solver produced an error	`e`	no

(18)

As status `p` indicates that the point was found by an unconstrained solver and has been projected inside the box. Re-using means that $\widetilde{x}$ is taken as as a starting point x_0 for the new local search for finding feasible points (discussed in Subsection 3.1). Not re-using means that we choose another starting point $x_0 \in \mathbf{x}$ (e.g. randomly but not too near to $\widetilde{x}$) for the new search. Note that depending on the different statuses other more sophisticated strategies for re-using or not re-using can be developed.

Combining the above with the a reduction step and adding natural safeguards we propose the following algorithm.

Algorithm: 1 *(***Feasible point search***).*

Inputs: *A constraint satisfaction problem given by* (3)*, a box* $\mathbf{x}_0 \subseteq \mathbf{x}$*, the feasibility tolerance* $\delta \geq 0$*, the starting point* x_0 *with status* s_0 *chosen from Table* (18) *and feasibility distance* $d_0 := d(x_0)$*.*
Outputs: *(*x*,* s*,* d*), where* x *is a point,* s *the corresponding status code and* d *the feasibility distance* $d(x)$*.*

1. *If the status of the starting point is either* $s_0 =$ `e` *(error) or* $s_0 =$ `l` *(local minimum) choose another* $x_0 \in \mathbf{x}$*, set* $s_0 =$ `r` *and recompute feasibility distance* $d_0 = d(x_0)$*.*
2. *Starting from* x_0 *use either a bound constrained local solver to solve* (12) *or an unconstrained local solver to solve* (14)*. The solution* $\widetilde{x}$ *and the solver status* s *are obtained.*
3. *If* $\widetilde{x} \notin \mathbf{x}_0$ *project* $\widetilde{x}$ *into the box* $\mathbf{x}_0$ *and change the status* $s =$ `p`*.*
4. *(a) If* $\widehat{d} > \delta$ *and* $d_0 < \widehat{d}$ *revert to the starting point by setting* $\widetilde{x} = x_0$*,* $s =$ `l`*,* $\widehat{d} = d_0$ *and go to Step 1.*
   (b) Otherwise return $(\widetilde{x}, s, \widehat{d})$*. If* $\widehat{d} \leq \delta$ *the point found is* δ*-feasible.*

Algorithm (1) can be used in combination with filtering methods like constraint propagation (e.g., see [5]) or linear relaxations (see, e.g., [6]). It can be also embedded in a branch and bound scheme as discussed in the next subsection.

3.3 Feasible point search in branch and bound

In this subsection we discuss how to integrate Algorithm 1 from Subsection 3.2 into a branch and bound scheme. This results in a cheap and fast global method for finding δ-feasible points of feasible problems where Algorithm 1 would have failed.

The points found during the search, which are the best (have smallest feasibility distance) are remembered for each box. This information is stored as a quadruple

$$(\mathbf{x},\ \widetilde{x},\ d(\widetilde{x}),\ \texttt{status}), \tag{19}$$

where $\mathbf{x}$ is a box, $\widetilde{x} \in \mathbf{x}$ is the point found, $d(\widetilde{x})$ is the feasibility distance of the point found and `status` is the status code according to Table 18.

When the box $\mathbf{x}$ is divided into subboxes the position of $\widetilde{x}$ decides to which part the information from (19) belongs, namely in the part(s) $\mathbf{x}'$ with $\widetilde{x} \in \mathbf{x}'$. According to this we set $(\mathbf{x}',\ \widetilde{x},\ d(\widetilde{x}),\ \texttt{status})$ for the subbox(es) $\mathbf{x}'$ and for all other parts the status is set to `e`.

In order to guarantee a finite termination of the following algorithm we define a **narrow condition for a box**; if this condition is met for a box $\mathbf{x}$ the box is not split again. We call a box $\mathbf{x}$ **σ-narrow** (for some constant $\sigma > 0$) if

$$\overline{x}_i - \underline{x}_i < \sigma \text{ for all } i = 1, \ldots, n. \tag{20}$$

The subdivision method has to match the narrow condition; a narrow condition and a subdivision method are sound when all boxes in the branch and bound scheme will become narrow sooner or later. A subdivision method matching (20) would be to divide $\mathbf{x}$ into two parts by splitting the widest component of $\mathbf{x}$ at the midpoint.

Assuming we have specified a sound narrow condition and subdivision method, the following algorithm is finite and makes a global search for a δ-feasible point.

Algorithm: 2 *(***Improved feasible point search***)*

Inputs: *A constraint satisfaction problem given by* (3)*, the tolerance* $\delta \geq 0$ *and the box* $\mathbf{x}$*.*
Outputs: *(*x*,* d*,* S*), where* x *is a point,* d *is the corresponding feasibility distance* $d(x)$ *and* S_t *is a set of boxes, called the "narrow box storage".*
Results:

- *If* $d \leq \delta$ *we have found the* δ*-feasible point* x*.*
- *If* $d > \delta$ *and the narrow box storage* S *is empty the problem is infeasible.*
- *Otherwise we were not able to find a feasible point, but the problem can be still feasible. In this case we may start the algorithm again with smaller* δ*, putting all the boxes from* S *to the stack and setting* $\mathbf{x} = \emptyset$*.*

1. *Set* x_b *(the most feasible point found so far) as a random point of the box* $\mathbf{x}$ *and its feasibility distance as* $d_b = d(x_b)$*. Initialize both the stack and* S *as empty.*
2. *If the box* $\mathbf{x}$ *is empty:*
   (a) *If the stack is not empty get the last box from it and place it into* $\mathbf{x}$*.*
   (b) *If the stack is empty terminate the algorithm and returning (*x_b*,* d_b*,* S*). In this case no* δ*-feasible point was found.*
3. *If the box* $\mathbf{x}$ *has an assigned search information, place this information into* x_0*,* s_0 *and* d_0 *otherwise initialize* x_0 *as a random point of the box* $\mathbf{x}$ *and set* $s_0 = \mathtt{r}$*,* $d_0 = d(x_0)$*.*
4. *Use filtering methods (constraint propagation, linear relaxation, etc.) on the box* $\mathbf{x}$ *obtaining a new box* $\mathbf{x}'$*.*
5. *If* $\mathbf{x}'$ *is empty set* $\mathbf{x}$ *as empty and go to Step 2.*
6. *Otherwise:*
   (a) *Apply Algorithm 1 with* $\mathbf{x}'$*,* δ*,* x_0*,* s_0 *and* d_0 *resulting in (*$\widetilde{x}$*,* s*,* d*).*

(b) If $d < d_b$ then the new point is more feasible than the most feasible one found so far, therefore set $x_b \leftarrow \widetilde{x}$ and $d_b \leftarrow d$.

