Rigorous global filtering methods with interval unions*

Ferenc Domes and Tiago Montanher and Hermann Schichl and Arnold Neumaier

Dedicated to Vladik Kreinovich on the occasion of his 65th birthday.

Abstract This paper presents rigorous filtering methods for constraint satisfaction problems based on the interval union arithmetic. Interval unions are finite sets of closed and disjoint intervals that generalize the interval arithmetic. They allow a natural representation of the solution set of interval powers, trigonometric functions and the division by intervals containing zero. We show that interval unions are useful when applied to the forward-backward constraint propagation on directed acyclic graphs (DAGs) and can also replace the interval arithmetic in the Newton operator. Empirical observations support the conclusion that interval unions reduce the search domain even when more expensive state-of-the-art methods fail. Interval unions methods tend to produce a large number of boxes at each iteration. We address this problem by taking a suitable gap-filling strategy. Numerical experiments on

Ferenc Domes
Faculty of Mathematics, University of Vienna Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria, e-mail: ferenc.domes@univie.ac.at

Tiago Montanher(✉)
Faculty of Mathematics, University of Vienna Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria, e-mail: tiago.de.morais.montanher@univie.ac.at

Hermann Schichl
Faculty of Mathematics, University of Vienna Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria, e-mail: hermann.schichl@univie.ac.at

Arnold Neumaier
Faculty of Mathematics, University of Vienna Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria, e-mail: arnold.neumaier@univie.ac.at

* This research was supported through the research grants P25648-N25 of the Austrian Science Fund (FWF) and 853930 of the Austrian Research Promotion Agency (FFG)

constraint satisfaction problems from the *COCONUT* show the capabilities of the new approach.

1 Introduction

1.1 Context

Let $F : \mathbb{R}^n \to \mathbb{R}^m$. **Nonlinear systems of equations** can be written as

$$F(x) = 0 \tag{1}$$

and play a central role in several areas of scientific computing and numerical analysis. A point $x^* \in \mathbb{R}^n$ satisfying (1) is called a **root** of the system. The task of finding one or all roots of F under certain conditions is the subject of an extensive literature on numerical analysis.

If one or more parameters of F are not known exactly but belong to some set (typically an interval or the finite union of intervals), then we say that the nonlinear system is **uncertain**. Traditional methods for nonlinear systems are usually not suitable to tackle uncertainty. Kreinovich gives a pedagogical introduction to this subject in [13]. He also considered uncertain problems in a wide range of applications like decision making [12, 14], data fitting [17], indirect measurements [11], outlier detection [16], geophysical tomography [8] among others. We dedicate this paper to his contributions to uncertain problems

Constraint satisfaction problems (CSPs) generalize nonlinear systems of equations and ask for one or all admissible solutions of nonlinear equalities or inequalities. For example,

$$\begin{aligned}
&\text{find} && x && (2)\\
&\text{s.t.} && [1.0, 1.1]x_1 + x_2 = 1 && (3)\\
& && 0 \leq (x_1 - [0.5, 1.5]x_2)^2 \leq 1 && (4)\\
& && x_1 \geq 0 && (5)\\
& && x_2 \in \mathbb{R} && (6)
\end{aligned}$$

is a constraint satisfaction problem. In the CSP framework, relations (3) and (4) are called **constraints** while (5) and (6) are **bound constraints**. The word **find** in this example and throughout the paper denotes the task of finding one solution of the CSP. However, the methods in this paper can also be used to find enclosures for all solutions of a CSP.

The point $x^* \in \mathbb{R}^n$ is called **weakly feasible** if it satisfies all constraints and bound constraints for at least one configuration of the parameters. We say that x^* is **strongly feasible** if the constraints hold for any choice of

parameters. For example, if one fix the unknown parameters in (3) and (4) to 1.0 and $\frac{1}{2}$ respectively, then the column vector $(\frac{1}{2}; \frac{1}{2})$ satisfies all constraints and bound constraints. Therefore it is a weakly feasible point for (2) but not a strong one. The problem is **infeasible** if it has no weak solution.

Neumaier [21] classifies the algorithms to solve CSPs into four groups, according to the degree of rigor. **Incomplete** methods use intuitive heuristics to find approximate feasible points. **Asymptotically complete** procedures reach a solution with probability one if allowed to run indefinitely. **Complete** methods solve the problem with certainty, assuming exact computations. **Rigorous** methods solve CSPs with mathematical certainty and within given tolerances even in the presence of rounding errors.

Interval arithemtic methods are commonly used to solve CSPs from a rigorous perspective. Interval arithmetic is a tool from numerical analysis introduced by Moore in his Ph.D. thesis [19] to automatically evaluate the errors involved in complex calculations. The concept was later extended to prove computational fixed point theorems (see, for example, [20] and the references therein) and found applications in several areas. For a survey of interval arithmetic methods, see [21].

This paper considers only **factorable functions**. The function F is factorable if one can write it as a finite sequence of arithmetic operations and elementary functions. If the function is factorable then it can be represented in a **directed acyclic graph** (DAG) as discussed in [23]. Directed acyclic graphs denote each variable and simple mathematical operation as a node.

Filtering stands for methods to reduce the search domain in constraint satisfaction problems. **Constraint propagation** is a class of filtering that takes the structure of each constraint into account. Two examples of constraint propagation methods applied to factorable functions are the **forward** and **backward** procedures [23]. In the forward mode, we propagate the uncertainty through each node of the DAG, starting from the variables until it reaches each constraint node. The forward procedure is used to obtain enclosures of the range for each constraint and to reduce the uncertainty in the parameters of F. In the backward mode, we propagate the uncertainty reversely, i.e., we walk the graph from the constraints nodes to the variables. The backward mode is used to reduce the search domain.

The **interval Newton operator** is a filtering method extensively studied in the last 40 years. It uses first order information of the function F in a rigorous algorithm that resembles the improvement step of the classical Newton operator. See [20].

1.2 Interval unions and related work

Interval unions are finite sets of closed and disjoint intervals, introduced by [22] and used to enclose all solutions of linear systems under uncertainty in [18]. Interval unions extend the interval arithmetic and provide a natural representation of the solution set of interval power, trigonometric functions, and the division by intervals containing zero. For example, the solution set of $x^2 \in [4,9]$ in the interval space is $[-3,3]$. However, taking the interval union arithmetic into account, we obtain the better enclosure $[-3,-2] \cup [2,3]$.

Multi-intervals are sets of closed intervals that are not necessarily disjoint [26]. They were introduced by YAKOVLEV [27] and TELERMAN (see Telerman et al. [25]). Parallel algorithms for interval and multi-interval arithmetic are the subject of [15]. We review the literature of multi-intervals and their applications in [18].

Another variant of interval unions are the discontinuous intervals by HYVÖNEN [9]. They are disjoint unions of closed, half-open, or open intervals. In our opinion, the extra bookkeeping effort to distinguish between closed and open endpoints is not warranted in most applications.

