# On solving mixed-integer constraint satisfaction problems with unbounded variables

Hermann Schichl*, Arnold Neumaier,
Mihály Csaba Markót, and Ferenc Domes

Faculty of Mathematics, University of Vienna, Austria

**Abstract.** Many mixed-integer constraint satisfaction problems and global optimization problems contain some variables with unbounded domains. Their solution by branch and bound methods to global optimality poses special challenges as the search region is infinitely extended. Many usually strong bounding methods lose their efficiency or fail altogether when infinite domains are involved. Most implemented branch and bound solvers add artificial bounds to make the problem bounded, or require the user to add these. However, if these bounds are too small, they may exclude a solution, while when they are too large, the search in the resulting huge but bounded region may be very inefficient. Moreover, if the global solver must provide a rigorous guarantee (as for the use in computer-assisted proofs), such artificial bounds are not permitted without justification by proof.

We developed methods based on compactification and projective geometry as well as asymptotic analysis to cope with the unboundedness in a rigorous manner. Based on projective geometry we implemented two different versions of the basic idea, namely (i) projective constraint propagation, and (ii) projective transformation of the variables, in the rigorous global solvers COCONUT and GloptLab. Numerical tests demonstrate the capability of the new technique, combined with standard pruning methods, to rigorously solve unbounded global problems. In addition, we present a generalization of projective transformation based on asymptotic analysis.

Compactification and projective transformation, as well as asymptotic analysis, are fruitless in discrete situations but they can very well be applied to compute bounded relaxations, and we will present methods for doing that in an efficient manner.

**Keywords:** Mixed-integer CSPs, constraint propagation, relaxation methods, unbounded variables, interval analysis, directed acyclic graphs

* This work was supported by the Austrian Science Fund (FWF) grant P22239.

## 1 Introduction

### 1.1 Mixed integer Constraint Satisfaction Problems

Many real-world problems lead to mixed-integer and numerical constraint satisfaction problems (MICSPs). Every MICSP is a triplet $(\mathcal{V}, \mathcal{C}, \mathcal{D})$ consisting of a finite set $\mathcal{V}$ of variables taking their values in domains $\mathcal{D}$ over the reals (possibly restricted to the subset of integers) subject to a finite set $\mathcal{C}$ of *numerical* or purely *combinatorial* constraints. A tuple of values assigned to the variables such that all the constraints are satisfied is called a solution. The set of all the solutions is called the solution set. When dealing with a MICSP, depending on the application, it might suffice to find *one* solution, but in some cases it might be necessary to identify the whole solution set.

In practical problems, numerical constraints are often expressed as equations and inequalities in *factorable* form, that is, they are described by functions that are recursively composed of elementary functions such as arithmetic operators (+, −, ∗, /), and univariate (sometimes bivariate) basic functions like log, exp, sin, cos,... In other words, such an MICSP can be expressed as

$$F(x) \in \mathbf{b},\ x \in \mathbf{x},\ x_I \in \mathbb{Z}^{|I|}, \tag{1}$$

where $F : \mathbb{R}^n \to \mathbb{R}^m$ is a factorable function, $x$ is a vector of $n$ real variables, $\mathbf{x}$ and $\mathbf{b}$ are interval vectors of sizes $n$ and $m$ respectively, and $I$ is the set of integer variables.

Many solution techniques have been proposed in *Constraint Programming* and *Mathematical Programming* to solve MICSPs. A difficulty when dealing with continuous variables is roundoff errors. For achieving full rigor, almost all solution techniques for MICSPs use *interval arithmetic* (see [12, 16–18]) or some of its variants (affine arithmetic [30], Taylor arithmetic [4, 5, 19], etc.). During the last two decades, a lot of work has been put into the development of *inclusion tests* and *contractors* based on interval arithmetic. In addition, numerous *relaxation techniques* (many of them based on interval arithmetic combined with algorithmic differentiation methods [11, 25]) have been devised (see [13, 20]).

The function of an inclusion test is to check whether the domain of a variable is included in the projection of the solution set. A contractor, also called a *narrowing operator* [2, 10] or *contracting operator* [1, 29, 32], is a method that computes a (hopefully proper) subset of the variable domains such that all solutions are retained. Various basic inclusion tests and contractors have been described in [13] and [20].

In particular, a contraction operator approach called *interval constraint propagation* was developed [2, 3, 31], which associates *constraint propagation/local consistency* techniques, as defined in artificial intelligence, with interval analytic methods. Advanced contractors, such as the *forward-backward contractor* [2, 13], result from the interval constraint propagation (CP) approach. It is a way to propagate domain reductions forwards and backwards through the computational trees of the constraints. Based on the fundamental framework for interval analysis on directed acyclic graphs (DAGs) [27], a high performance constraint propagator `FBPD` for continuous CSPs has been developed in [34].

In practical constraint solvers inclusion tests and contractors are interleaved with some form of *exhaustive search* to compute a representation of the solution set. Search by *bisection* or more advanced branching is the most commonly used technique. In the context of MICSPs this leads to the branch and bound class of algorithms, which generate a search graph consisting of subproblems that are subsequently solved or further subdivided.

A *relaxation* is a (usually much easier solvable) replacement MICSP whose solution set provably contains all solutions of problem (1). There are several classes of relaxations, although linear and convex ones are mostly used, see [20, 21]. Relaxations usually are an efficient tool for fathoming nodes of the search graph during the search procedure.

### 1.2 Unbounded Variables

An especially difficult class of MICSPs are those which contain variables whose domain set is unbounded. Their solution by branch and bound methods poses special challenges as the search region is infinitely extended. On the one hand, the unboundedness cannot be removed by splitting. On the other hand most inclusion and contraction operators become inefficient or dysfunctional when applied to unbounded domains.

Most branch and bound solvers add artificial bounds to make the problem bounded, or require the user to add these by forbidding unbounded problems altogether. However, if these artificial bounds are too small, they may exclude a solution, even render the problem infeasible, while when they are too large, the search in the resulting huge but bounded region may be very inefficient. Moreover, if the global solver must provide a rigorous guarantee (as for the use in computer-assisted proofs), such artificial bounds are not permitted without justification.

The contribution of this paper is twofold. Firstly, we developed methods based on compactification and projective geometry to cope with the unboundedness in a rigorous manner. We implemented two different versions of the basic idea, namely

1. projective transformation of the variables, and
2. projective constraint propagation.

They are implemented in the rigorous global solvers GloptLab [6–8] and COCONUT [23, 24], respectively. Numerical tests demonstrate the capability of the new technique, combined with standard pruning methods, to rigorously solve unbounded global problems.

Secondly, these projective transformations are most efficient for those MICSPs, whose unbounded variables are continuous and all constraints involving them are rational. Although the method is still applicable when transcendental functions are involved, the effectiveness is significantly reduced. Therefore, we developed an extension using asymptotic analysis that is more efficient in the presence of transcendental functions. This is based on ideas from the unpublished thesis [9]. Since for discrete variables the transformation method is not applicable, we shortly describe asymptotic relaxations for improved node fathoming in the unbounded case.

In Section 2 we will explain explicit projective transformation and projective constraint propagation. Section 3 generalizes that to asymptotic transformations. Some information on projective and asymptotic relaxations are given in Section 4, and numerical results are provided in Section 5.