(c) One of the following actions is taken depending on the point $\widetilde{x}$ and the box $\widehat{\mathbf{x}}$:

i. If $d \leq \delta$, a δ-feasible point was found, terminate the algorithm returning $(\widetilde{x}, d, S)$.

ii. Else if the box $\widehat{\mathbf{x}}$ is δ-narrow, it is put into the narrow box storage S. We also assign the search information $(\widetilde{x}, s, d)$ to $\widehat{\mathbf{x}}$, set the box $\mathbf{x}$ as empty and go to Step 2.

iii. Otherwise subdivide $\mathbf{x}'$ and assign the search information $(\widetilde{x}, s, d)$ to the part in which the point $\widetilde{x}$ is contained. Set one part in $\mathbf{x}$ and push the remaining parts on the stack. Go to Step 2.

4 Discussion of existing verification methods

In this section we consider the decision problem whether the continuous nonlinear system

$$E(x) = 0, \quad F(x) \leq 0, \quad x \in \mathbf{x} \tag{21}$$

with, $E : D_e \subseteq \mathbb{R}^n \to \mathbb{R}^{m_e}$ and $F : D_f \subseteq \mathbb{R}^n \to \mathbb{R}^{m_f}$ has a solution $x^* \in \mathbf{z}$ with $E(x^*) = 0$ and $F(x^*) \leq 0$ for a suitable $\mathbf{z} \subseteq \mathbf{x}$.

In applications this problem arises when a δ-feasible solution has been found and one tries to find a (narrow) box close to it which provably contains a feasible point (see Figure 2).

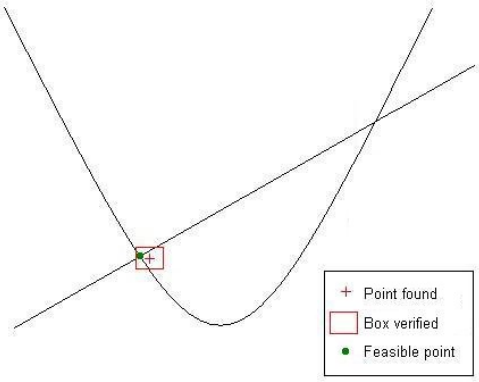

Fig. 2 Verifing approximately feasible points of a problem consisting two equality constraints: we define a box (box verified) around the δ-feasible point found (point found) and verify that it contains a feasible point. Since in this setting the box contains a feasible point the test is expected be positive.

4.1 Nonsingular systems of nonlinear equations

Problem (21) can be solved by the so called interval Newton methods, which, at the most fundamental level, are computational versions of Brouwer's fixed point theorem:

Theorem 3 (Brouwer) *Let D be a convex and compact subset of $\mathbb{R}^n$ with $\mathrm{int}(D) \neq \emptyset$ then every continuous mapping $G : D \to D$ has at least one fixed point $x^* \in D$, i.e. a point with $G(x^*) = x^*$*

We consider (21) with $m_e = n$ and $m_i = 0$. We also assume that E is Lipschitz continuous in $D_0 \in D$ i.e. there is an $l \in D_0$ such that $|E(x) - E(x)| \leq l^T|x - y|$ for all $x, y \in D_0$. In addition to this we assume that $\mathbf{A}$ is a **slope matrix** of E at $[\mathbf{x}, \mathbf{y}]$ (see. e.g., [14]), formally that for all $x \in \mathbf{x} \subseteq D$, $y \in \mathbf{y} \subseteq D$ the equation

$$E(x) - E(y) = A(x - y)$$

holds for some $A \in \mathbf{A}$. If E is differentiable on $\mathbf{x}$ the matrix $\mathbf{A}$ can be chosen as the **interval extension of the Jacobian** $E'(\mathbf{x})$ but in general there is a slope matrix at $[x^*, \mathbf{x}]$ with roughly half the radius.

Let A^H denote the hull inverse $\mathbf{A}^H\mathbf{b} := \square\{A^{-1}b,\ A \in \mathbf{A},\ b \in \mathbf{b}\}$ of $\mathbf{A}$ then every $\mathbf{z}$ satisfying

$$N(\mathbf{x}, \widetilde{x}) := \widetilde{x} - \mathbf{A}^H E(\widetilde{x}) \subseteq \mathbf{z} \tag{22}$$

has the following properties:

1. Every zero $\mathrm{t}x^* \in \mathbf{x}$ of E satisfies $x^* \in \mathbf{z}$.
2. If $\mathbf{x} \cap \mathbf{z} = \emptyset$ then E has no zero in $\mathbf{x}$.
3. If $\mathbf{z} \in \mathrm{int}(\mathbf{x})$ and $\mathbf{z} \subseteq \mathbf{x}$ then $\mathbf{z}$ contains a unique zero of E.

(23)

The operator $N(\widetilde{x}, \mathbf{x})$ defined by (21) is called the (optimal) **interval Newton operator**. While the newton operator can be used only if $\mathbf{A}$ is regular, there are two other interval Newton like operators with the property (23), which require only weaker assumptions, the **Krawczyk operator**

$$K(\mathbf{x}, \widetilde{x}) := \widetilde{x} - CF(\widetilde{x}) - (C\mathbf{A} - I)(\mathbf{x} - \widetilde{x}), \tag{24}$$

and the **Hansen-Sengupta operator**

$$H(\mathbf{x}, \widetilde{x}) := \widetilde{x} + \Gamma(C\mathbf{A},\ -CF(\widetilde{x}),\ \mathbf{x} - \widetilde{x}), \tag{25}$$

where C is a preconditioner and the operator Γ is defined as

$$\Gamma(\mathbf{a}, \mathbf{b}, \mathbf{x}) := \square\{x \in \mathbf{x} \mid ax = b \text{ for some } a \in \mathbf{a} \text{ and } b \in \mathbf{b}\} \tag{26}$$

(see, e.g., [16]) and has the property that $\Gamma(\mathbf{a}, \mathbf{b}, \mathbf{x}) = \emptyset$ if $\mathbf{b} = \emptyset$ or $\mathbf{x} = \emptyset$. Note that the Hansen-Sengupta operator (and the Krawczyk operator) do not require the computation of an inverse an thus they can be applied to rectangular systems.

In addition to this we have the **general interval Newton operator**

$$N_G(\mathbf{x}, \widetilde{x}) := \widetilde{x} - (C\mathbf{A})^I(CF(\widetilde{x})), \tag{27}$$

where C is a preconditioner and $(C\mathbf{A})^I$ denotes the inverse of $C\mathbf{A}$. The problem (21) can be solved by the interval Newton iteration

$$\mathbf{x}^0 = \mathbf{x},\ \mathbf{x}^{l+1} := I(\mathbf{x}^l, \widetilde{x}^l) \cap \mathbf{x}^l,\ \widetilde{x}^l \in \mathbf{x}^l,\ \text{with } (l = 0, 1, \dots), \tag{28}$$

where $\widetilde{x}^l \in \mathbf{x}^l$ and $I(\mathbf{x}, \widetilde{x}) \in \{N(\mathbf{x}, \widetilde{x}),\ K(\mathbf{x}, \widetilde{x}),\ H(\mathbf{x}, \widetilde{x}),\ N_G(\mathbf{x}, \widetilde{x})\}$ can be chosen arbitrarily.