1.3 Contribution

This paper presents rigorous filtering methods based on the interval union arithmetic. In particular, we discuss the forward-backward constraint propagation and the Newton method using interval unions. The central issue associated with interval unions is the exponential growth in the number of boxes produced after each computation. We introduce a normalized-gap-filling strategy to handle this difficulty.

We integrate the new methods into GLOPTLAB [1, 2], a rigorous solver for constraint satisfaction problems. On the other hand, one can easily implement the algorithms discussed here on any system where an interval library is available. We integrate the new methods with several state-of-the-art filtering procedures such as linear and quadratic contraction [4, 5], feasibility verification [6] and constraint aggregation [7].

Numerical experiments on CSPs from the *COCONUT* test set [24] indicate that interval union methods can reduce the search domain even when more sophisticated approaches fail. The test set consists of 233 small instances, where the number of variables and constraints are not bigger than 9, and 38 cases medium-sized where at least one between the number of variables and constraints belongs to the range $[10,50]$.

The interval union constraint propagation with no-gap-filling is 15% faster than the interval method on small and medium-sized problems on average. The difference rises to 20% if one considers only the last class of instances.

The interval union Newton method with the normalized-gap-filling strategy is 10% faster than the interval one in small instances on average. We found no significant difference between both arithmetics on the Newton operator applied to medium-sized problems.

We conclude from the experiments that the interval union constraint propagation with no-gap-filling is the best option for small and medium-sized problems. If one has access to first-order information of the constraints, then the interval union Newton method with normalized-gap-filling should be the method of choice for low-dimensional instances.

We outline the paper as follows. Section 2 introduces the required basics of interval unions, while Section 3 presents the new enhancements for CSPs. Section 4 gives an overview of GloptLab used in our tests. Numerical experiments are presented in Section 5. We present a supplementary material containing auxiliary Algorithms, and detailed descriptions of the test problems in:
`http://www.mat.univie.ac.at/~montanhe/publications/iucpSup.pdf`

1.4 Notation

This paper employs a Matlab like notation for indices. We write $1:k$ to denote the set of indices $\{1,\ldots,k\}$. The number of elements in an index set N is given by $|N|$.

For vectors and matrices, the relations $=$, $\leq$, $\geq$ and the absolute value $|A|$ of the matrix A are interpreted component-wise. The n-dimension identity matrix is given by I, the transpose of $A \in \mathbb{R}^{n\times m}$ is given by A^T and A^{-T} is a short for $(A^T)^{-1}$.

We assume familiarity with the fundamentals of the interval arithmetic. For a comprehensive approach to this subject, see [20]. The interval notation mostly follows [10].

Let $\underline{a},\overline{a} \in \mathbb{R}$ with $\underline{a} \leq \overline{a}$ then $\mathbf{a} = [\underline{a},\overline{a}]$ denotes an interval with $\inf(\mathbf{a}) := \min(\mathbf{a}) := \underline{a}$ and $\sup(\mathbf{a}) := \max(\mathbf{a}) := \overline{a}$. The set of nonempty compact real intervals is given by

$$\mathbb{IR} := \{[\underline{a},\overline{a}] \mid \underline{a} \leq \overline{a},\ \underline{a},\overline{a} \in \mathbb{R}\}.$$

The extremes of the intervals can assume the ideal points $-\infty$ and ∞. We define $\overline{\mathbb{IR}}$ as the set of closed real intervals. Formally, it can be written as

$$\overline{\mathbb{IR}} := \{[\underline{a},\overline{a}] \cap \mathbb{R} \mid \underline{a} \leq \overline{a},\ \underline{a},\overline{a} \in \mathbb{R} \cup \{-\infty,\infty\}\}.$$

The width of an interval $\mathbf{a}$ is defined by $\mathrm{wid}(\mathbf{a}) := \overline{a} - \underline{a}$. For any set $S \subseteq \mathbb{R}$, the smallest interval containing S is called the interval hull of S and denoted

by $\square S$. The notions of elementary operations between intervals and inclusion properties are the same as presented in [20].

A **box** (or interval vector) $\mathbf{x} = [\underline{x}, \overline{x}]$ is the Cartesian product of the closed real intervals $\mathbf{x}_i := [\underline{x}_i, \overline{x}_i] \in \overline{\mathbb{IR}}$. We denote the set of all interval vectors of dimension n by $\overline{\mathbb{IR}}^n$. We indicate interval matrices by bold capital letters ($\mathbf{A}$, $\mathbf{B}$,...) and the set of all $m \times n$ interval matrices is given by $\overline{\mathbb{IR}}^{m \times n}$.

2 Interval unions

This section reviews the fundamentals of interval unions. A comprehensive description of the arithmetic is the subject of [22].

Definition 1. An interval union $\boldsymbol{u}$ of length $l(\boldsymbol{u}) := k$ is a finite set of k disjoint intervals. We denote the elements of $\boldsymbol{u}$ by $\mathbf{u}_i$ and write

$$\boldsymbol{u} = (\mathbf{u}_1, \ldots, \mathbf{u}_k) \quad \text{with} \quad \begin{array}{ll} \mathbf{u}_i \in \overline{\mathbb{IR}} & \forall\ i = 1:k, \\ \overline{\mathbf{u}}_i < \underline{\mathbf{u}}_{i+1} & \forall\ i = 1:k-1. \end{array} \tag{7}$$

We denote the set of all interval unions of length $\leq k$ by $\mathcal{U}_k$. The set of all interval unions is given by $\mathcal{U} := \bigcup_{k \geq 0} \mathcal{U}_k$ where $\mathcal{U}_0 := \emptyset$ and $\mathcal{U}_1 := \overline{\mathbb{IR}}$.

Definition 2. Let S be a finite set of intervals, the union creator $\mathcal{U}(S)$ is defined as the smallest interval union $\boldsymbol{u}$ that satisfies $\mathbf{a} \subseteq \boldsymbol{u}$ for all $\mathbf{a} \in S$.

It is clear from the definition of union creator that the inclusion isotonic property holds. Formally, $S \subseteq S' \Longrightarrow \mathcal{U}(S) \subseteq \mathcal{U}(S')$.

Definition 3. The set of all interval union vectors of dimension n is given by $\mathcal{U}^n$. In the same way $\mathcal{U}^{n \times m}$ denotes the sets of all interval union matrices of size $n \times m$. The usual operations between matrices and vectors extend naturally to interval unions. We denote interval union matrices by capital bold calligraphic letters like $\boldsymbol{\mathcal{A}}$ or $\boldsymbol{\mathcal{B}}$ and interval union vectors by lower case bold calligraphic letters like $\boldsymbol{x}$ or $\boldsymbol{y}$.

Let $\boldsymbol{u} \in \mathcal{U}_k \setminus \{\emptyset\}$ be an interval union, we denote the interval-wise midpoint of $\boldsymbol{u}$ by $\check{\boldsymbol{u}}_{iw} := (\check{\mathbf{u}}_1, \ldots, \check{\mathbf{u}}_k)$ whenever $-\infty < \underline{\mathbf{u}}_1 \leq \overline{\mathbf{u}}_k < \infty$.