Throughout this paper we will need some notation: a real interval $\mathbf{a} \in \overline{\mathbb{IR}}$ is defined as $[\underline{a}, \overline{a}] = \{a \in \mathbb{R} \mid \underline{a} \le a \le \overline{a}\}$, with $\underline{a} \in \mathbb{R} \cup \{-\infty\}$ and $\overline{a} \in \mathbb{R} \cup \{\infty\}$. In case both bounds $\underline{a}$ and $\overline{a}$ of $\mathbf{a}$ are finite, we call $\mathbf{a}$ a finite or bounded interval, otherwise $\mathbf{a}$ is an infinite or unbounded interval. We will also need the set $\overline{\mathbb{UR}}$ of all finite disjoint unions of intervals. Real arithmetic and elementary functions can be extended to intervals, see [18], and to interval unions. An $n$-dimensional real box (union box) $\mathbf{x} \in \overline{\mathbb{IR}}^n (\overline{\mathbb{UR}}^n)$ is a vector of $n$ real intervals (interval unions). If all components of $\mathbf{x}$ are finite, then $\mathbf{x}$ is a finite or bounded box or interval union, otherwise $\mathbf{x}$ is infinite or unbounded.

## 2 Projective Transformation

Throughout this section we will consider factorable MICSPs of the form (1). We will assume that the variables $x_J$ have unbounded domains and that the other variables $x_K$ have bounded domains. Furthermore, we will for the moment require that all integer variables are bounded, i.e., that $I \subset K$.

Since all functions involved in (1) are factorable, the problem can be represented as a reduced computational directed acyclic graph $\Gamma = (V(\Gamma), E(\Gamma))$, see [27]. All nodes $\nu \in V(\Gamma)$ represent intermediate expressions $y_\nu$ of some constraints. The local sources of $\Gamma$ correspond to constants and variables, i.e., $x_k = y_{\nu_k}$ for all $k$ and some $\nu_k \in V(\Gamma)$, and the local sinks correspond to the constraints.

The basic idea of the projective transformation is the natural embedding of $\mathbb{R}^{|J|} \times \mathbf{x}_K$, which contains the feasible set, into the compact manifold with boundary $P\mathbb{R}^{|J|} \times \mathbf{x}_K$, where $P\mathbb{R}^{|J|}$ is the projective space over $\mathbb{R}^{|J|}$. For the transformation we represent each intermediate node $y_\nu$ for $\nu \in V(\Gamma)$ in the form

$$y_\nu = \widehat{y}_\nu / t^{m_\nu}, \tag{2}$$

where $m_\nu$ is a rational number and $t$ is a scaling factor to be chosen. The new variable $t$ and the exponents $m_\nu$ are defined such that $t \in [0,1]$ and the $\widehat{y}_\nu$ are well-bounded. (Actually, that can only be guaranteed in the case of a rational MICSP. In the presence of transcendental functions some intermediate $\widehat{y}_\nu$ may still be unbounded. This is the motivation for the generalization in Section 3.) Note that while these transformations are singular, the transformed problem has no singularities, and the solution set is preserved with full mathematical rigor.

The transformation is achieved by a recursive construction, implemented in a forward walk through $\Gamma$. For the original variables $x_k$ and all interval constants, we define

$$m_{\nu_k} = \begin{cases} 0 & \text{if } k \in K, \\ 1 & \text{if } k \in J. \end{cases} \tag{3}$$

For practical reasons we put in the implementation also those indices $j$ into $J$, for which the bounds $\mathbf{x}_j$ are huge, e.g., bigger than $10^7$ but this limit is problem and scaling dependent. Those bounds, in general, are artificial in the first place and in a branch and bound context pose similar problems as unbounded variables.

For constructing the $\widehat{y}_\nu$ we choose a real number $0 \le s \le 1$ and set

$$t := \Big(1 - s + \sum_{k\in K} d_k x_k^2\Big)^{-1/2} \tag{4}$$

with scaling factors $d_k > 0$. This leads to the constraint

$$(1-s)t^2 + \sum d_k \widehat{x}_k^2 = 1, \tag{5}$$

from which we deduce the bounds

$$\begin{aligned} t \in \mathbf{t} &:= [0, \max(0, 1-s)^{-1/2}], \\ |\widehat{x}_k| &\le d_k^{-1/2} \quad \text{for } k \in K. \end{aligned} \tag{6}$$

To guarantee that $t$ is real, we need to choose $s$ such that

$$\sum_{k\in K} d_k x_k^2 \ge s$$

is a valid constraint. For example,

$$s := \inf \sum_{k\in K} d_k \mathbf{x}_k^2$$

qualifies (if necessary, rescale the $d_k$ to have $s \le 1$), but better bounds might be available. A possible choice is $s = 0$, however in general this is suboptimal. Then

$$t \in [0,1].$$

Since $\widehat{y}_{\nu_k} = x_k$ for the well-bounded variables, we have expressed all variables in terms of bounded ones.

The exponent $m_\nu$ for an intermediate variable $y_\nu$ depends on the operation that creates it. If $y = \sum \alpha_\nu y_\nu$ then $y = \widehat{y}/t^m$ with

$$m := \max m_\nu, \quad \widehat{y} := \sum \alpha_\nu t^{m-m_\nu} \widehat{y}_\nu, \tag{7}$$

and we get the finite enclosure

$$\widehat{y} \in \widehat{\mathbf{y}} := \sum \alpha_\nu \mathbf{t}^{m-m_\nu} \widehat{\mathbf{y}}_\nu. \tag{8}$$

If $y = \prod y_\nu^{\alpha_\nu}$ with rational $\alpha_\nu$ then $y = \widehat{y}/t^m$ with

$$m := \sum \alpha_\nu m_\nu, \quad \widehat{y} := \prod \widehat{y}_\nu^{\alpha_\nu}, \tag{9}$$

and we get the finite enclosure

$$\widehat{y} \in \widehat{\mathbf{y}} := \prod \widehat{\mathbf{y}}_\nu^{\alpha_\nu}. \tag{10}$$

This accounts for all elementary operations and powers with fixed exponent.

For other elementary functions, one can derive similar formulas, though their derivation and implementation is more complex. For example, if $y = \log y_\nu$ then $y = \widehat{y}/t^m$ with

$$m := 1, \quad \widehat{y} := t(\log \widehat{y}_\nu - m_\nu \log t),$$

and we get a finite enclosure derivable by monotony considerations.

For some transcendental functions $y = \varphi(y_\nu)$ even a projective transformation cannot in general guarantee boundedness of $\widehat{y}$ (one example is $\varphi = \exp$). In that case, we define $\widehat{y} := \varphi(t^{-m_\nu}\widehat{y}_\nu)$ and get a possibly unbounded enclosure of $\varphi(\mathbf{t}^{-m_\nu}\mathbf{y}_\nu)$ for $\widehat{y}$.

There are two possibilities to utilize projective transformations. The problem can be explicitly transformed (see Section 2.1), or the projective transformation can be used implicitly during constraint propagation (see Section 2.2) and relaxation calculation (see Section 4).

## 2.1 Explicit projective transformation

We have implemented the explicit transformation method in the software package GloptLab [6–8], a constraint satisfaction package for enclosing all solutions of systems of quadratic equations and inequalities.