Note that the methods discussed in this and the next section apply only to systems of equations but it is possible to transform the inequalities of (21) to equalities by introducing slack variables.

4.2 Systems of nonlinear equations

We consider (21) with $0 < m_e \leq n$ and $m_i = 0$. Let $\widetilde{x} \in \mathbf{x}$ an approximate solution of (21) and let S denote an index set of the selected variables with $S \subseteq \{1, \dots, n\}$ and $|S| = m$. The solution method given in [9] can be summarized as follows:

Algorithm: 4 *(***Verification method by Kearfott***)*

1. *Let J be the Jacobian of E at $\widetilde{x}$ and let $J^O \in \mathbb{R}^{n \times n-m}$ be the matrix built columnwise from the basis vectors of the null space* $\mathrm{null}(J) := \{x \in \mathbb{R}^n \mid Jx = 0\}$ *of the matrix J. Then the index set S with $|S| = m$ is selected such that the indices $i \in S$ correspond to the variable indices where the row sum of J^O is minimal. This heuristic should maximize the chance that after fixing the variables x_i for $i \notin S$ (in Step 2), the linear subspace in which the Hansen-Sengupta operator is applied (in Step 3) contains a feasible point.*
2. *A box $\mathbf{z}$ is constructed around the approximate solution $\widetilde{x}$ by*

$$\mathbf{z}_i = \begin{cases} [\widetilde{x}_i - \epsilon, \widetilde{x}_i + \epsilon] & \text{if } i \in S \\ \widetilde{x}_i & \text{if } i \notin S \end{cases}, \tag{29}$$

   where ϵ is a small positive number on the order of machine epsilon. The box $\mathbf{z}$ should be sufficiently narrow to avoid overestimation of the constraint gradient ranges by the interval evaluations used to compute $\mathbf{A}$ in Step 3. It should be also large enough, that the solution set of (21) *can pass through it.*
3. *The matrix $\mathbf{A}$ is chosen as the interval extension of the Jacobian of E at $\mathbf{z}$, the preconditioner matrix $C \in \mathbb{R}^{m \times m}$ is chosen approximately as* $\mathrm{mid}(\mathbf{A}_{:S})^{-1}$ *and the Hansen-Sengupta operator* (25) *is used to obtain a new box $\widehat{\mathbf{z}} := H(\mathbf{z}, \widetilde{x})$.*
4. *By* (23)*, if $\widehat{\mathbf{z}} \subseteq \mathrm{int}(\mathbf{z})$ holds, it is proved that the box $\widehat{\mathbf{z}}$ contains a solution of* (21) *and if $\widehat{\mathbf{z}} = \emptyset$ then $\mathbf{z}$ contains no solution.*
5. *Otherwise an ϵ-inflation procedure (see e.g., [18]) is used on $\mathbf{z}_S$ to obtain $\widetilde{\mathbf{z}}$, and provided that $\widetilde{\mathbf{z}} \subseteq \mathbf{x}$ we return to the Step 3 using $\widetilde{\mathbf{z}}$ in place of $\mathbf{z}$.*

4.3 Systems of nonlinear inequalities

We note that Algorithm 4 is an modified version of Hansen's method [8, **S**12]. As Kearfott noticed the algorithm is likely to fail if the solution $x_i^* \approx \underline{x}_i$ or $x_i^* \approx \overline{x}_i$ for an unfixed coordinate $i \in S$. This can be avoided by projecting the corresponding

coordinates of the approximately feasible solution $\widetilde{x}$ onto the active set of the bound constraints and led to a new improved method given in [11].

We consider (21) with $m_e = 0$ and $0 < m_i$. Let $\widetilde{x} \in \mathbf{x}$ denote an approximate solution of (21), then task is to find a solution $x^* \in \mathbf{x}$ with $F(x^*) = 0$ near to $\widetilde{x}$. Verifying if $\widetilde{x}$ is a valid solution of (21) is possible by evaluating the interval extension of F at $[\widetilde{x}, \widetilde{x}]$, formally

$$F([\widetilde{x}, \widetilde{x}]) \leq 0 \implies F(\widetilde{x}) \leq 0, \tag{30}$$

which holds because the inclusion isotonicity of the interval arithmetic. This however usually does not work if $F(\widetilde{x})_i \approx 0$ for some $i \in \{1, \ldots, n\}$. For example if the constraint $f(x) < 0$ with $f(x) := 0.1x - 0.1$ is evaluated in binary, floating point, interval arithmetic at the obvious solution $[1, 1]$, we obtain $f([1, 1]) = [\nabla(0.1), \Delta(0.1)] - [\nabla(0.1), \Delta(0.1)] \not\leq 0$. Therefore it is crucial that the approximate solution $\widetilde{x}$ is slightly moved into the solution set of (21), allowing the verification by (30).

According to [11] we denote the approximately active constraints of (30) by $G_i(x) \leq 0$ with $i \in \{1, \ldots, m_i\}$ where

$$G_i(x) := \begin{cases} F_j(x) & \text{if } F_j(\widetilde{x}) \approx 0, \text{ or} \\ x_k - \overline{x}_k & \text{if } \widetilde{x}_k \approx \overline{x}_k, \text{ or} \\ \overline{x}_l - x_l & \text{if } \widetilde{x}_l \approx \underline{x}_l, \end{cases} \tag{31}$$

and for the transposed Jacobian matrix of $G(x)$ at $\widetilde{x}$ of $A^\iota \in \mathbb{R}^{n \times m_i}$ we compute the QR factorization $A^\iota := Q^\iota R^\iota$ of A^ι, then the following algorithm can be used to create a vector $v \in \mathbb{R}^n$ which likely points from $\widetilde{x}$ into the interior of the feasible region of (21):

Algorithm: 5 *(***Producing a direction into the feasible region (Kearfott)***)*

1. *If* $Q^{\iota T}_{:1} A^\iota_{:1} < 0$ *then* $Q^\iota_{:1} \leftarrow -Q^\iota_{:1}$.
2. *Set* $v = -Q^\iota_{:1} \mathrm{sign}(R^\iota_{11})$
3. *For* $j = 2 \ldots m_i$ *do:*
   (a) *Set* $\delta = Q^{\iota T}_{:k} A^\iota_{:k}$.
   (b) *If* $\delta < 0$ *then set* $Q^\iota_{:k} \leftarrow -Q^\iota_{:k}$ *and* $\delta \leftarrow -\delta$.
   (c) *Set* $\alpha = (2v^T A^\iota_{:k})/\delta$.
   (d) *If* $\alpha > 0$ *then set* $u = v - \alpha Q^\iota_{:k}$ *and* $v \leftarrow u/\|u\|_2$.
4. *Return* v

As proven in [11] Suppose that at $\widetilde{x} \in \mathbf{x}$ only $m_i \leq n$ constraints are active, then for a sufficiently small δ, $x^* := \widetilde{x} + \delta v$ is strictly feasible with respect to the constraints (21). In this case the feasibility of x^* can be verified by applying (30).