Definition 4. Let $\boldsymbol{u} := (\mathbf{u}_1, \ldots, \mathbf{u}_k)$ and $\boldsymbol{\mathcal{s}} := (\mathbf{s}_1, \ldots \mathbf{s}_k)$ be interval unions of the same length and let $\circ \in \{+, -\}$ then the interval-wise interval union operation corresponding to $\circ$ applied to $\boldsymbol{u}$ and $\boldsymbol{\mathcal{s}}$ is given by

$$\boldsymbol{u} \circ_{iw} \boldsymbol{\mathcal{s}} := \mathbf{u}_1 \circ \mathbf{s}_1 \cup \ldots \cup \mathbf{u}_k \circ \mathbf{s}_k.$$

Definition 5. An interval union function $\mathbf{f} : \mathcal{U}^n \to \mathcal{U}$ is said to be inclusion isotone if $\boldsymbol{u}' \subseteq \boldsymbol{u} \Rightarrow \mathbf{f}(\boldsymbol{u}') \subseteq \mathbf{f}(\boldsymbol{u})$. Moreover, we say $\mathbf{f} : \mathcal{U}^n \to \mathcal{U}$ is the interval union extension of $f : D \subseteq \mathbb{R}^n \Rightarrow \mathbb{R}$ in $\boldsymbol{u} \in \mathcal{U}^n$ if

$$\mathbf{f}(x) = f(x) \quad \text{for } x \in D \cap \boldsymbol{u}, \text{ and } \quad f(x) \in \mathbf{f}(\boldsymbol{u}) \quad \text{for all } x \in D \cap \boldsymbol{u}.$$

3 Interval union and CSPs

This section applies the interval union arithmetic to constraints satisfaction problems. We start with the formal definition of a CSP under the interval union framework. Subsection 3.1 gives one example of the interval union arithmetic in the forward-backward constraint propagation procedure. Subsection 3.2 presents the interval union Newton operator. Subsection 3.3 describes the gap-filling strategy adopted to avoid the exponential growth of intervals in an interval union.

Let $\mathbf{F} : \mathcal{U}^n \to \mathcal{U}^m$ be a factorable function, $\boldsymbol{x} \in \mathcal{U}^n$ and $\boldsymbol{\mathcal{F}} \in \mathcal{U}^m$ then

$$\begin{aligned} &\text{find} && x \\ &\text{s.t.} && \mathbf{F}(x) \in \boldsymbol{\mathcal{F}}, \quad x \in \boldsymbol{x}, \end{aligned} \tag{8}$$

is a constraint satisfaction problem. We also denote constraint satisfaction problems by the triple $(\mathbf{F}, \boldsymbol{\mathcal{F}}, \boldsymbol{x})$.

3.1 The forward-backward constraint propagation

Schichl and Neumaier [23] show that the constraint propagation method in directed acyclic graphs is useful for both, complete and rigorous global optimization. An advantage of the DAG is that it is independent of data types, which means that the same representation can handle intervals or interval unions. Therefore the Algorithms in [23] can be applied to interval unions without any modification. The approach by Schichl and Neumaier consists of performing constraint propagation in the forward and backward modes.

This subsection illustrates how the interval union arithmetic in the forward-backward constraint propagation produces better results than its interval counterpart. Consider the following example

$$\text{find} \quad x \tag{9}$$

$$\text{s.t.} \quad \cos(2\pi x_1) + cos(2\pi x_2) \geq 1, \tag{10}$$

$$x_2 - x_1^2 \leq 0, \tag{11}$$

$$x_1 \in [-2, 2], \quad x_2 \in [-1, 1]. \tag{12}$$

Figure 1 gives a possible DAG for (9). The nodes $\mathbf{x}_1$ and $\mathbf{x}_2$ denote the decision variables with the indicated initial bounds given by (12). Dashed circles denote the constraints (10) and (11), with their respective right hand sides in the interval form. Parameters 2π and -1 are multiplicative constants defined on each constraint. We identify intermediate nodes with labels $\mathbf{T}_i$ and constraints with labels $\mathbf{C}_i$.

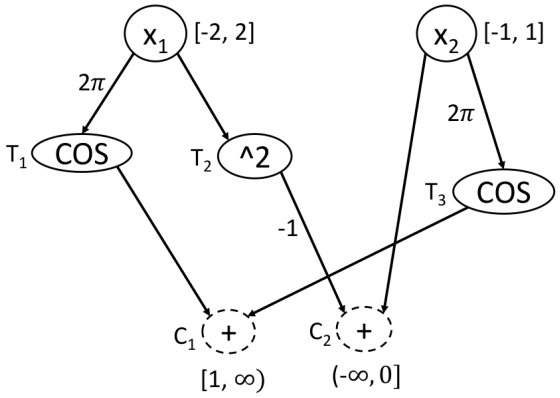

Fig. 1: Directed acyclic graph for the CSP (9)

In the forward mode, the uncertainty flows from the variable nodes to the constraint nodes. In this example, T_1, T_2 and T_3 are given by

$$\mathbf{T}_1 = \cos(2\pi[-2,2]) = [-1,1],$$

$$\mathbf{T}_2 = [0,4] \quad \text{and} \quad \mathbf{T}_3 = [-1,1]$$

and

$$\mathbf{T}_3 = \cos(2\pi[-1,1]) = [-1,1],$$

respectively. To evaluate the range at the constraint nodes, we also take the bounds given by the right hand side of (9) into account to obtain

$$\mathbf{C}_1 = (\mathbf{T}_1 + \mathbf{T}_3) \cap [1,\infty] = [1,2]$$

and

$$\mathbf{C}_2 = (\mathbf{x}_2 - \mathbf{T}_2) \cap [-\infty,0] = [-5,0].$$

Since the parameters are exactly determined, the forward mode does not update them. The backward mode propagates the uncertainty from the constraint nodes to the variable nodes. Let the arc-cosine of an interval union be defined as

$$\arccos(\boldsymbol{a}) := \{x \in \mathbb{R} \mid \cos(x) = a, \quad \forall a \in \boldsymbol{a}\}$$

and define the square root of interval unions in the same way. We denote by T_i^{-1} the reverse operation of the intermediate node T_i. In this case, we have

$$\mathbf{T}_1^{-1} = \arccos((\mathbf{C}_1 - \mathbf{T}_3) \cap \mathbf{T}_1) \cap [-4\pi, 4\pi].$$

The interval $[-4\pi, 4\pi]$ in the expression above is the inflow of the node T_1 in the forward mode. The values of C_1, T_1 and T_3 also come from the forward evaluation. Note that $\mathbf{T}_1^{-1}$ is an interval union since the arc-cosine function produces several gaps. In particular, we have

$$T_1 = \mathcal{U}([-12.57, -10.99], [-7.85, -4.71], [-1.58, 1.58], [4.71, 7.85], [10.99, 12.57]).$$