The special quadratic structure allows one to implement projective transformations explicitly by rewriting the original equations after a projective transformation (2) on the variables $x_k$ using (4). Then all linear inequality constraints

$$Ax \geq b$$

are transformed into the homogeneous linear constraints

$$A\widehat{x} - bt \geq 0.$$

Bound constraints are treated as linear constraints, too. Nonlinear quadratic constraints

$$x^T G x + c^T x \geq \gamma$$

are transformed into the homogeneous quadratic constraints

$$\widehat{x}^T G \widehat{x} + t c^T \widehat{x} - \gamma t^2 \geq 0.$$

In all cases, equations and inequalities with the opposite sign are handled analogously. The additional constraints (5) and (6) are also quadratic and linear, respectively, and thus the transformed problem is again quadratic but bounded. Hence, it can be solved with traditional methods. After solving the transformed problem, one can recover the original solution from

$$x_i = \widehat{x}_i / t, \quad \text{if } t \neq 0.$$

Solutions of the transformed problem with $t = 0$ correspond to limiting solutions at infinity of the original problem. They can be discarded in general.

Alternatively, one can solve a bigger constraint satisfaction problem containing both the original and the transformed variables and constraints. In that case, however, the transformation equations themselves have to be added as additional quadratic constraints

$$x_i t - \widehat{x}_i = 0.$$

This allows one to exploit the features of both the original and the transformed problem at the same time, at the cost of doubling the problem size.

*Example 1.* The constraint satisfaction problem

$$\begin{aligned} 0.36x_1 - x_2 &= 0.75, \\ 2x_1^2 - x_2^2 &= 1, \\ x_1 \geq 0, \quad x_2 &\geq 0 \end{aligned}$$

is infeasible but the equations have a solution at

$$x_1 \approx 0.6491, \; x_2 \approx -0.5163,$$

slightly outside the defining box, see Fig. 1.

The problem is difficult to solve with standard CP and branch and bound, since no box $\mathbf{x}$ of the form $\mathbf{x}_1 = [a, \infty]$, $\mathbf{x}_2 = [b, \infty]$ can be reduced by CP.

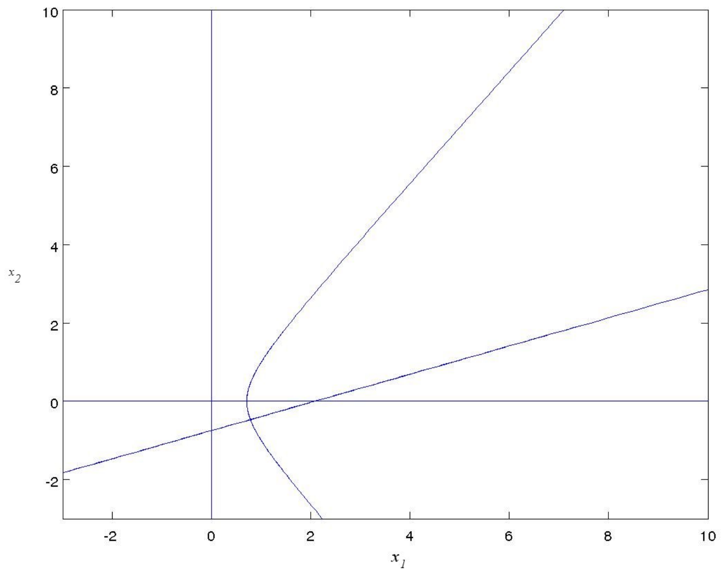

**Fig. 1.** Example 1

The projective transformation leads to the problem

$$\begin{aligned} 0.36\widehat{x}_1 - \widehat{x}_2 - 0.75t &= 0, \\ 2\widehat{x}_1^2 - \widehat{x}_2^2 - t^2 &= 0, \\ \widehat{x}_1^2 + \widehat{x}_2^2 + t^2 &= 1, \\ \widehat{x}_1, \widehat{x}_2, t &\in [0,1] \end{aligned}$$

This problem is easily found to be infeasible by CP.

## 2.2 Projective constraint propagation

For more general problems explicit transformation becomes more cumbersome. For the implementation in the COCONUT Environment [23, 24], a software platform for global optimization, we have therefore chosen a different approach. Frequently, in the non-quadratic case a transformed MICSP is not easier to solve than the original problem. Therefore, we utilize the projective transformation together with a special split into a bounded subproblem and its complement.

For that observe that the transformation (2) on the variables $x_k$ has the following property. If $\|\widehat{x}\|_p = \alpha$ and $t \in [0,1]$ then $\|x\|_p \geq \alpha$, for every $p \in [1,\infty]$. Adding the constraint $\|x\|_p \leq \alpha$ to problem (1) makes it bounded, so it can be solved by standard methods.

The complement of that ball, described by the complementary constraint $\|x\|_p \geq \alpha$, must be handled as well and can be projectively transformed using (2) and the constraint $\|\widehat{x}\|_p = \alpha$. The choice of the constant $\alpha$ is application specific.

However, this transformation is never performed explicitly. Rather, many of the bounding tools, and foremost CP, implicitly make use of the transformation by calculating in $\mathbb{I}P\mathbb{R}$, the set of so-called *projective intervals*. Those are pairs $(\widehat{\mathbf{x}}, r; \mathbf{t})$ of intervals and rational numbers together with a common interval $\mathbf{t}$ representing the range of the scaling parameter. The operations of projective intervals are defined according to (7–10) with extensions for transcendental elementary functions.

Not performing the transformation explicitly has an additional advantage which is connected to the following important observation: There exist CSPs for which CP proceeds with range reduction on the original problem but where it has no reducing effect on the projectively transformed problem.

*Example 2.* Consider the constraints (with $\alpha = 1$)

$$y = x^2 - 3 \tag{11}$$

$$x^2 + y^2 \geq 1. \tag{12}$$

From (11) we get $y \in [-\infty, \infty] \cap ([-\infty, \infty]^2 - 3) = [-3, \infty]$, which reduces the range of $y$.

The projectively transformed problem associated to (11–12) is as follows:

$$\hat{y}t = \hat{x}^2 - 3t \tag{13}$$

$$\hat{x}^2 + \hat{y}^2 = 1 \tag{14}$$

$$\hat{x} \in [-1, 1],\ \hat{y} \in [-1, 0],\ t \in (0, 1]. \tag{15}$$

From (13) we get

$$y \in [-1, 0] \cap ([-1, 1]^2 - 3 \cdot (0, 1]) = [-1, 0].$$

Also from (13),

$$x^2 = \hat{y}t + 3t \in [-1, 0] + (0, 3] = (-1, 3],$$

so the current range $[-1, 1]$ of $\hat{x}$ cannot be reduced; (14) yields no improvement as well, thus, CP cannot reduce any of the initial variable ranges of the transformed problem.

The effect that no range reduction is possible on the original problem whereas CP works on the transformed problem was already demonstrated in Example 1.

Consequently, we need to utilize CP simultaneously on the original and on the transformed problem. For that we developed the algorithm *Projective Forward and Backward Propagation on DAGs* (`PFBPD`), which is an advanced version of `FBPD` from [33, 34]. It is based on propagating enclosures of the form

$(\mathbf{x}, (\widehat{\mathbf{x}}, r; \mathbf{t})) \in \overline{\mathbb{UR}}^n \times \mathbb{I}P\mathbb{R}^n$, of pairs of interval unions (not projectively transformed) and projective intervals in parallel, in order to get the advantages of both approaches. These pairs are interwoven since after each forward or backward propagation step an internal intersection between the two enclosures is performed for additional reduction.