4.4 General constraint satisfaction problems

In this subsection we consider (21) with $0 < m_e \leq n$ and $0 < m_i$. In this case could apply the technique in Subsection 4.2 where the inequality constraints are transformed to equalities by introducing slack variables.

The other possibility is using the Kearfott's method which is based on subsections 4.2 and 4.3 as well as on [11]: Suppose that there are m_i active inequality constraints and that $m_e \leq n - m_i$. Let $G(x)$ as defined in (31) and let $A^\iota \in \mathbb{R}^{n \times m_i}$

be the transposed Jacobian matrix of $G(x)$ at $\widetilde{x}$. Similarly, let $A^{\varepsilon} \in \mathbb{R}^{n \times m_e}$ be the transposed Jacobian matrix of $E(x)$ at $\widetilde{x}$. We construct two orthogonal matrices Q^{ι} and $\widehat{Q}^{\varepsilon}$ by performing two QR factorizations. The matrix Q^{ι} is constructed such as that it will give us directions tangent to the equality constraints, in which we can perturb $\widetilde{x}$ into the region that is feasible with respect to the inequality constraints and is obtained by

$$A := (A^{\varepsilon}\ A^{\iota}) = QR = (Q^{\varepsilon}\ Q^{\iota}\ Q^{O})R. \tag{32}$$

The matrix $\widehat{Q}^{\varepsilon}$ is constructed such as its column vectors form a basis of the space spanned by the column vectors of A^{ε} but within the tangent space of the inequality constraints and is obtained by

$$\widehat{A} := (A^{\iota}\ A^{\varepsilon}) = \widehat{Q}\hat{R} = (\widehat{Q}^{\iota}\ \widehat{Q}^{\varepsilon}\ \widehat{Q}^{O})\widehat{R}. \tag{33}$$

Then according to [11] we define:

Algorithm: 6 *(***Improved verification method of Kearfott***)*

1. *Let* $\widetilde{x} \in \mathbf{x}$ *an approximate solution of* (21).
2. *If* (30) *cannot verify* $F(\widetilde{x}) \leq 0$*, use* (32) *to obtain* Q^{ι}*. Then apply Algorithm 5 with this* Q^{ι} *in order to produce a vector* v *pointing into the solution set of the inequality constraints but tangent to the equality constraints. Finally set* $\widetilde{x} \leftarrow \widetilde{x} + \delta v$ *for a suitably chosen* δ*. If* (30) *still cannot verify* $F(\widetilde{x}) \leq 0$ *signal failure.*
3. *Create and factor* $\widetilde{A}$ *as in* (33) *to obtain* $\widehat{Q}^{\varepsilon}$*.*
4. *Use an interval Newton iteration as given in* (28) *with* $\mathbf{x}$ *defined by* $\mathbf{x}_i := [-\epsilon/10, \epsilon/10]$ *for suitable small* ϵ *and with* $\widetilde{x}^l \approx \operatorname{mid}(\mathbf{x}^l)$ *on the square system of equations* $\psi(x) = 0$*, where the function* $\psi : \mathbb{R}^{m_e} \to \mathbb{R}^{m_e}$ *is defined as*

$$\psi(x) = F\Big(\widetilde{x} + \sum_{i=1}^{m_e} x_i \widehat{Q}^{\varepsilon}_{:i}\Big). \tag{34}$$

   The iteration either converges to a nonempty box $\widehat{\mathbf{x}} \in \mathbb{IR}^{m_e}$ *with* $\widehat{\mathbf{x}} \subseteq \operatorname{int}(\mathbf{x})$ *or we signal failure.*
5. *Form the box* $\widetilde{\mathbf{x}} := \widetilde{x} + \sum_{i=1}^{m_e} \widehat{\mathbf{x}}_i \hat{Q}^{\varepsilon}_{:i}$ *and use* (30) *to check the feasibility of the inequality constraint, i.e., check if* $F(\widetilde{\mathbf{x}}) \leq 0$*. If the verification succeeds return the narrow box* $\widehat{\mathbf{x}}$ *containing a feasible solution of* (21)*, otherwise signal failure.*

5 New verification method

In this section we discuss a new method for verifying approximately feasible points. The method applies the preconditioning directly to the original system, simultaneously finding a suitable subset of the variables which are not fixed in the verification process. The optimal values for the remaining variables are computed by linearly relaxing the problem and solving a suitable linear program.

Let the constraint satisfaction problem be given in the equality form (possibly by introducing slack variables) as in (16), namely

$$EF(x) = 0,\ x \in \mathbf{x},\ E \in \mathbf{E}. \tag{35}$$

Let $\widetilde{x} \in \mathbf{x}$ be a point with $EF(\widetilde{x}) \approx 0$. This $\widetilde{x}$ can be constructed by finding a δ-feasible point in the from of (3) by solving (12) and then transforming it to match the equality form (16) (for details see the end of Subsection 3.1).

Let s be a scaling vector for the variables and let $\delta > 0$. Then

$$\mathbf{z} := \mathbf{x} \cap [\widetilde{x} - \delta s, \widetilde{x} + \delta s],$$

is a narrow box around $\widetilde{x}$ which is contained in the box $\mathbf{x}$. For quadratic problems the variable scaling vector s can be obtained by the method described in [4] while the constant δ should be chosen reasonably small. The box $\mathbf{z}$ around the δ-feasible point $\widetilde{x}$ may contain a feasible point. To verify whether this is true or not we discuss a test for deciding if there is a feasible point contained in $\mathbf{z}$. If the test succeeds we know that there is a $y \in \mathbf{z}$ such that the equality $EF(y) = 0$ is satisfied, proving that $\mathbf{z}$ contains at least one feasible point without actually finding this point (see Figure 2).

Let $C \in \mathbb{R}^{m \times m}$ and let S be the index set of the selected variables with $S \subseteq \{1, \ldots, n\}$ and with $|S| = m$. Then the index set $U := \{1, \ldots, n\} \backslash S$ of the unselected variables has $|U| = n - m$. We define the function $p : \mathbb{R}^n \to \mathbb{R}^m$ with

$$p(x) := x_S - (CE)F(x). \tag{36}$$

If we compute the linear relaxation of $p(x)$ according to [6], in the box $\mathbf{z}$, we obtain a matrix $\widehat{B}$ and a box $\widehat{\mathbf{b}}$ such that

$$p(x) \in \widehat{B}x + \widehat{\mathbf{b}} \quad \text{for all } x \in \mathbf{z}.$$

Substituting the bounds $\mathbf{z}_S$ for the selected variables x_S results in

$$p(x) \in Bx_U + \mathbf{b} \text{ for all } x \in \mathbf{z}, \tag{37}$$

where $B := \widehat{B}_{:U} \in \mathbb{R}^{m \times n-m}$ and $\mathbf{b} := \widehat{\mathbf{b}} + B_{:S}\mathbf{z}_S \in \mathbb{IR}^m$.