In the same way, we have

$$\mathbf{T}_2^{-1} = \sqrt{(\mathbf{x}_2 - \mathbf{C}_2) \cap \mathbf{T}_2} \cap [-2, 2]$$

and

$$\mathbf{T}_3^{-1} = \arccos((\mathbf{C}_1 - \mathbf{T}_1) \cap \mathbf{T}_3) \cap [-2\pi, 2\pi].$$

Applying the reverse operation to $\mathbf{x}_1$, we obtain

$$\mathbf{x}_1 = \frac{\mathbf{T}_1^{-1}}{2\pi} \cap \mathbf{x}_1 \quad \text{and} \quad \mathbf{x}_1 = \mathbf{T}_2^{-1} \cap \mathbf{x}_1.$$

Therefore, the search domain for x_1 reduces to the interval union

$$\boldsymbol{x}_1 = \mathcal{U}([-2, -1.75], [-1.25, -0.75], [-0.25, 0.25], [0.75, 1.25], [1.75, 2]).$$

The search domain for x_2 reduces to

$$\boldsymbol{x}_2 = \mathcal{U}([-2, -1.75], [-0.25, 0.25], [1.75, 2]).$$

In the interval arithmetic approach, we lose the gaps produced by the arc-cosine in T_1^{-1} and T_3^{-1}. In this case the search domain is not updated.

3.2 The interval union Newton operator

This subsection presents the interval union Newton operator. We mostly follow and adjust the theory of the interval Newton method given by [20].

Let $F : \mathbb{R}^n \to \mathbb{R}^m$ and $\boldsymbol{x} \in \mathcal{U}^n$. We are interested in finding a rigorous enclosure of the **solution set**

$$\mathcal{S} := \{x \in \boldsymbol{x} \mid F(x) = 0\}.$$

Let $\mathbf{A} \in \mathbb{IR}^{m \times n}$ be a bounded interval matrix and $F : \boldsymbol{x} \subseteq \mathbb{R}^n \to \mathbb{R}^m$ be a function such that

$$F(\tilde{x}) - F(\tilde{y}) = \tilde{A}(\tilde{x} - \tilde{y}) \tag{13}$$

for every $\tilde{x}, \tilde{y} \in \boldsymbol{x}$ and some $\tilde{A} \in \mathbf{A}$. We call $\mathbf{A}$ a **Lipschitz matrix** of F.

In particular, if the function F is continuously differentiable, well defined on every point $x \in \boldsymbol{x}$ and we denote the interval extension of the Jacobian matrix of F by $\mathbf{J}$ then, $\mathbf{A} := \mathbf{J}(\square\boldsymbol{x})$ is a Lipschitz matrix for F.

An **interval union linear system** with coefficients $\boldsymbol{\mathcal{A}} \in \mathcal{U}^{m\times n}$ and $\boldsymbol{\mathcal{b}} \in \mathcal{U}^m$ is the family of linear equations

$$Ax = b \quad (A \in \boldsymbol{\mathcal{A}}, b \in \boldsymbol{\mathcal{b}}). \tag{14}$$

The solution set of (14) is defined by

$$\Sigma(\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{b}}) := \{x \in \mathbb{R}^n \mid Ax = b \text{ for some } A \in \boldsymbol{\mathcal{A}}, b \in \boldsymbol{\mathcal{b}}\}. \tag{15}$$

Let $\boldsymbol{a}, \boldsymbol{\mathcal{b}}, \boldsymbol{x} \in \mathcal{U}$ then the **univariate interval union Gauss-Seidel operator** is given by

$$\Gamma(\boldsymbol{a}, \boldsymbol{\mathcal{b}}, \boldsymbol{x}) := \frac{\boldsymbol{\mathcal{b}}}{\boldsymbol{a}} \cap \boldsymbol{x}. \tag{16}$$

It is clear from the definition that $\Sigma(\boldsymbol{a}, \boldsymbol{\mathcal{b}}) \cap \boldsymbol{x} \subseteq \Gamma(\boldsymbol{a}, \boldsymbol{\mathcal{b}}, \boldsymbol{x})$. The interval union Gauss-Seidel operator can be extended to linear systems with higher dimension assuming the form

$$\boldsymbol{y} := \Gamma(\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{b}}, \boldsymbol{x})$$

where

$$\boldsymbol{y}_i := \Gamma\Big(\boldsymbol{\mathcal{A}}_{ii}, \boldsymbol{\mathcal{b}}_i - \sum_{j\neq i} \boldsymbol{\mathcal{A}}_{ij}\boldsymbol{y}_j, \boldsymbol{x}_i\Big) \text{ for } i = 1:n. \tag{17}$$

Let F and $\mathbf{A}$ be a function and an interval matrix satisfying (13). The interval union **Hansen-Sengupta operator** is given by

$$H(\boldsymbol{x}, \bar{\boldsymbol{x}}) := \bar{\boldsymbol{x}} + \Gamma(C\mathbf{A}, -C\mathbf{F}(\bar{\boldsymbol{x}}), \boldsymbol{x} -_{iw} \bar{\boldsymbol{x}}) \tag{18}$$

where $C \in \mathbb{R}^{n\times m}$ is a preconditioner matrix and $\bar{\boldsymbol{x}}$ is called the **expansion point**. The typical choice for C is the pseudo-inverse of the mid-point of $\mathbf{A}$ ($C = \check{\mathbf{A}}^{-1}$). A better alternative based on the Gauss-Jordan decomposition is presented in [18]. In this paper, we consider the expansion point as the interval-wise midpoint of $\boldsymbol{x}$, i.e., $\bar{\boldsymbol{x}} := \check{\boldsymbol{x}}_{iw}$.

Proposition 1. *Let $F : \boldsymbol{x} \subseteq \mathbb{R}^n \to \mathbb{R}^m$ be Lipschitz continuous on $\boldsymbol{x}$ and let $\mathbf{A} \in \mathbb{IR}^{m\times n}$ be a Liptschitz matrix for F on $\boldsymbol{x}$. Then*

1. $\mathcal{S} \subseteq H(\boldsymbol{x}, \bar{\boldsymbol{x}})$.
2. If $H(\boldsymbol{x}, \bar{\boldsymbol{x}}) \cap \boldsymbol{x} = \emptyset$ then $\mathcal{S}$ is empty.

Proof. Let $x^* \in \mathcal{S}$. By applying (13) with $\tilde{y} = x^*$ we have

$$-F(\tilde{x}) = F(x^*) - F(\tilde{x}) = \tilde{A}(x^* - \tilde{x}) \quad \text{for some} \quad \tilde{A} \in \mathbf{A}.$$

Therefore

$$x^* \in (\tilde{x} + \Sigma(\mathbf{A}, -\mathbf{F}(\tilde{x}))) \cap \boldsymbol{x} = \tilde{x} + (\Sigma(\mathbf{A}, -\mathbf{F}(\tilde{x}))) \cap (\boldsymbol{x} -_{iw} \check{\boldsymbol{x}}_{iw}) \subseteq H(\boldsymbol{x}, \check{\boldsymbol{x}}_c).$$

Hence $x^* \in H(\boldsymbol{x}, \check{\boldsymbol{x}}_c)$ for any $x^* \in \mathcal{S}$ and the result follow. □

Operator (18) requires the solution of a linear system of equations with interval union uncertainties. We can solve it with the interval union Gauss-Seidel procedure [18]. The supplementary material gives a detailed description of the interval union Gauss-Seidel procedure. This paper considers the procedure as a black box algorithm with the input and output given by the Algorithm 1.