For this algorithm let $D(\mathbf{G})$ be a DAG with the ground $\mathbf{G}$, $\mathcal{C}$ the set of active constraints, and $\mathcal{D}$ the variable domains. Furthermore, for every node $\nu$ of the DAG we introduce the set $\mathcal{N}(\nu) \subseteq \overline{\mathbb{UR}} \times \mathbb{I}P\mathbb{R}$, or $\mathcal{N}(\nu) \subseteq \mathbb{UZ} \times \mathbb{I}P\mathbb{R}$ for the integer variables, containing the current enclosure of the range of $\nu$. Note that we keep the integer information only in the untransformed problem since the transformation destroys the integrality information.

```
Algorithm PFBPD(in : $D(\mathbf{G}), \mathcal{C}, \alpha$; in/out : $\mathcal{D}$)
00:   $\mathcal{L}_f := \emptyset$; $\mathcal{L}_b := \emptyset$; $V_{oc} := (0,\dots,0)$; $V_{ch} := (0,\dots,0)$; $\mathbf{t}_o := [0,1]$;
01:   Set the node ranges $\mathcal{N}_{\nu_k}$ of every variable $x_k$ to $(\mathcal{D}_k, (\widehat{x}_k, m_k; \mathbf{t}_o))$;
02:   $V_{lvl} := (0,\dots,0)$;
03:   for each node $\mathbf{C}$ representing an active constraint in $\mathcal{C}$ do
04:       NodeOccurrences($\mathbf{C}, V_{oc}$);
05:       NodeLevel($\mathbf{C}, V_{lvl}$); /* this can be made optional */
06:   end-for
07:   Add a virtual node $\mathbf{V}$ with maximal node level (for constraint $\|\hat{x}\|_2^2 = \alpha^2$).
08:   $\mathcal{C} := \mathcal{C} \cup \{\mathbf{V}\}$;
09:   for each node $\mathbf{C}$ representing an active constraint in $\mathcal{C} \setminus \{\mathbf{V}\}$ do
10:       FindVirtualEdges($\mathbf{C}, \mathbf{V}, \mathcal{V}$);
11:       ForwardEvaluation($\mathbf{C}, V_{ch}, \mathcal{L}_b$);
12:   end-for
13:   while $\mathcal{L}_b \neq \emptyset \vee \mathcal{L}_f \neq \emptyset$ do
14:       $\mathbf{N}$ := getNextNode($\mathcal{L}_b, \mathcal{L}_f$);
15:       if $\mathbf{N}$ was taken from $\mathcal{L}_b$ then
16:           for each child $\mathbf{C}$ of $\mathbf{N}$ do
17:               BP($\mathbf{N}, \mathbf{C}$);
18:               if $\mathcal{N}(\mathbf{C}) = \emptyset$ then return infeasible;
19:               if $\mathcal{N}(\mathbf{C})$ changed enough for forward evaluation then
20:                   for each $\mathbf{P} \in parents(\mathbf{C}) \setminus \{\mathbf{N}, \mathbf{G}\}$ do
21:                       if $V_{oc}[\mathbf{P}] > 0$ then put $\mathbf{P}$ into $\mathcal{L}_f$;
22:               end-if
23:               if $\mathcal{N}(\mathbf{C})$ changed enough for backward propagation then
24:                   Put $\mathbf{C}$ into $\mathcal{L}_b$;
25:           end-for
26:       else /* $\mathbf{N}$ was taken from $\mathcal{L}_f$ */
27:           FE($\mathbf{N}, [f]$); /* $f$ is the operator at $\mathbf{N}$ */
28:           if $\mathcal{N}(\mathbf{N}) = \emptyset$ then return infeasible;
29:           if $\mathcal{N}(\mathbf{N})$ changed enough for forward evaluation then
30:               for each $\mathbf{P} \in parents(\mathbf{N}) \setminus \{\mathbf{G}\}$ do
31:                   if $V_{oc}[\mathbf{P}] > 0$ then put $\mathbf{P}$ into $\mathcal{L}_f$;
```

```
32:            end-if
33:            if $\mathcal{N}(\mathbf{N})$ changed enough for backward propagation then
34:                Put $\mathbf{N}$ into $\mathcal{L}_b$;
35:        end-if
36:        if $\mathbf{t} \neq \mathbf{t}_o$ then
37:            if $\mathbf{t}$ changed enough for forward propagation then
38:                Put all nodes $C \in \mathcal{V}$ into $\mathcal{L}_f$;
39:            end-if
40:            $\mathbf{t}_o := \mathbf{t}$;
41:        end-if
42:    end-while
43:    Update $\mathcal{D}$ with the ranges of the nodes representing the variables;
end
```

```
procedure ForwardEvaluation(in : $\mathbf{N}$;  in/out : $V_{ch}, \mathcal{L}_b$)
01:    if $\mathbf{N}$ is a leaf or $V_{ch}[\mathbf{N}] = 1$ then return;
02:    for each child $\mathbf{C}$ of node $\mathbf{N}$ do ForwardEvaluation($\mathbf{C}, V_{ch}, \mathcal{L}_b$);
03:    if $\mathbf{N} = \mathbf{G}$ then return;
04:    FE($\mathbf{N}, [f]$); /* $f$ is the operator at $\mathbf{N}$ */
05:    $V_{ch}[\mathbf{N}] := 1$; /* the range of this node is cached */
06:    if $\mathcal{N}(\mathbf{N}) = \emptyset$ then return infeasible;
07:    if $\mathcal{N}(\mathbf{N})$ changed enough for backward propagation then put $\mathbf{C}$ into $\mathcal{L}_b$;
end
```

```
procedure FindVirtualEdges(in : $\mathbf{N}, \mathbf{V}$;  in/out : $\mathcal{V}$)
01:    if $\mathbf{N}$ is a leaf and $m_N \neq 0$ then put $\mathbf{N}$ into the set of children of $\mathbf{V}$;
02:    if $\mathbf{t}$ is explicitly needed in the calculation of $\mathbf{y}_N$ then add $\mathbf{N}$ to $\mathcal{V}$;
03:    for each child $\mathbf{C}$ of node $\mathbf{N}$ do FindVirtualEdges($\mathbf{C}, \mathbf{V}, \mathcal{V}$);
end
```

```
procedure NodeLevel(in : $\mathbf{N}$;  out : $V_{lvl}$)
01:    for each child $\mathbf{C}$ of node $\mathbf{N}$ do
02:        $V_{lvl}[\mathbf{C}] := \max\{V_{lvl}[\mathbf{C}], V_{lvl}[\mathbf{N}] + 1\}$;
03:        NodeLevel($\mathbf{C}, V_{lvl}$);
04:    end-for
end
```

Apart from the virtual nodes and constraints the layout of the `PFBPD` algorithm is analogous to the `FBPD` algorithm from [33, 34]. The main difference lies in the forward and backward propagation operators $\texttt{FE}(\mathbf{N}, [f])$ and $\texttt{BP}(\mathbf{N}, \mathbf{C})$. The aim of *forward evaluation* `FE` is the reduction of $\mathcal{N}(\mathbf{N})$ of the node $\mathbf{N}$ based on the known $\mathcal{N}(\mathbf{C})$ for all children $\mathbf{C}$ of $\mathbf{N}$. It is performed by first calculating the $\overline{\mathbb{UR}}$ and the $\mathbb{I}P\mathbb{R}$ parts of $\mathcal{N}(\mathbf{N}) = (\mathbf{x}, (\widehat{\mathbf{x}}, m; \mathbf{t}))$ separately by interval extension functions of $f$. Immediately thereafter the internal intersection

$\mathcal{N}'(\mathbf{N}) = (\mathbf{x}', (\widehat{\mathbf{x}}', m; \mathbf{t}'))$ is computed as follows:

$$\begin{aligned}
\mathbf{x}' &= \mathbf{x} \cap \mathbf{t}^{-m}\widehat{\mathbf{x}} \\
\widehat{\mathbf{x}}' &= \widehat{\mathbf{x}} \cap \mathbf{t}^{m}\mathbf{x} \\
\mathbf{t}' &= \mathbf{t} \cap (\mathbf{x}/\widehat{\mathbf{x}})^{-1/m} \cap (\widehat{\mathbf{x}}/\mathbf{x})^{1/m}.
\end{aligned} \tag{16}$$

The backward propagation `BP` is concerned with reducing the sets $\mathcal{N}(\mathbf{C}_i)$ of all children $\mathbf{C}_i$ of $\mathbf{N}$ using $\mathcal{N}(\mathbf{N})$ and all $\mathcal{N}(\mathbf{C}_j)$ for all other children $\mathbf{C}_j$ with $j \neq i$. Again, first the $\overline{\mathbb{UR}}$ and $\mathbb{I}P\mathbb{R}$ parts are calculated separately by inclusion extensions of the partial inverse functions, followed by an internal intersection operation (16).