Theorem 7 *Suppose* $\mathbf{b}$ *is bounded and the matrix* C *is non-singular, and suppose that there is a* w *such that*

$$w \in \mathbf{z}_U \in \mathbb{IR}^{n-m}, \quad Bw \in \mathbf{z}_S \ominus \mathbf{b} =: \mathbf{c} \in \mathbb{IR}^m. \tag{38}$$

Then the box $\mathbf{z}' \subseteq \mathbf{z}$ *with*

$$\mathbf{z}' := \begin{cases} \mathbf{z}'_S = \mathbf{v} := Bw + \mathbf{b} \\ \mathbf{z}'_U = [w, w] \end{cases} \tag{39}$$

contains a solution of constraint satisfaction problem (35).

Here ($\mathbf{a} \ominus \mathbf{b} := [\underline{a} - \underline{b}, \overline{a} - \overline{b}]$ defines the **inner subtraction** of two boxes $\mathbf{a}$ and $\mathbf{b}$.

Proof We define the function $\widetilde{p} : \mathbb{R}^m \to \mathbb{R}^m$ by fixing the unselected variables in p to w, as

$$\widetilde{p}(v) := p(y), \quad \text{with } y_S = v,\ y_U = w. \tag{40}$$

Then by (38) we have $y_U = w \in \mathbf{z}_U$ and by reversing the direct subtraction in (38) we also have $y_S \in \mathbf{v} = Bw + \mathbf{b} \subseteq \mathbf{z}_S$. Therefore we find that $y \in \mathbf{z}$ and we can apply (37) to obtain

$$\widetilde{p}(v) = p(y) \in Bw + \mathbf{b} = \mathbf{v},$$

implying that

$$\widetilde{p}(v) \in \mathbf{v} \text{ for all } v \in \mathbf{v}. \tag{41}$$

Since $\mathbf{b}$ is bounded, $\mathbf{v}$ is bounded and the continuous function $\widetilde{p}$ maps the nonempty, compact and convex set $\mathbf{v}$ into itself. The fixed point theorem of Brouwer [24] implies that $\widetilde{p}$ has a fixed point $v^* \in \mathbf{v}$ with $v^* = \widetilde{p}(v^*)$. By (39) and (40) there is a corresponding $x^* \in \mathbf{z}'$ with $x^*_U = w$ and $x^*_S = v^*$, satisfying

$$x^*_S = v^* = \widetilde{p}(v^*) = p(x^*).$$

By (36) we have

$$x^*_S = p(x^*) = x^*_S - CEF(x^*),$$

giving

$$CEF(x^*) = 0. \tag{42}$$

Since the matrix C is assumed to be non-singular, $EF(x^*) = 0$ holds, proving that $x^* \in \mathbf{z}'$ is a solution of the constraint satisfaction problem (35).

A suitable preconditioner C and an index set S can be selected by computing the $m \times n$ interval extension of the Jacobian or the slope matrix $\mathbf{S} := E\mathbf{F}[\widetilde{x}, \mathbf{z}]$ then using the Gauss–Jordan inversion on $\mathbf{S}$ as given in [6]. Note that for the Gauss–Jordan inversion given in [6] the permutation has to be taken in account. It also works for under- or overdetermined systems, however in this application the set index set S must be m-dimensional otherwise the verification will fail. The preconditioning results in the matrix $C\mathbf{S}_{:S} \approx I$, and if $C\mathbf{S}_{:S}$ is a H-matrix then $C\mathbf{S}_{:S}$ and thus C is (strongly) regular, as required by the Theorem 7. The test whether or not the matrix $C\mathbf{S}_{:S}$ is an H-matrix can be done by choosing a suitable $u > 0$ and test whether or not $\langle C\mathbf{S}_{:S} \rangle u > 0$ holds. Different choices of u are

- $u = (1, \ldots, 1)^T$ is the simplest, proving diagonal dominance, but not scaling invariant,
- $u \approx 1/\operatorname{diag}(C\mathbf{S}_{:S})$, is a generally good and cheap choice,
- $u \approx \langle C\mathbf{S}_{:S} \rangle^{-1} (1, \ldots, 1)^T$, is the best choice (see Neumaier [16]), but requires $O(m^3)$ operation for solving the linear system.

Note that by the choice of C, $p(x)$ depends only weakly on the variables x_J since $\partial p(x)/\partial x_k \approx 0$ for all $k \in S$.

In practice, an approximate linear solver can be used to find a solution of (38). First the width $d := \overline{c} - \underline{c}$ of $\mathbf{c}$ should be slightly reduced by setting $\widetilde{\mathbf{c}} := [\underline{c} + \delta d, \overline{c} - \delta d]$ then the linear solver is used to solve (38) with $\widetilde{\mathbf{c}}$ instead of $\mathbf{c}$, obtaining the approximate solution $\widetilde{w}$. Finally

$$w := \min(\max(\underline{z_U}, \widetilde{w}), \overline{z_U}) \in \mathbf{z}_U, \quad \mathbf{v} := Bw + \mathbf{b},$$

is computed. If $\mathbf{v} \subseteq \mathbf{z}_J$ then the box $\mathbf{z}'$ defined by (39) contains a solution of (38).

Algorithm 8 summarizes the results of this section. If the algorithm fails in Step 2 we can start it again with an smaller δ, hoping that the new $\mathbf{S}$ can be fully inverted. On the other hand, choosing a bigger δ increases the chances that the linear program can be solved and the condition $Bw + \mathbf{b} \subseteq \mathbf{x}_J$ still holds.

Algorithm: 8 *(***Verify a feasible point***) Let the constraint satisfaction problem be given in the equality form* (35) *and let* $\widetilde{x} \in \mathbf{x}$ *such that* $EF[\widetilde{x}, \mathbf{z}] \approx 0$.