Algorithm 1 Interval union Gauss-Seidel: enclose all solutions of an interval union linear system.

Input: The interval union matrix $\boldsymbol{\mathcal{A}}$ and interval union vectors $\boldsymbol{\mathcal{b}}$ and $\boldsymbol{x}$. The absolute and relative tolerances $\epsilon_{Abs} > 0$ and $\epsilon_{Rel} > 0$. The maximum number of iterations K.

Output: The interval union vector $\boldsymbol{y}$ such that $\Sigma(\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{b}}) \cap \boldsymbol{x} \subseteq \boldsymbol{y} \subseteq \boldsymbol{x}$ and a flag indicating one of the following termination status:

1: The problem is infeasible;
2: The absolute or relative gain of $\boldsymbol{y}$ over $\boldsymbol{x}$ do not satisfy the tolerances ϵ_{Abs} or ϵ_{Rel};
3: The absolute and the relative gains of $\boldsymbol{y}$ over $\boldsymbol{x}$ satisgy the tolerance parameters.

The interal union Newton methods is then given by the Algorithm 2.

3.3 Gap filling

The number of boxes produced with the interval union arithmetic may increase exponentially depending on the structure of the constraints. This problem can be solved by applying gap-filling strategies. A gap-filling is a mapping $g : \mathcal{U}_k \to \mathcal{U}_k$ satisfying $\boldsymbol{x} \subseteq g(\boldsymbol{x})$ and $\square \boldsymbol{x} \equiv \square g(\boldsymbol{x})$ for any $\boldsymbol{x} \in \mathcal{U}_k$.

Two trivial gap-filling strategies are the **hull-gap-filling** defined by $g(\boldsymbol{x}) := \square \boldsymbol{x}$ and the **no-gap-filling** where $g(\boldsymbol{x}) := \boldsymbol{x}$. This subsection presents the normalized-gap-filling, a non-trivial gap-filling strategy for interval union scalars and vectors.

Let $\boldsymbol{x} \in \mathcal{U}$ be an interval union and let $\mathbf{x}_i, \mathbf{x}_{i+1} \in \boldsymbol{x}$. The open interval $\mathbf{g}_i$ between the intervals $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ is called the ith **gap** of $\boldsymbol{x}$ and is defined as

$$\mathbf{g}_i = (\overline{\mathbf{x}}_i,\ \underline{\mathbf{x}}_{i+1}). \tag{19}$$

We say that $\mathbf{g}_i \lhd \mathbf{g}_j$ if

$$\mathbf{g}_i \lhd \mathbf{g}_j \Leftrightarrow \Big(\frac{\mathrm{wid}(\mathbf{g}_i)}{\overline{\mathbf{x}}_{i+1} + \underline{\mathbf{x}}_i} < \frac{\mathrm{wid}(\mathbf{g}_j)}{\overline{\mathbf{x}}_{j+1} + \underline{\mathbf{x}}_j} \Big) \vee \Big(\frac{\mathrm{wid}(\mathbf{g}_i)}{\overline{\mathbf{x}}_{i+1} + \underline{\mathbf{x}}_i} = \frac{\mathrm{wid}(\mathbf{g}_j)}{\overline{\mathbf{x}}_{j+1} + \underline{\mathbf{x}}_j} \wedge C(\mathbf{x}_i, \mathbf{x}_j) \Big) \tag{20}$$

where

Algorithm 2 Interval union Newton method: this algorithm applies the interval union Gauss-Seidel procedure to the linearized system until the termination criteria is met

Input: The nonlinear system of equations F, the initial interval union vector $\boldsymbol{x}_0$, the absolute and relative tolerances ϵ_{Abs} and ϵ_{Rel}, the maximum number of iterations K for the Gauss-Seidel procedure and the maximum number of iterations for the Newton method T

Output: The interval union vector $\boldsymbol{y} \subseteq \boldsymbol{x}_0$ such that $\mathcal{S} \subseteq \boldsymbol{y}$.

$\boldsymbol{x} \leftarrow \boldsymbol{x}_0$;
for $t = 1 : T$ **do**
  $\check{\boldsymbol{x}} \leftarrow \check{\boldsymbol{x}}_{iw}$;
  $\mathbf{A} \leftarrow \mathbf{F}(\square \boldsymbol{x})$;
  $C \leftarrow$ Precondition($\mathbf{A}$);
  $\mathbf{A} \leftarrow C\mathbf{A}$;
  $\boldsymbol{\mathcal{b}} \leftarrow -C\mathbf{F}(\boldsymbol{x})$;
  $\boldsymbol{y} \leftarrow$ Gauss-Seidel($\mathbf{A}, \boldsymbol{\mathcal{b}}, \boldsymbol{x} -_{iw} \check{\boldsymbol{x}}, \epsilon_{Abs}, \epsilon_{Rel}, K$);
  if Gauss-Seidel termination status is infeasible **then**
    return $\emptyset$;
  end if
  $\boldsymbol{y} \leftarrow (\check{\boldsymbol{x}} + \boldsymbol{y}) \cap \boldsymbol{x}$;
  if Gauss-Seidel termination status is not enough gain **then**
    return $\boldsymbol{y}$;
  end if
  $\boldsymbol{x} \leftarrow \boldsymbol{y}$;
end for
return $\boldsymbol{x}$;

$$C(\mathbf{x}_1, \mathbf{x}_2) \Leftrightarrow (\langle \mathbf{x}_1 \rangle > \langle \mathbf{x}_2 \rangle \vee (\langle \mathbf{x}_1 \rangle = \langle \mathbf{x}_2 \rangle \wedge \underline{\mathbf{x}_1} < \underline{\mathbf{x}_2})).$$

Intuitively, (20) orders the gaps of the interval union according to its normalized width w.r.t $\square(\mathbf{x}_i, \mathbf{x}_{i+1})$. Algorithm 3 describes the **normalized-gap-filling** strategy.

Algorithm 3 Norm-gap-filling

Input: The interval union vector $\boldsymbol{x}$ with dimension n, the maximum number of gaps in an interval union scalar p and the maximum number of gaps in the interval union vector q.