Like `FBPD` the `PFBPD` algorithm is contractive and complete in the following sense.

**Proposition 1.** *We define a function* $P : (\overline{\mathbb{UR}} \times \mathbb{I}P\mathbb{R})^n \times 2^{\mathbb{R}^n} \to (\overline{\mathbb{UR}} \times \mathbb{I}P\mathbb{R})^n$ *to represent the* `PFBPD` *algorithm. This function takes as input the variable domains* $\mathbf{B}$ *(in form of a combined interval-union enclosure and an enclosure of the projective transformation) and the exact solution set* $S$ *of the input problem. The function* $P$ *returns an enclosure, denoted by* $P(\mathbf{B}, S)$*, that represents the variable domains of the output of the* `PFBPD` *algorithm, again for the original and the projectively transformed problem. If the input problem is factorable, then the* `PFBPD` *algorithm stops after a finite number of iterations and the following properties hold:*

$$\begin{aligned}
&(i)\ \ P(\mathbf{B}, S) \subseteq \mathbf{B} &&\textit{(Contractiveness)} \\
&(ii)\ P(\mathbf{B}, S) \supseteq \mathbf{B} \cap S &&\textit{(Completeness)}
\end{aligned}$$

The proof is completely analogous to [34, Proposition 2].

## 3 Asymptotic Transformation

As mentioned in Section 2 for MICSPs involving transcendental functions, the projective transformation does not necessarily lead to bounded internal variables $\widehat{y}_\nu$ for all nodes $\nu \in V(\mathit{\Gamma})$. Therefore, we have developed a more general transformation, based on asymptotic analysis.

Let $\mathit{\Psi} \subseteq C(\mathbb{R}^2, \mathbb{R})$ be a subset of functions $\psi(x, t; \alpha)$ depending on the real parameter vector $\alpha \in \mathbb{R}^n$.

For the asymptotic transformation we enclose each intermediate node $y_\nu$ for $\nu \in V(\mathit{\Gamma})$ in the form

$$\psi(\widehat{y}_\nu, t; \underline{\alpha}_\nu) \leq y_\nu \leq \psi(\widehat{y}_\nu, t; \overline{\alpha}_\nu). \tag{17}$$

The new variable $t$ and the parameters $\underline{\alpha}_\nu$ and $\overline{\alpha}_\nu$ are defined such that $t \in [0, 1]$ and the $\widehat{y}_\nu$ are well-bounded. Clearly, the projective transformation is a special case of that scheme, by choosing $\psi(x, t, \alpha) := x/t^\alpha$ with $\underline{\alpha} = \overline{\alpha} = \alpha \in \mathbb{Q}$.

The transformation is like in the projective case achieved by a recursive construction, implemented in a forward walk through $\Gamma$. For the original variables $x_k$ and all interval constants, we define a map $f : \overline{\mathbb{IR}} \to \overline{\mathbb{IR}} \times \mathbb{R}^n \times \mathbb{R}^n$ with the property that for all $t \in [0,1]$ there exists a $\widehat{x}_k \in f_1(\mathbf{x}_k)$ with

$$\psi(\widehat{x}_k, t; f_2(\mathbf{x}_k)) \le x_k \le \psi(\widehat{x}_k, t; f_3(\mathbf{x}_k)), \tag{18}$$

for all $x_k \in \mathbf{x}_k$.

The parameters $\underline{\alpha}_\nu$ and $\overline{\alpha}_\nu$ for an intermediate variable $y_\nu$ depend on the operation that creates it. If $y = g(y_{\nu_1}, \ldots, y_{\nu_\ell})$ then we must have $\widehat{\mathbf{y}}$ such that

$$\psi(\widehat{y}, t; \underline{\alpha}) \le g(\psi(\widehat{y}_{\nu_1}, t; \mathbf{a}_{\nu_1}), \ldots, \psi(\widehat{y}_{\nu_\ell}, t; \mathbf{a}_{\nu_\ell})) \le \psi(\widehat{y}, t; \overline{\alpha}), \tag{19}$$

for $\mathbf{a}_{\nu_i} = [\underline{\alpha}_{\nu_i}, \overline{\alpha}_{\nu_i}]$, all $\widehat{y}_\nu \in \widehat{\mathbf{y}}_\nu$, and some $\widehat{y} \in \widehat{\mathbf{y}}$, and the inequalities should be as tight as possible, $\widehat{\mathbf{y}}$ should be bounded, and there should be a simple way to calculate it for all elementary operations $g$.

If we have a constraint $y \in \mathbf{y}$ we transform it to the two constraints

$$\begin{aligned} \psi(\widehat{y}, t; \underline{\alpha}) &\le \overline{\mathbf{y}} \\ \psi(\widehat{y}, t; \overline{\alpha}) &\ge \underline{\mathbf{y}}, \end{aligned}$$

ensuring that the transformed problem is a relaxation of the original problem.

*Example 3.* A very useful set of functions is $\Psi := \{\psi(x, t; \alpha) := x t^{-\alpha_1} e^{\alpha_2 t^{-\alpha_3}} \mid \alpha \in \mathbb{R}^3\}$. The corresponding transformation is then

$$y_\nu = \widehat{y}_\nu t^{-\alpha_{\nu,1}} e^{\alpha_{\nu,2} t^{-\alpha_3}},$$

where $\overline{\alpha} = \underline{\alpha} = \alpha$. For constructing the $\widehat{y}_\nu$ we choose again a real number $0 \le s \le 1$ and set

$$t := \Big(1 - s + \sum_{k \in K} d_k x_k^2\Big)^{-1/2} \tag{20}$$

with scaling factors $d_k > 0$, like in the projective case. This again leads to the constraint (5).

The parameters $\alpha_\nu$ for an intermediate variable $y_\nu$ depend on the operation that creates it. If, e.g., $y = \sum \beta_\nu y_\nu$ then $y = \widehat{y} t^{-\alpha_1} e^{\alpha_2 t^{-\alpha_3}}$ with

$$\begin{aligned} &\alpha_1 := \max \alpha_{\nu,1}, \quad \alpha_3 := \max \alpha_{\nu,3}, \quad \alpha_2 := \max\{\alpha_{\nu,2} \mid \alpha_{\nu,3} = \alpha_3\}, \\ &\widehat{y} := \sum \beta_\nu t^{\alpha_1 - \alpha_{\nu,1}} e^{-\alpha_2 t^{-\alpha_3}(1 - \frac{\alpha_{\nu,2}}{\alpha_2} t^{\alpha_3 - \alpha_{\nu,3}})} \widehat{y}_\nu, \end{aligned} \tag{21}$$

and we get the finite enclosure

$$\widehat{y} \in \widehat{\mathbf{y}} := \sum \beta_\nu \mathbf{t}^{\alpha_1 - \alpha_{\nu,1}} e^{-\alpha_2 \mathbf{t}^{-\alpha_3}(1 - \frac{\alpha_{\nu,2}}{\alpha_2} \mathbf{t}^{\alpha_3 - \alpha_{\nu,3}})} \widehat{\mathbf{y}}_\nu. \tag{22}$$

Note that the lower bound of the exponential term is 0 by construction.