1. *Find a variable scaling vector* s*, choose a suitable* δ *and compute* $u = \delta s$*,* $\mathbf{z} := \mathbf{x} \cap [\widetilde{x} - u, \widetilde{x} + u]$ *and* $\mathbf{S} := E\mathbf{F}[\widetilde{x}, \mathbf{z}]$*.*
2. *Using the Gauss–Jordan inversion try to find a* C *with* $C\mathbf{S}_{:S} \approx I$*. If* $|S| \neq m$ *or* $C\mathbf{S}_{:S}$ *is not a* H*-matrix, signal failure.*
3. *Define the interval function* $\mathbf{p}(x) := x_S - (C\mathbf{E})F(x)$*.*
4. *Linearize* $\mathbf{p}(x)$ *in the box* $\mathbf{z}$ *obtaining a matrix* $\widehat{B}$ *and an interval vector* $\widehat{\mathbf{b}}$ *such that* $\mathbf{p}(x) \subseteq \widehat{B}x + \widehat{\mathbf{b}}$ *holds for all* $x \in \mathbf{z}$*.*
5. *Compute* $\mathbf{b} := \widehat{\mathbf{b}} + \widehat{B}_{:S}\mathbf{z}_k$ *by interval computations and set* $B := \widehat{B}_{:U}$*.*
6. *Compute* $\mathbf{c}$ *with* $\underline{c} := \underline{z}_J - \underline{b}$ *and* $\overline{c} := -(\overline{b} - \overline{z}_J)$ *using upward rounding as well as* $d := \overline{c} - \underline{c}$ *and* $\widetilde{\mathbf{c}} := [\underline{c} + \delta d, \overline{c} - \delta d]$*.*
7. *Solve the system of two-sided inequalities* $Bw \in \widetilde{\mathbf{c}},\ w \in \mathbf{z}_U$*, using a linear solver obtaining the approximate solution* $\widetilde{w}$*.*
8. *Compute* $w = \min(\max(\underline{x}, \widetilde{w}), \overline{x})$ *and* $\mathbf{v} := Bw + \mathbf{b}$*.*
9. *If* $\mathbf{v} \subseteq \mathbf{x}_J$ *holds, then by Proposition 7 the box* $\mathbf{z}'$ *with* $\mathbf{z}'_J = \mathbf{v}$ *and* $\mathbf{z}'_U = w$ *contains a feasible point. Otherwise signal failure.*
10. *Use a rigorous filtering method (e.g., constraint propagation) on* (35) *with* $x \in \mathbf{z}'$ *in order to reduce the width of* $\mathbf{z}'$*.*

6 Comparison of the verification methods

In this section we present a comparison of the different verification methods discussed in the paper. In order to be fair, all of the methods have been (re)implemented in the same programming language and integrated in our GloptLab solver. This ensues that all of them work on the same problem representation and use the same interval arithmetic, linear algebra implementation etc.

From the whole Coconut Environment Testset (see [23]), we selected all quadratic problems with no more than 300 variables and 300 constraints. The Test Environment [3] provides the best approximate solutions of the resulting 238 problems. For 53 problems the approximate solutions were already feasible, since the rigorous, interval-valued constraint violations computed by the Test Environment were zero. The remaining 185 problems were simplified to a standard quadratic format. Then we constructed a narrow box $\mathbf{z}$ around each approximate solution ($\mathrm{rad}(\mathbf{z}_i) = 10^{-5}$) and applied Interval evaluation(`InEv`), the Krawczyk Interval Newton (`Kraw`), the Hansen-Sengupta Interval Newton (`HaSe`), the basic method of Kearfott (`KeBa`), the improved method of Kearfott (`KeIm`), and our new method (`DoNe`) to the constructed boxes. The results of the attempted verification is summarized in the following table:

Test Result Summary (238 problems)				
method	tried	verified	failed	error
InEv	238	53	185	0
Kraw	185	47	138	0
HaSe	185	41	144	0
KeBa	185	59	124	2
KeIm	185	57	126	2
DoNe	185	86	99	0
Total	238	143	95	0
%	100	61	39	0

The results show that for the given narrow boxes 143 out of 238 problems (61%) were verified. From the 185 nontrivial problems where simple interval evaluation was not sufficient, our method proved for 86 problems that the box constructed around the approximate solution contains a feasible point. The new method was the only successful one in 29 of the problems (namely bt1, dual1, dual2, dual3, dual4, eigmaxa, eigminc, ex2-1-2, ex3-1-2, ex9-2-8, extrasim, hs012, hs022, hs042, hs044, hs054, hs061, hs083, maratos, meanvar, portfl1, portfl2, portfl3, portfl4, portfl6, try-b, zecevic2, zecevic3, zecevic4). On the other hand, our method failed in 4 cases only where one or two of the other methods succeeded in the verification: dispatch (verified by KeIm), hs028 (verified by KeBa and KeIm), hs35mod (verified by KeBa), and immun (verified by KeBa and KeIm). The detailed test results can be found in the following tables.

Test Result Details			
`n`	number of variables,	`m`	number of constraints,
`eq`	number of equality constraints,	`ieq`	number of ineqality constraints,
`nd`	number of nodes in the DAG,	`qnd`	number of quadratic nodes
`+`	verification succeeded,	`-`	verification failed,
`!`	verification error,	×	no feasible point in the box checked.

Trivial problems (53 problems)

problem	n	m	eq	ieq	nd	qnd
3pk	30	0	0	0	158	32
arglina	100	0	0	0	304	102
arglinb	10	0	0	0	52	12
arglinc	8	0	0	0	46	10
bqp1var	1	0	0	0	4	3
bt13	5	1	1	0	15	7
dixon3dq	10	0	0	0	32	12
ex2-1-1	5	1	0	1	13	8
ex2-1-6	10	5	0	5	27	17
gottfr	2	2	2	0	11	5
h106	8	7	0	7	21	16
h113	10	9	0	9	57	20
h76	4	4	0	4	15	9
h83	5	6	0	6	29	12
h84	5	4	0	4	14	10
h95	6	5	0	5	23	12
harkerp2	100	0	0	0	400	102
hilberta	10	0	0	0	32	12
hilbertb	50	0	0	0	1327	52
hs003	2	0	0	0	6	4
hs021	2	1	0	1	7	5
hs084	5	3	0	3	26	10
hs106	8	7	0	7	21	16
hs116	13	14	0	14	51	28
hs21mod	7	1	0	1	17	10
hs23	2	5	0	5	14	8
hs35	3	2	0	2	11	6

problem	n	m	eq	ieq	nd	qnd
hs3mod	2	0	0	0	6	4
hs44	4	7	0	7	16	12
kear3	4	4	4	0	11	9
matrix2	6	2	0	2	20	10
nasty	2	0	0	0	6	4
nnls	300	1	0	1	2302	302
o32	5	7	0	7	33	13
obstclal	64	0	0	0	766	66
obstclbl	64	0	0	0	766	66
obstclbu	64	0	0	0	766	66
oslbqp	8	0	0	0	18	10
palmer6c	8	0	0	0	36	10
palmer7c	8	0	0	0	36	10
palmer8c	8	0	0	0	34	10
powell	4	4	4	0	15	9
qudlin	12	0	0	0	20	14
rediff3	3	3	3	0	13	7
s324	2	3	0	3	10	6
sim2bqp	2	0	0	0	8	4
simbqp	2	0	0	0	8	4
tf12	3	101	0	101	105	105
tridia	30	1	0	1	92	32
virasoro	8	8	8	0	102	17
vrahatis	9	9	9	0	28	19
zangwil2	2	0	0	0	7	4
zangwil3	3	3	3	0	7	7