Output: The vector $\boldsymbol{y}$ such that $\boldsymbol{x} \subseteq \boldsymbol{y}$, $l(\boldsymbol{y}_i) \leq p$, $\prod_{i=1}^{n} l(\boldsymbol{y}_i) \leq q$ and $\square \boldsymbol{x} \equiv \square \boldsymbol{y}$.

if $l(\boldsymbol{x}_i) \leq p$ for $i = 1 : n$ and $\prod_{i=1}^{n} l(\boldsymbol{x}_i) \leq q$ **then**
  return $\boldsymbol{x}$;
end if
$\boldsymbol{y} \leftarrow \boldsymbol{x}$;
while $l(\boldsymbol{y}_i) > p$ for $i = 1 : n$ or $\prod_{i=1}^{n} l(\boldsymbol{y}_i) > q$ **do**
  Find the smallest gap in $\boldsymbol{y}$ according to (20) and call it $\boldsymbol{g}$;
  $\boldsymbol{y} \leftarrow \boldsymbol{y} \cup \boldsymbol{g}$;
end while
return $\boldsymbol{y}$;

4 GloptLab

This section gives a short overview of the rigorous solver GloptLab [1, 2], a configurable framework for global optimization and constraint satisfaction problems. GloptLab implements several state-of-the-art methods for rigorous computations as, for example, linear and quadratic filtering methods [4, 5], feasibility verification [6] and constraint aggregation [7]. We review the basic solver and discuss how the new methods from Section 3 are used to improve its efficiency.

The GloptLab **solver** consists of an **optimizer** carrying out a branch-and-bound process and a **memory** supporting this process. The **optimizer** calls a **preprocessor** properly initializing the memory, then alternates calls to the **reducer**, the **problem selector** and the **splitter**, until a termination criterion is met.

Each bold expression in the last paragraph denotes a configurable module in the system. The methods presented in this paper are useful for the reducer, which employs strategies to reduce the search domain. Algorithm 4 defines a simple, rigorous branch and bound procedure which embeds the reducer.

The inner loop of the Algorithm 4 (lines 14 - 22) describes the reducer. The significant gain (Line 17) in the algorithm depends on the chosen strategy. We consider the feasibility verification methods as black boxes that receive a subproblem $(F, \mathbf{F}, \mathbf{x})$ and return true only if it can prove that $\mathbf{x}$ contains a feasible solution of the problem.

Note that the inner loop restarts whenever a method produces significant contraction of the input box. In practice, we sort the list $\mathcal{M}$ in ascending order according to the computational effort required to run each method. Therefore, cheaper methods are always performed first. The inner loop can also be posed as a finite state machine. Table 1 shows the state machine currently implemented in GloptLab. The interval union forward-backward

Current state	Next state	Condition
Constraint propagation	Constraint propagation Feasibility verification	$G_{Rel}(\boldsymbol{x},\boldsymbol{y}) \geq \epsilon_{CP}$ otherwise
Feasibility verification	Linear contraction	true
Linear contraction	Constraint propagation Quadratic relaxation	$G_{Rel}(\boldsymbol{x},\boldsymbol{y}) \geq \epsilon_{LC}$ otherwise
Quadratic contraction	Constraint propagation exit	$G_{Rel}(\boldsymbol{x},\boldsymbol{y}) \geq \epsilon_{CA}$ otherwise

Table 1: The finite state machine implemented in the inner loop of the Algorithm 4.

constraint propagation described on Section 3 can be used in both the first state of Table 1 and in the step 2 of the Algorithm 4. The interval union

Algorithm 4 Simplified solver

Input: The CSP $(F, \mathbf{F}, \mathbf{x})$ of form (8), the tolerance parameter ϵ_x and the list $\mathcal{M}$ of rigorous methods used to reduce the search domain.

Output: The box $\mathbf{y}$ such that $\text{wid}(\mathbf{y}) < \epsilon_x$ and the certificate that $\mathbf{y}$ contains a feasible point of $(F, \mathbf{F}, \mathbf{x})$ or the certificate that the problem is infeasible.

Run a local solver to obtain the candidate solution $y^* \in \mathbf{x}$;
Run the feasibility verification methods described in [6] to the box $\mathbf{y}$ of width ϵ_x built around y^*;
if $\mathbf{y}$ is verified **then**
    Return $\mathbf{y}$;
end if
Run the forward-backward constraint propagation procedure to $(F, \mathbf{F}, \mathbf{x})$ to reduce the search domain $\mathbf{x}$; Save the reduced domain in $\mathbf{y}$;
if $\mathbf{y}$ is proved to be infeasible **then**
    Return $\emptyset$;
end if
Start the memory with $(F, \mathbf{F}, \mathbf{y})$;
while memory is not empty **do**
    Run the problem selector to obtain the subproblem $(F, \mathbf{F}, \mathbf{x})$;
    $i \leftarrow 1$
    while $i \leq |\mathcal{M}|$ **do**
        Run the strategy $\mathcal{M}_i$ on $(F, \mathbf{F}, \mathbf{x})$ to obtain $(F, \mathbf{F}, \mathbf{y})$;
        $(F, \mathbf{F}, \mathbf{x}) \leftarrow (F, \mathbf{F}, \mathbf{y})$;
        if $\mathbf{y}$ is significantly smaller than $\mathbf{x}$ **then**
            $i \leftarrow 1$;
            continue;
        end if
        $i \leftarrow i + 1$;
    end while
    if $\mathbf{x}$ is verified and $\text{wid}(\mathbf{x}) \leq \epsilon_x$ **then**
        Return $\mathbf{x}$;
    end if
    Run the splitter and stack all subproblems in the memory;
end while
Return $\emptyset$;

Newton operator is a linear contraction method and therefore can be used in the second step of the state machine.

5 Numerical experiments

5.1 The COCONUT test set

This section performs numerical experiments on constraint satisfaction problems from the *COCONUT* test set [24] to evaluate the capabilities of the interval union filtering methods. The *COCONUT* test set contains 306 constraint satisfaction problems. Using the *TestEnvironment* [3], we selected all

instances with the number of variables and constraints in the range $[1, 50]$ to obtain 271 CSPs. We obtained 3 linear problems (i.e., all constraints are linear), 86 quadratics (all constraints are linear or quadratic polynomials), 121 polynomial, 24 rational, 31 smooth, and 6 non-smooth instances. The supplementary material gives a detailed description of the selected problems.

We also selected a subset of medium-sized problems from the set of 271 instances resulting in 38 cases with more than 9 variables and constraints. Again, the supplementary material gives detailed descriptions of the tested problems.

5.2 Forward-backward constraint propagation

We run the Algorithm 4 with the state machine defined by the Table 1 and the linear contraction described by Domes and Neumaier in [5] (also referred simply as **relaxation** in the remainder of this section) to compare the interval union forward-backward constraint propagation with its interval counterpart. We test the normalized-gap-filling strategy as described by the Algorithm 3 with $p = 5$ and $q = 32$. We also consider the no-gap-filling and the hull-gap-filling strategies in our experiments.

We limit the execution time of the Algorithm 4 to 60 seconds for each test problem. All parameters in the Table 1 are set to 0.1. We ran the experiment in a *core i7* processor with frequency of 2.6 GHz, *Windows 10* and *JVM* 1.8.021.

Figures 2 and 3 show that the interval union constraint propagation with no-gap-filling is always better than the hull or normalized strategies. It means that the forward-backward constraint propagation does not generate an excessive number of intervals at each iteration. On average, the no-gap-filling strategy is 15% faster than the hull-gap-filling one in the full set of instances and the difference rises to 20% if we consider only medium-sized problems.