If $y = \prod y_\nu^{\beta_\nu}$ with real $\beta_\nu$ then $y = \widehat{y} t^{-\alpha_1} e^{\alpha_2 t^{-\alpha_3}}$ with

$$\begin{aligned} \alpha_1 &:= \sum \beta_\nu \alpha_{\nu,1}, \quad & \alpha_3 &:= \max \alpha_{\nu,3}, \\ \alpha_2 &:= \max \beta_\nu \alpha_{\nu,2}, \quad & \widehat{y} &:= e^{\sum \frac{\beta_\nu \alpha_{\nu,2}}{\alpha_2} t^{\alpha_3 - \alpha_{\nu,3}}} \prod \widehat{y}_\nu, \end{aligned} \tag{23}$$

and we get the finite enclosure

$$\widehat{y} \in \widehat{\mathbf{y}} := e^{\sum \frac{\beta_\nu \alpha_{\nu,2}}{\alpha_2} \mathbf{t}^{\alpha_3 - \alpha_{\nu,3}}} \prod \widehat{\mathbf{y}}_\nu. \tag{24}$$

This accounts for all elementary operations and powers with fixed exponent.

For other elementary functions, one can again derive similar formulas, which are rather complex. E.g., if $y = \log y_\nu$ then $y = \widehat{y} t^{-\alpha_1} e^{\alpha_2 t^{-\alpha_3}}$ with

$$\begin{aligned} \alpha_1 &:= \alpha_{\nu,3} + \delta, \quad & \alpha_2 &:= 0 \\ \alpha_3 &:= 0, & \widehat{y} &:= t^{\alpha_1 + \delta} (\log \widehat{y}_\nu - \alpha_{\nu,1} \log t + \alpha_{\nu,2}), \end{aligned}$$

and $\delta = 0$ for $\alpha_1 > 0$, and $\delta = \varepsilon - \alpha_1$ for some small $\varepsilon > 0$, if $\alpha_1 \le 0$, providing a finite enclosure for $\widehat{y}$.

For $y = \exp(\beta y_\nu^\gamma)$ we find $y = \widehat{y} t^{-\alpha_1} e^{\alpha_2 t^{-\alpha_3}}$ with

$$\begin{aligned} &\alpha_1 = 0, \quad \alpha_2 = \sup(\widehat{\mathbf{y}}_\nu^\gamma), \quad \alpha_3 = \alpha_{\nu,1} \gamma \\ &\widehat{y} = e^{t^{-\alpha_1} (\widehat{y}^\gamma e^{\gamma \alpha_{\nu,2} t^{-\alpha_{\nu,3}}} - \alpha_2)}, \end{aligned}$$

giving the enclosure

$$\widehat{y} \in \widehat{\mathbf{y}} := e^{\mathbf{t}^{-\alpha_1} (\widehat{\mathbf{y}}^\gamma e^{\gamma \alpha_{\nu,2} \mathbf{t}^{-\alpha_{\nu,3}}} - \alpha_2)},$$

which is finite for $\alpha_2 \le 0$ or $\alpha_3 \le 0$.

This asymptotic transformation, therefore, can also cope with exponentials, as long as they are not nested.

An analysis of the DAG $\Gamma$ can provide information about which asymptotic transformation is most useful for transforming the MICSP to a bounded form. Of course, a generalization of $\mathbb{I}P\mathbb{R}$ to a more general set of *asymptotic intervals* implementing the above operations provides an algorithm analogous to `PFBPD`.

## 4 Projective and asymptotic relaxations

A very useful tool for solving MICSPs are relaxations of all kind. There are many different classes of relaxations utilized—linear, mixed-integer linear, convex quadratic, semidefinite, general convex to name only the most important

| #var | #problems | PFBPD | | PFBPD $\mathbb{I}P\mathbb{R}$ only | | PFBPD $\overline{\mathbb{UR}}$ only | | FBPD | | HC4 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | #solved | $\Sigma$msec | #solved | $\Sigma$msec | #solved | $\Sigma$msec | #solved | $\Sigma$msec | #solved | $\Sigma$msec |
| 1 | 4 | 4 | 1.30 | 4 | 1.70 | 2 | 0.70 | 1 | 0.20 | 0 | 20.20 |
| 2 | 112 | 112 | 40.90 | 102 | 26.00 | 22 | 21.20 | 5 | 8.00 | 7 | 545.60 |
| 3 | 104 | 57 | 75.80 | 28 | 38.50 | 20 | 33.10 | 7 | 14.90 | 8 | 1354.10 |
| 4 | 70 | 41 | 87.60 | 24 | 41.70 | 22 | 33.20 | 4 | 12.10 | 6 | 1212.60 |
| 5 | 68 | 42 | 89.30 | 17 | 43.80 | 14 | 34.20 | 7 | 16.20 | 8 | 1294.20 |
| 6 | 48 | 23 | 90.20 | 13 | 38.30 | 11 | 28.80 | 6 | 10.10 | 6 | 1404.40 |
| 7 | 18 | 6 | 34.90 | 2 | 12.90 | 1 | 12.00 | 1 | 3.90 | 1 | 444.50 |
| 8 | 43 | 20 | 178.70 | 5 | 43.60 | 5 | 38.90 | 2 | 22.70 | 3 | 1697.20 |
| 9 | 24 | 16 | 52.20 | 10 | 25.60 | 10 | 18.90 | 0 | 6.80 | 1 | 949.60 |
| 10 | 36 | 24 | 66.00 | 10 | 28.60 | 9 | 20.80 | 7 | 9.30 | 7 | 980.60 |
| 11–15 | 32 | 24 | 54.20 | 6 | 26.90 | 7 | 21.50 | 5 | 7.40 | 5 | 667.50 |
| 16–20 | 17 | 13 | 38.90 | 3 | 37.20 | 1 | 16.20 | 1 | 10.10 | 0 | 737.30 |
| 21–30 | 25 | 20 | 116.50 | 3 | 49.60 | 3 | 40.70 | 2 | 17.00 | 2 | 1421.70 |
| 31–46 | 15 | 12 | 80.10 | 2 | 174.00 | 3 | 32.90 | 3 | 40.50 | 3 | 2186.10 |

**Table 1.** Comparison of `PFBPD`, `FBPD`, and `HC4`, easy problems

ones, see [20]. Most of these relaxations come in two flavors: They can be of reformulation type, like reformulation linearization [14, 15, 22], and be much higher dimensional than the original problem. They can also be dimension preserving, like the ones in [21, 27]. However, usually the computation of the relaxations requires that all variables are bounded.

This problem can be overcome by computing a relaxation of a suitably transformed problem, like the projectively transformed problem of Section 2 or the asymptotically transformed problem of Section 3. Even for mixed-integer problems the relaxations have the additional advantage that they are continuous problems. Hence, the transformations can be readily applied.