Verified problems (143 problems)											
problem	n	m	eq	ieq	nd	qnd	Kraw	HaSe	KeBa	KeIm	DoNe
a	2	2	2	0	8	5	+	+	+	+	+
aircrfta	5	5	5	0	22	11	+	+	+	+	+
b	4	4	4	0	17	9	-	-	+	+	+
b1	2	2	2	0	7	5	+	+	+	+	+
bellido	9	9	9	0	37	19	+	+	+	+	+
booth	2	2	2	0	5	5	+	+	+	+	+
bronstein	3	3	3	0	12	7	+	+	+	+	+
bt1	2	1	1	0	9	5	-	-	-	-	+
chemkin	10	10	10	0	39	21	+	-	+	+	+
clo1	3	3	3	0	10	7	+	+	+	+	+
czaporgeddes	3	3	3	0	24	7	+	+	+	+	+
didrit	9	9	9	0	37	19	+	+	+	+	+
discret3	8	8	8	0	73	17	+	+	+	+	+
dispatch	4	2	1	1	17	8	-	-	-	+	-
dual1	85	1	1	0	3646	88	-	-	-	-	+
dual2	96	1	1	0	4607	99	-	-	-	-	+
dual3	111	1	1	0	6222	114	-	-	-	-	+
dual4	75	1	1	0	2877	78	-	-	-	-	+
eco9	8	8	8	0	53	17	+	+	+	+	+
eiger	4	4	4	0	13	9	+	+	+	+	+
eigmaxa	101	101	101	0	404	204	-	-	-	-	+
eigmaxb	101	101	101	0	404	204	+	+	+	+	+
eigmaxc	22	22	22	0	88	46	+	+	+	+	+
eigminb	101	101	101	0	403	203	+	+	+	+	+
eigminc	22	22	22	0	87	45	×	×	-	-	+
eqlin	3	3	3	0	7	7	-	-	+	+	+
ex2-1-2	6	2	0	2	15	10	-	-	-	-	+
ex3-1-2	5	6	0	6	33	13	-	-	-	-	+
ex9-2-8	3	2	2	0	7	7	-	-	-	-	+
extrasim	2	1	1	0	5	5	-	-	-	-	+
hatfldg	25	25	25	0	98	51	+	+	+	+	+
himmelbc	2	2	2	0	7	5	+	+	+	+	+
hong1	4	4	4	0	17	9	-	-	+	+	+
hong2	3	3	3	0	12	7	+	+	+	+	+
hs006	2	1	1	0	7	5	-	-	+	+	+
hs008	2	2	2	0	7	5	+	+	+	+	+
hs011	2	1	0	1	9	5	-	-	+	+	+
hs012	2	1	0	1	10	5	-	-	-	-	+
hs022	2	2	0	2	11	6	-	-	-	-	+
hs028	3	1	1	0	10	6	-	-	+	+	-
hs035	3	1	0	1	11	6	-	-	+	-	+
hs042	3	1	1	0	14	6	-	-	!	!	+
hs044	4	6	0	6	16	12	-	-	-	-	+
hs054	6	1	1	0	16	9	-	-	!	!	+
hs061	3	2	2	0	12	7	-	-	-	-	+
hs065	3	1	0	1	15	6	-	-	+	-	+
hs083	5	3	0	3	21	10	-	-	-	-	+
hs35mod	2	1	0	1	8	5	-	-	+	-	-
hs8	2	2	2	0	7	5	+	+	+	+	+
hypcir	2	2	2	0	7	5	+	+	+	+	+
immun	19	6	6	0	37	27	-	-	+	+	-
ipp	8	8	8	0	53	17	+	+	+	+	+
kapur	9	9	9	0	27	19	+	+	+	+	+
katsura5	6	6	6	0	37	13	+	+	+	+	+
kear11	8	8	8	0	30	17	+	+	+	+	+
kin2	8	8	8	0	56	17	+	+	+	+	+
kincox	4	4	4	0	17	9	+	+	+	+	+
kinema	9	9	9	0	37	19	+	-	+	+	+
kink	8	8	8	0	30	17	+	+	+	+	+
ku	10	10	10	0	41	21	+	-	+	+	+
ku10	10	10	10	0	31	21	+	-	+	+	+
lorentz	4	4	4	0	17	9	+	+	+	+	+
lsqfit	2	1	0	1	15	5	-	-	+	-	+
maratos	2	1	1	0	9	5	-	-	-	-	+
mathews	3	3	3	0	16	7	+	+	+	+	+
meanvar	7	2	2	0	39	11	-	-	-	-	+
mickey	2	2	2	0	8	5	+	+	+	+	+
monfroy1	4	4	4	0	14	9	+	+	+	+	+
nauheim	8	8	8	0	25	17	+	-	+	+	+
parabola	2	2	2	0	8	5	+	+	+	+	+
portfl1	12	1	1	0	139	15	-	-	-	-	+
portfl2	12	1	1	0	139	15	-	-	-	-	+
portfl3	12	1	1	0	139	15	-	-	-	-	+
portfl4	12	1	1	0	139	15	-	-	-	-	+
portfl6	12	1	1	0	139	15	-	-	-	-	+
precondk	2	2	2	0	7	5	+	+	+	+	+
puma	8	8	8	0	30	17	+	+	+	+	+
redeco5	5	5	5	0	17	11	+	+	+	+	+
redeco6	6	6	6	0	23	13	+	-	+	+	+
redeco7	7	7	7	0	30	15	+	+	+	+	+
redeco8	8	8	8	0	38	17	+	+	+	+	+
s9-1	8	8	8	0	25	17	+	+	+	+	+
simpllpa	2	2	0	2	6	6	-	-	-	+	+
supersim	2	2	2	0	5	5	+	+	+	+	+
tame	2	1	1	0	6	5	-	-	+	+	+
try-b	2	1	1	0	10	5	-	-	-	-	+
wright	5	5	5	0	16	11	+	+	+	+	+
zecevic2	2	2	0	2	7	6	-	-	-	-	+
zecevic3	2	2	0	2	10	6	-	-	-	-	+
zecevic4	2	2	0	2	9	6	-	-	-	-	+