5.3 Interval union Newton method

We ran the Algorithm 4 with the Newton operator as the linear contraction method, under the same conditions as given in the last subsection. For the Algorithm 2, we set $\epsilon_{Abs} = 10^{-4}$, $\epsilon_{Rel} = 0.1$, $K = 10$ and $T = 5$. Figures 4 and 5 show the results of the experiment

Figures 4 and 5 show that the interval union Newton method with no gap-filling strategy can solve fewer problems within the one-minute time limit than their hull and normalized counterparts. This behavior is due to the cost of each function evaluation which increases proportionally with the number of

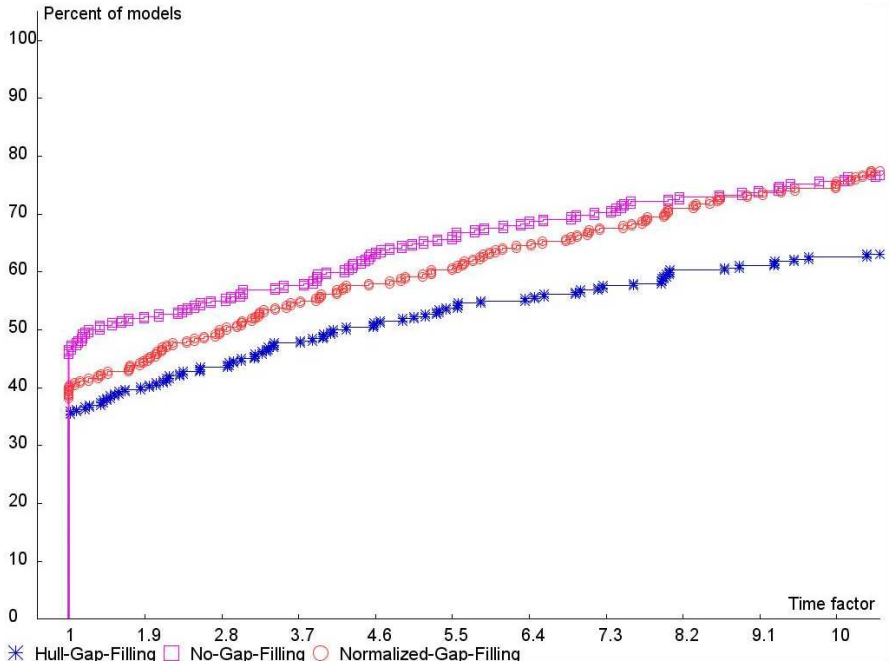

Fig. 2: Time performance profile for the Algorithm 4 with the **linear constraction** described in [5] and three gap-filling strategies for the forward-backward constraint propagation. All 271 instances.

gaps. The normalized-gap-filling presents better results in small problems and is competitive with the hull-gap-filling strategy on medium-sized problems.

The experiment also shows that the interval union Newton method with the normalized-gap-filling strategy is 10% faster than the interval one in the full test set on average. In this case, the number of gaps produced during the linearization is significant, and the simple application of the interval union Newton method (without gap-filling strategies) can be catastrophic. We also note that for the test set of 38 medium-sized problems, the interval Newton operator outperforms the interval union counterpart even with the normalized-gap-filling strategy.

Acknowledgements

The authors would like to thank Dr. Ali Baharev for his valuable comments on the manuscript.

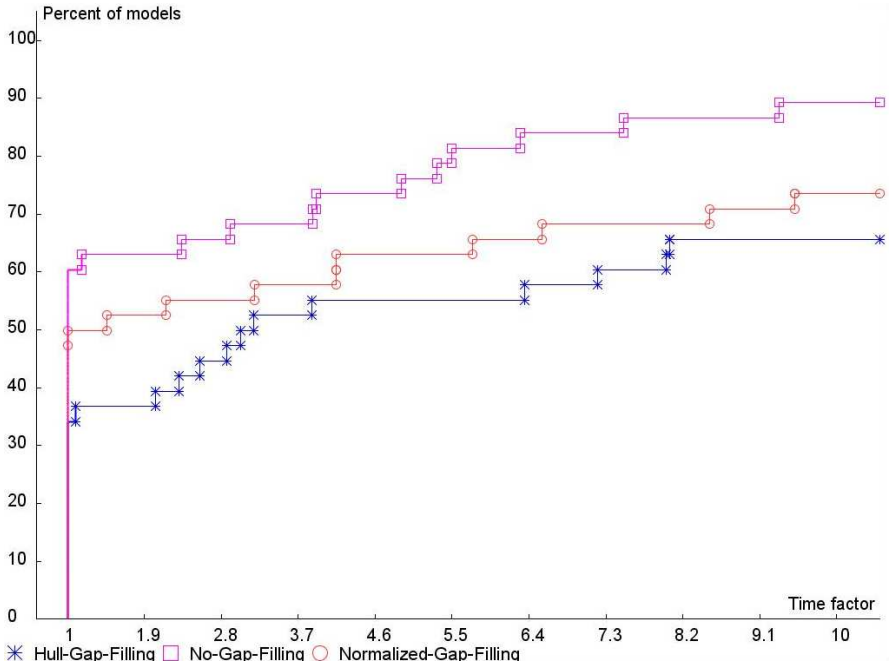

Fig. 3: Time performance profile for the Algorithm 4 with the **linear constraction** described in [5] and three gap-filling strategies for the forward-backward constraint propagation. Medium-sized instances.

References

[1] F. Domes. GloptLab – a configurable framework for the rigorous global solution of quadratic constraint satisfaction problems. *Optimization Methods and Software*, 24:727–747, 2009.

[2] F. Domes. JGloptLab – a rigorous global optimization software. in preparation, 2017.

[3] F. Domes, M. Fuchs, H. Schichl, and A. Neumaier. The Optimization Test Environment. *Optimization and Engineering*, 15:443–468, 2014.

[4] F. Domes and A. Neumaier. Constraint propagation on quadratic constraints. *Constraints*, 15:404–429, 2010.

[5] F. Domes and A. Neumaier. Rigorous filtering using linear relaxations. *Journal of Global Optimization*, 53:441–473, 2012.

[6] F. Domes and A. Neumaier. Rigorous verification of feasibility. *Journal of Global Optimization*, 61:255–278, 2015.

[7] F. Domes and A. Neumaier. Constraint aggregation in global optimization. *Mathematical Programming*, 155:375–401, 2016.