Reformulation type relaxations can be computed directly from the structure of the operators separately for each node $\nu \in V(\Gamma)$. Dimension preserving relaxations are usually computed by algorithmic differentiation techniques. Those can be generalized to projective or asymptotic intervals by careful examination of the differentiation rules and the properties of first and second order slopes [26].

## 5 Numerical Results

We tested all global optimization and constraint satisfaction problems of the COCONUT test set [28] of dimensions up to 50. They are of general structure

$$\begin{aligned}&\min f(x)\\ &\text{s.t. } F(x) \in \mathbf{F},\ x \in \mathbf{x}.\end{aligned}$$

Of those 865 test problems 15 failed for various reasons (e.g. missing operators, local optimization failed, ...). Of the remaining 850 problems 663 contained at least one unbounded variable. We used local optimization to find at least one local minimum $\tilde{x}$ with objective function value $\tilde{f}$. Then we added the constraint

| Name/Lib. | #var | PFBPD | | PFBPD $\mathbb{I}P\mathbb{R}$ only | | PFBPD $\overline{\mathbb{UR}}$ only | | FBPD | | HC4 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Res. | msec | Res. | msec | Res. | msec | Res. | msec | Res. | msec |
| esfl/3 | 2 | I | 4909.60 | I | 1702.70 | | 1610.00 | I | 181.20 | I | 11.40 |
| pt/2 | 2 | I | 27.60 | I | 12.60 | | 11.20 | | 3.10 | | 818.10 |
| sipow1/2 | 2 | I | 514.20 | I | 264.10 | | 201.30 | | 58.90 | | 314635.00 |
| sipow1m/2 | 2 | I | 517.60 | I | 259.70 | | 206.70 | | 56.80 | | 300657.00 |
| sipow2/2 | 2 | I | 253.60 | I | 121.10 | | 94.90 | | 23.60 | | 73982.50 |
| sipow2m/2 | 2 | I | 256.50 | I | 127.00 | | 99.80 | | 22.60 | | 71689.80 |
| gulf/2 | 3 | | 55.10 | | 21.90 | | 18.20 | | 2.70 | | 222.20 |
| oet1/2 | 3 | I | 83.20 | | 40.20 | | 31.00 | | 92.80 | | 3403.70 |
| oet2/2 | 3 | I | 80.30 | | 34.50 | | 28.60 | | 28.70 | | 3312.80 |
| tfi2/2 | 3 | | 849.70 | | 402.90 | | 358.80 | | 84.20 | | 288961.00 |
| fourbar/3 | 4 | | 20.20 | | 6.90 | | 7.00 | | 1.90 | | 101.00 |
| oet3/2 | 4 | I | 126.60 | | 49.40 | | 45.30 | | 19.60 | | 3848.10 |
| sipow3/2 | 4 | I | 1037.30 | | 391.30 | | 317.90 | | 263.50 | | 318190.00 |
| sipow4/2 | 4 | | 1036.70 | | 459.80 | | 439.00 | | 112.10 | | 315615.00 |
| cpdm5/3 | 5 | | 17.10 | | 6.60 | | 5.40 | | 1.60 | | 90.90 |
| expfitb/2 | 5 | | 29.10 | | 11.30 | | 9.10 | | 2.10 | | 333.30 |
| expfitc/2 | 5 | | 400.20 | | 158.50 | | 115.70 | | 19.40 | | 1959.40 |
| rbpl/3 | 6 | I | 22.50 | | 7.80 | | 7.00 | | 4.30 | | 90.90 |
| oet7/2 | 7 | | 173.30 | | 83.90 | | 67.90 | | 17.10 | | 6352.90 |
| arglinb/2 | 10 | | 14.60 | | 1.60 | | 1.40 | | 1.50 | | 90.90 |
| fir_convex/3 | 11 | | 181.10 | | 60.10 | | 52.90 | | 17.60 | | 1343.30 |
| osborneb/2 | 11 | | 13.40 | | 7.70 | | 5.80 | | 1.30 | | 464.60 |
| watson/3 | 12 | I | 35.20 | | 4.00 | | 3.60 | | 7.50 | | 272.70 |
| ex2_1_10/1 | 20 | I | 23.40 | | 8.20 | | 6.70 | | 0.70 | | 70.70 |
| ex2_1_7/1 | 20 | I | 18.30 | | 6.60 | | 5.40 | | 1.60 | | 60.60 |
| ksip/2 | 20 | | 2174.20 | | 695.40 | | 732.40 | | 233.30 | | 11392.80 |
| antenna2/3 | 24 | | 3552.70 | | 1135.30 | | 1040.90 | | 365.60 | | 14725.80 |
| himmelbk/2 | 24 | I | 47.00 | | 16.00 | | 16.30 | | 5.00 | | 141.40 |
| 3pk/2 | 30 | I | 17.20 | | 3.90 | | 3.60 | | 1.90 | | 80.80 |
| loadbal/2 | 31 | I | 16.70 | | 7.50 | | 6.10 | | 2.80 | | 101.00 |
| lowpass/3 | 31 | | 2760.00 | | 884.80 | | 843.40 | | 285.40 | | 7908.30 |
| watson/2 | 31 | I | 183.70 | | 6.60 | | 6.40 | | 48.30 | | 717.10 |
| hs088/2 | 32 | I | 1358.10 | | 408.30 | | 443.80 | | 253.60 | | 323.20 |
| hs089/2 | 33 | I | 1379.60 | | 453.90 | | 474.20 | | 262.30 | | 363.60 |
| hs090/2 | 34 | I | 1373.10 | | 427.40 | | 420.60 | | 243.10 | | 484.80 |
| hs091/2 | 35 | I | 1390.70 | | 496.00 | | 458.40 | | 237.80 | | 484.80 |
| hs092/2 | 36 | I | 1275.40 | | 436.80 | | 437.00 | | 238.80 | | 676.70 |
| chemeq/3 | 38 | I | 15.70 | | 6.10 | | 5.20 | | 2.50 | | 151.50 |
| polygon2/3 | 38 | I | 41.40 | | 13.20 | | 12.00 | | 3.90 | | 414.10 |
| srcpm/1 | 38 | I | 13.20 | | 4.60 | | 3.80 | | 0.60 | I | 0.70 |
| gridnetg/2 | 44 | I | 49.60 | | 15.20 | | 13.60 | | 4.60 | | 141.40 |
| chnrosnb/2 | 50 | I | 53.10 | | 17.80 | | 16.10 | | 2.70 | | 80.80 |
| errinros/2 | 50 | I | 53.90 | | 17.70 | | 16.20 | | 4.10 | | 70.70 |
| hilbertb/2 | 50 | | 9529.80 | | 2986.50 | | 3034.10 | | 898.80 | | 818.10 |
| qp1/1 | 50 | I | 9407.00 | | 3139.90 | | 2906.00 | | 976.70 | | 575.70 |
| qp2/1 | 50 | I | 9589.70 | | 3115.90 | | 2910.20 | | 993.30 | | 606.00 |
| tointqor/2 | 50 | I | 39.70 | | 12.80 | | 11.50 | | 2.00 | | 70.70 |

**Table 2.** Comparison of `PFBPD`, `FBPD`, and `HC4`, complex problems

$f(x) \leq \tilde{f}$ on the objective function for converting the global optimization problem to a CSP. Then we tried to exclude the region where all unbounded variables are outside the box $[-1000, 1000]$.