Failed problems (95 problems)						
problem	n	m	eq	ieq	nd	qnd
abel	28	14	14	0	128	44
aircrftb	15	10	10	0	48	27
airport	84	42	0	42	3736	128
aljazzaf	3	1	1	0	15	6
ampl	2	2	1	1	6	5
biggsc4	4	7	0	7	15	13
bqpgabim	199	154	153	1	507	354
braess	4	5	4	1	14	10
braess-n	5	5	4	1	16	11
bt12	5	3	3	0	17	10
bt3	5	3	3	0	18	10
bt8	5	2	2	0	17	9
chandheq	100	100	100	0	10201	201
congigmz	3	5	0	5	13	9
dccircuit	9	10	10	0	20	20
degenlpa	20	14	14	0	36	36
discs	33	66	18	48	464	101
dualc1	9	13	1	12	69	24
dualc2	7	9	1	8	46	18
ex14-1-6	9	15	1	14	51	25
ex2-1-10	20	10	0	10	72	32
ex2-1-3	13	6	0	6	25	21
ex2-1-4	6	4	0	4	13	12
ex2-1-7	20	10	0	10	72	32
ex2-1-8	24	10	10	0	60	36
ex3-1-1	8	6	0	6	21	16
ex3-1-4	3	3	0	3	14	8
ex5-2-2	9	6	4	2	21	17
ex5-2-4	7	6	1	5	25	15
ex5-2-5	32	19	3	16	158	53
ex5-3-3	62	53	53	0	217	117
ex7-3-3	5	8	2	6	18	14
ex8-3-5	110	76	76	0	407	188
ex9-1-1	13	12	12	0	27	27
ex9-1-10	14	12	11	1	28	28
ex9-1-2	10	9	9	0	21	21
ex9-1-4	10	9	9	0	21	21
ex9-1-5	13	12	12	0	27	27
ex9-1-8	14	12	11	1	28	28
ex9-2-1	10	9	9	0	27	21
ex9-2-2	10	9	8	1	25	21
ex9-2-4	8	7	7	0	23	17
ex9-2-5	8	7	7	0	23	17
ex9-2-6	16	12	12	0	34	30
ex9-2-7	10	9	9	0	27	21
fredimage	13	12	12	0	50	26
genhs28	10	8	8	0	38	20
grouping	100	125	125	0	357	227
hanging	288	180	0	180	2058	470
hatfldh	4	7	0	7	15	13
himmel11	9	3	3	0	25	14
himmelbk	24	14	14	0	316	40
hs030	3	1	0	1	11	6
hs043	4	3	0	3	24	9
hs048	5	2	2	0	15	9
hs051	5	3	3	0	18	10
hs052	5	3	3	0	18	10
hs053	5	3	3	0	18	10
hs076	4	3	0	3	15	9
hs097	6	4	0	4	23	12
hs098	6	4	0	4	23	12
hs113	10	8	0	8	57	20
hs118	15	17	0	17	49	34
hs268	5	5	0	5	27	12
hs44new	4	5	0	5	15	11
hs6	2	2	1	1	7	5
ladders	7	13	13	0	47	21
linspanh	72	32	32	0	106	106
lotschd	12	7	7	0	27	21
makela3	21	20	0	20	62	42
makela4	21	40	0	40	62	62
markowitz	8	2	1	1	55	11
mifflin2	3	2	0	2	10	6
minmaxrb	3	4	0	4	10	8
model	60	32	26	6	94	94
optcntrl	28	19	19	0	67	49
optctrl3	118	80	79	1	551	200
optctrl6	118	80	79	1	551	200
polak4	3	3	0	3	14	7
polygon2	38	192	1	191	1005	231
prodpl0	60	29	20	9	107	91
prolog	20	22	0	22	52	44
qp1	50	2	1	1	1329	54
qp2	50	2	1	1	1329	54
qp3	100	52	51	1	204	154
qp4	79	31	30	1	141	112
qp5	108	31	30	1	141	141
res	11	7	7	0	19	19
rosenbr	4	3	2	1	11	8
rosenmmx	5	4	0	4	26	10
simpllpb	2	3	0	3	7	7
sseblin	192	72	48	24	266	266
ssebnln	192	96	72	24	290	290
swopf	82	91	77	14	270	175
tame1	2	2	1	1	6	5

Acknowledgments

This research was supported through the research grant P23554-N13 of the FWF (Fonds zur Förderung der wissenschaftlichen Forschung). We thank Baker Kearfott for answering our questions regarding his verification methods. We also thank the reviewer for the constructive suggestions.

References

1. G. Alefeld and J. Herzberger. *Introduction to Interval Computation*. Academic Press, 1984.
2. F. Domes. GloptLab – a configurable framework for the rigorous global solution of quadratic constraint satisfaction problems. *Optimization Methods and Software*, 24:727–747, 2009.
3. F. Domes, M. Fuchs, and H. Schichl. The Optimization Test Environment. *Optimization and Engineering*, 2013. accepted.
4. F. Domes and A. Neumaier. A scaling algorithm for polynomial constraint satisfaction problems. *Journal of Global Optimization*, 43:327–345, 2008.
5. F. Domes and A. Neumaier. Constraint propagation on quadratic constraints. *Constraints*, 15:404–429, 2010.
6. F. Domes and A. Neumaier. Rigorous filtering using linear relaxations. *Journal Global Optimization*, 53:441–473, 2012.
7. F. Domes and A. Neumaier. Constraint aggregation in global optimization. in preparation, 2013.
8. E. R. Hansen. *Global Optimization Using Interval Analysis*. Marcel Dekker Inc., 1992.
9. R. B. Kearfott. On proving existence of feasible points in equality constrained optimization problems. *Mathematical Programming*, 83(1–3):89–100, 1995.
10. R. B. Kearfott. On verifying feasibility in equality constrained optimization problems. Technical report, 1996.
11. R. Baker Kearfott. Improved and simplified validation of feasible points: Inequality and equality constrained problems. Technical report, 2005.
12. R. Baker Kearfott, Jianwei Dian, and A. Neumaier. Existence verification for singular zeros of nonlinear systems. *SIAM J. Numer. Anal*, 38:360–379, 2000.
13. R.B. Kearfott, M.T. Nakao, A. Neumaier, S.M. Rump, S.P. Shary, and P. van Hentenryck. Standardized notation in interval analysis. In *Proc. XIII Baikal International School-seminar "Optimization methods and their applications"*, volume 4, pages 106–113, Irkutsk: Institute of Energy Systems, Baikal, 2005.
14. R. Krawczyk and A. Neumaier. Interval slopes for rational functions and associated centered forms. *SIAM Journal Numer. Anal.*, 22:604–616, 1985.
15. R. E. Moore. *Interval analysis*. Prentice-Hall, 1966.
16. A. Neumaier. *Interval methods for systems of equations*, volume 37 of *Encyclopedia of Mathematics and its Applications*. Cambridge Univ. Press, Cambridge, 1990.
17. J. Nocedal and S. J. Wright. *Numerical Optimization*, volume 22 of *Series in Operations Research and Financial Engineering*. Springer, 2006.
18. S. M. Rump. *Kleine Fehlerschranken bei Matrixproblemen*. PhD thesis, Universität Karlsruhe, Germany, 1980.

19. S. M. Rump. Validated solution of large linear systems. In *Validation Numerics*. Springer, 1993.
20. S. M. Rump. Verification methods for dense and sparse systems of equations. In *Topics in validated computations*. North Holland, 1994.
21. S. M. Rump and T. Ogita. Super-fast validated solution of linear systems. *Journal Comput. Appl. Math.*, 199(2), 2007.
22. N. V. Sahinidis and M. Tawarmalani. *BARON 7.2.5: global optimization of mixed-integer nonlinear programs,* User's Manual, 2005.
23. O. Shcherbina, A. Neumaier, D. Sam-Haroud, Xuan-Ha Vu, and Tuan-Viet Nguyen. Benchmarking global optimization and constraint satisfaction codes. In Ch. Bliek, Ch. Jermann, and A. Neumaier, editors, *Global Optimization and Constraint Satisfaction*, pages 211–222. Springer, 2003.
24. V. I. Sobolev. Brouwer theorem. In M. Hazewinkel, editor, *Encyclopaedia of Mathematics*. Springer, 2001.