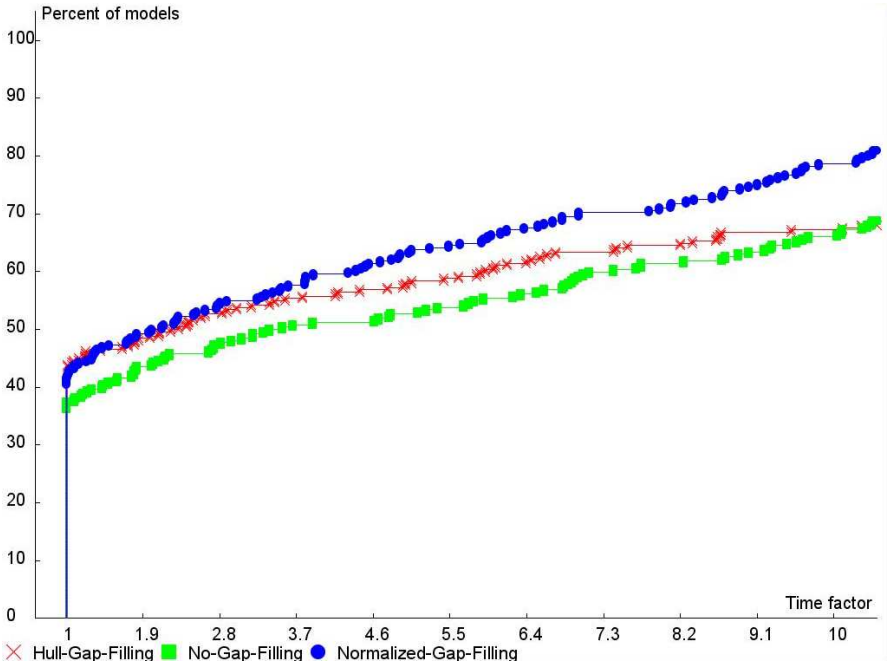

Fig. 4: Time performance profile for the Algorithm 4 with three different gap-filling strategies for the forward-backward constraint propagation and the **interval union Newton** method. All 271 instances.

[8] D. I. Doser, K. D. Crain, M. R. Baker, V. Kreinovich, and M. C. Gerstenberger. Estimating uncertainties for geophysical tomography. *Reliable Computing*, 4(3):241–268, 1998.

[9] E. Hyvönen. Constraint reasoning based on interval arithmetic: the tolerance propagation approach. *Artificial Intelligence*, 58:71–112, 1992.

[10] 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.

[11] V. Kreinovich. Interval computations and interval-related statistical techniques: tools for estimating uncertainty of the results of data processing and indirect measurements. *Data Modeling for Metrology and Testing in Measurement Science*, pages 1–29, 2009.

[12] V. Kreinovich. Decision making under interval uncertainty (and beyond). In *Human-Centric Decision-Making Models for Social Sciences*, pages 163–193. Springer, 2014.

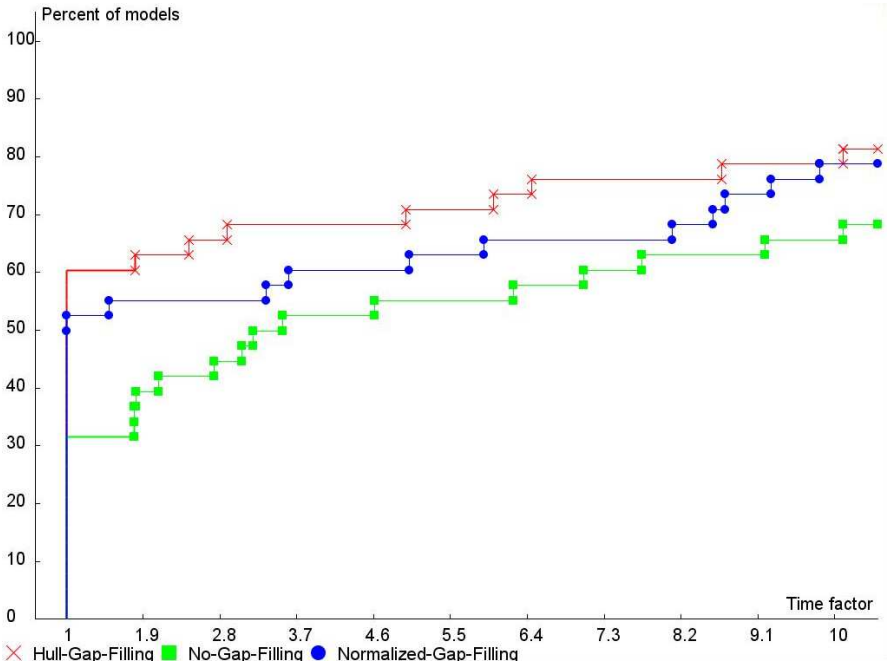

Fig. 5: Time performance profile for the Algorithm 4 with three different gap-filling strategies for the forward-backward constraint propagation and the **interval union Newton** method. Medium-sized problems.

[13] V. Kreinovich. Solving equations (and systems of equations) under uncertainty: how different practical problems lead to different mathematical and computational formulations. *Granular Computing*, 1(3):171–179, Sep 2016.

[14] V. Kreinovich. *Decision making under interval uncertainty (to appear).* Berlin: De Gruyter, 2018.

[15] V. Kreinovich and A. Bernat. Parallel algorithms for interval computations: an introduction. *Interval Computations*, 3:3–6, 1994.

[16] V. Kreinovich, P. Patangay, L. Longpré, S. A. Starks, C. Campos, S. Ferson, and L. Ginzburg. Outlier detection under interval and fuzzy uncertainty: algorithmic solvability and computational complexity. In *Fuzzy Information Processing Society, 2003. NAFIPS 2003. 22nd International Conference of the North American*, pages 401–406. IEEE, 2003.

[17] V. Kreinovich and S. P. Shary. Interval methods for data fitting under uncertainty: A probabilistic treatment. *Reliable Computing*, 23:105–140, 2016.

[18] T. Montanher, F. Domes, H. Schichl, and A. Neumaier. Using interval unions to solve linear systems of equations with uncertainties. *BIT*

Numerical Mathematics, pages 1–26, 2017.

[19] R. E. Moore. *Interval Arithmetic and Automatic Error Analysis in Digital Computing*. PhD thesis, Stanford University, Stanford, CA, USA, 1963.

[20] A. Neumaier. *Interval methods for systems of equations*, volume 37 of *Encyclopedia of Mathematics and its Applications*. Cambridge Univ. Press, Cambridge, 1990.

[21] A. Neumaier. Complete search in continuous global optimization and constraint satisfaction. *Acta Numerica*, 1004:271–369, 2004.

[22] H. Schichl, F. Domes, T. Montanher, and K. Kofler. Interval unions. *BIT Numerical Mathematics*, pages 1–26, 2016.

[23] H. Schichl and A. Neumaier. Interval analysis on directed acyclic graphs for global optimization. *Journal of Global Optimization*, 33(4):541–562, 2005.

[24] O. Shcherbina, A. Neumaier, D. Sam-Haroud, Xuan-Ha Vu, and T. V. 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.

[25] V. Telerman and D. Ushakov. Data types in subdefinite models. In *International Conference on Artificial Intelligence and Symbolic Mathematical Computing*, pages 305–319. Springer, 1996.

[26] I. D. Walker, C. Carreras, R. McDonnell, and G. Grimes. Extension versus bending for continuum robots. *International Journal of Advanced Robotic Systems*, 3(2):171–178, 2006.

[27] A. G. Yakovlev. Computer arithmetic of multiintervals. *Problems of Cybernetics*, pages 66–81, 1987.