These 663 constructed CSPs constitute our test set. As can be deduced from Table 3, for 446 of the problems `PFBPD` was able to prove infeasibility of the problem, effectively reducing the problem to the standard search box $[-1000, 1000]^n$ of global optimization algorithms like BARON. Using only projective intervals solved just 235 of the 675 problems, while pure interval union arithmetic proved infeasibility of only 130 problems. Of those problems 122 are solved by both methods, so they can be considered easy. It is thus indeed important to combine

| case | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $\overline{\mathbb{UR}}$ only | $+$ | $+$ | $-$ | $-$ | $-$ |
| $\mathbb{I}P\mathbb{R}$ only | $+$ | $-$ | $+$ | $-$ | $-$ |
| `PFBPD` | $+$ | $+$ | $+$ | $+$ | $-$ |
| #problems | 122 | 8 | 113 | 203 | 217 |

**Table 3.** Result summary for `PFBPD`

interval unions and projective intervals performing internal intersection after each operation, as described in Algorithm `PFBPD`. All calculations are performed in a completely rigorous way with full rounding error control.

Overall performance of `PFBPD` is very strong; it is comparable to `FBPD` being just a factor of 5-10 on average slower than the interval version and about half as fast as the interval union version of `FBPD`. It is still orders of magnitude faster than `HC4` [2] and many other numerical CP algorithms, as they were tested in [34]. However, there are exceptions like **`srcpm`** where HC4 performs faster and can still prove infeasibility. Detailed results can be found in Table 1, summarizing all easy problems with solution times up to 12 ms, and Table 2, containing the remaining problems. A result of `I` in Table 2 means that infeasibility was proved by the corresponding propagator for the respective problem. The running times were measured on an Intel Core i7 Q 720 running at 1.60GHz running Linux 3.6.11.

## 6 Conclusion

We provided several methods for solving MICSPs for which some variables have unbounded domains. In a large numerical test we showed effectiveness of this new approach.

## References

1. Benhamou, F., Goualard, F.: Universally Quantified Interval Constraints. In: Proceedings of the 6th International Conference on Principles and Practice of Constraint Programming (CP'2000). (2000) 67–82
2. Benhamou, F., Goualard, F., Granvilliers, L., Puget, J.F.: Revising Hull and Box Consistency. In: Proceedings of the International Conference on Logic Programming (ICLP'99), Las Cruces, USA (1999) 230–244
3. Benhamou, F., Older, W.J.: Applying Interval Arithmetic to Real, Integer and Boolean Constraints. Journal of Logic Programming (1997) 32–81
4. Berz, M., Makino, K.: Verified integration of odes and flows using differential algebraic methods on high-order taylor models. Reliable Computing **4** (1998) 361–369

5. Berz, M.: COSY INFINITY version 8 reference manual. Technical report, National Superconducting Cyclotron Lab., Michigan State University, East Lansing, Mich. (1997) MSUCL–1008.
6. Domes, F.: Gloptlab-a configurable framework for solving continuous, algebraic csps. In: IntCP, int. WS on interval analysis, constraint propagation, applications, at CP conference. (2009) 1–16
7. Domes, F.: Gloptlab: a configurable framework for the rigorous global solution of quadratic constraint satisfaction problems. Optimization Methods & Software **24**(4-5) (2009) 727–747
8. Domes, F., Neumaier, A.: Verified global optimization with gloptlab. PAMM **7**(1) (2008) 1020101–1020102
9. Eiermann, M.C.: Adaptive Berechnung von Integraltransformationen mit Fehlerschranken. PhD thesis, Institut für Angewandte Mathematik der Albert–Ludwigs–Universität Freiburg im Breisgau (October 1989)
10. Granvilliers, L., Goualard, F., Benhamou, F.: Box Consistency through Weak Box Consistency. In: Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'99). (November 1999) 373–380
11. Griewank, A., Corliss, G.F.: Automatic Differentiation of Algorithms. SIAM Publications, Philadelphia (1991)
12. Hansen, E.: Global Optimization using Interval Analysis. Marcel Dekker, New York (1992)
13. Jaulin, L., Kieffer, M., Didrit, O., Walter, E.: Applied Interval Analysis. First edn. Springer (2001)
14. Kearfott, R.: Decomposition of arithmetic expressions to improve the behavior of interval iteration for nonlinear systems. Computing **47**(2) (1991) 169–191
15. McCormick, G.: Computability of global solutions to factorable nonconvex programs: Part iconvex underestimating problems. Mathematical programming **10**(1) (1976) 147–175
16. Moore, R.E.: Interval Arithmetic and Automatic Error Analysis in Digital Computing. PhD thesis, Appl. Math. Statist. Lab. Rep. 25, Stanford University (1962)
17. Moore, R.E.: Interval Analysis. Prentice-Hall, Englewood Cliffs, NJ (1966)
18. Neumaier, A.: Interval Methods for Systems of Equations. Cambridge University Press, Cambridge (1990)
19. Neumaier, A.: Taylor forms - use and limits. Reliable Computing **9** (2002) 43–79
20. Neumaier, A.: Complete search in continuous global optimization and constraint satisfaction. Acta Numerica **13**(1) (2004) 271–369
21. Ninin, J., Messine, F., Hansen, P.: A reliable affine relaxation method for global optimization (2010) Optimzation Online.
22. Ryoo, H., Sahinidis, N.: A branch-and-reduce approach to global optimization. Journal of Global Optimization **8**(2) (1996) 107–138
23. Schichl, H.: Global optimization in the coconut project. Numerical Software with Result Verification (2004) 277–293
24. Schichl, H., Markót, M.C.e.a.: The COCONUT Environment. software.
25. Schichl, H., Markót, M.: Algorithmic differentiation techniques for global optimization in the coconut environment. Optimization Methods and Software **27**(2) (2012) 359–372
26. Schichl, H., Neumaier, A.: Exclusion regions for systems of equations. SIAM journal on numerical analysis **42**(1) (2004) 383–408
27. Schichl, H., Neumaier, A.: Interval analysis on directed acyclic graphs for global optimization. Journal of Global Optimization **33**(4) (2005) 541–562

28. Shcherbina, O., Neumaier, A., Sam-Haroud, D., Vu, X.H., Nguyen, T.V.: Benchmarking global optimization and constraint satisfaction codes. In et al., C.B., ed.: Global Optimization and Constraint Satisfaction. Springer, Berlin (2003) 211–222
29. Silaghi, M.C., Sam-Haroud, D., Faltings, B.: Search Techniques for Non-linear CSPs with Inequalities. In: Proceedings of the 14th Canadian Conference on Artificial Intelligence. (2001)
30. Stolfi, J., Andrade, M., Comba, J., Van Iwaarden, R.: Affine arithmetic: a correlation-sensitive variant of interval arithmetic (1994) Web document.
31. Van Hentenryck, P.: Numerica: A Modeling Language for Global Optimization. In: Proceedings of IJCAI'97. (1997)
32. Vu, X.H., Sam-Haroud, D., Silaghi, M.C.: Numerical Constraint Satisfaction Problems with Non-isolated Solutions. In: Global Optimization and Constraint Satisfaction. Volume LNCS 2861., Springer-Verlag (October 2003) 194–210
33. Vu, X., Schichl, H., Sam-Haroud, D.: Using directed acyclic graphs to coordinate propagation and search for numerical constraint satisfaction problems. In: Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on, IEEE (2004) 72–81
34. Vu, X., Schichl, H., Sam-Haroud, D.: Interval propagation and search on directed acyclic graphs for numerical constraint solving. Journal of Global Optimization **45**(4) (2009) 499–531