On feasibility based bounds tightening[0]

Pietro Belotti[1], Sonia Cafieri[2], Jon Lee[3], Leo Liberti[4]

[1] *Dept. of Mathematical Sciences, Clemson University, Clemson SC*
Email: pbelott@clemson.edu

[2] *Laboratoire MAIAA, École Nationale de l'Aviation Civile, 7 Ave. E. Belin, 31055 Toulouse, France*
Email: sonia.cafieri@enac.fr

[3] *Ind. and Op. Eng. Dept., University of Michigan, Ann Arbor MI, USA*
Email: jonxlee@umich.edu

[4] *LIX, École Polytechnique, F-91128 Palaiseau, France*
Email: liberti@lix.polytechnique.fr

January 24, 2012

Abstract

Mathematical programming problems involving nonconvexities are usually solved to optimality using a (spatial) Branch-and-Bound algorithm. Algorithmic efficiency depends on many factors, among which the widths of the bounding box for the problem variables at each Branch-and-Bound node naturally plays a critical role. The practically fastest box-tightening algorithm is known as FBBT (Feasibility-Based Bounds Tightening): an iterative procedure to tighten the variable ranges. Depending on the instance, FBBT may not converge finitely to its *limit ranges*, even in the case of linear constraints. Tolerance-based termination criteria yield finite termination, but not in worst-case polynomial-time. We model FBBT by using fixed-point equations in terms of the variable bounding box, and we treat these equations as constraints of an auxiliary mathematical program. We demonstrate that the auxiliary mathematical problem is a linear program, which can of course be solved in polynomial time. We demonstrate the usefulness of our approach by improving an existing Branch-and-Bound implementation. global optimization, MINLP, spatial Branch-and-Bound, range reduction.

1 Introduction

MINLP (Mixed-Integer Nonlinear Programming) is the class of all MP (Mathematical Programming) problems that might involve integrality constraints on some of the decision variables and nonlinear terms in the objective function and/or constraints. This is a very large class of MPs, which also includes MILP (Mixed-Integer Linear Programming) and NLP (Nonlinear Programming) as subclasses. For our purposes, MINLPs are cast in the form $\min_{x \in \mathscr{X}} f(x)$, where $\mathscr{X}$ includes range constraints $x \in X^0 \in \mathscr{I}^n$ (with $\mathscr{I}$ being the real interval lattice) as well as equality, inequality and integrality constraints.

FBBT (Feasibility-based bounds tightening) is an iterative range-reduction technique that is commonly employed at each node of a sBB (spatial Branch-and-Bound) algorithm for MINLP (see Sect. 1.2 below). FBBT is also known as *bounds propagation* in the CP (Constraint Programming) community. It involves tightening the variable ranges using all of the constraint restrictions; the procedure is iterated as long as the variable ranges keep changing. Because FBBT might fail to converge finitely, in practice one stops iterating when the improvement after one or several iterations is small, say within a given tolerance $\varepsilon > 0$. However, because the improvements are not guaranteed to be monotonically non-increasing, terminating the procedure after one or perhaps several small improvements might in principle overlook the possibility of a larger improvement later on. Furthermore, even when an instance leads to monotonic

[0] This paper extends [7].

improvement, the running time might be exponential in the encoding size. In what follows, we provide a method for *modeling* FBBT by means of fixed-point equations

$$\mathcal{F}(X) = X \ , \tag{1}$$

where $\mathcal{F}$ represents the action of the bounds propagation up and down the expression trees (see Sect. 2.2 below), and $X \in \mathscr{I}^n$. We show that the limit point of FBBT is the largest solution X^* of Eq. (1) in the set-inclusion partial order of the complete lattice $\mathscr{I}^n$. We then write Eq. (1) as a set of constraints derived from the LP (Linear Program) that represents a convex relaxation of the original MINLP, with the objective of minimizing the sum of the variable-range widths. This yields a new method for tightening variable ranges, and also shows that the problem of determining the limit point of FBBT (with $\varepsilon = 0$) acting on continuous intervals and linear constraints can be solved in polynomial time. We also give a sufficient condition for the non-convergence of FBBT (Sect. 3.4). This condition helps identify instances where solving the LP might be more efficient than running FBBT. Lastly, we exploit our theoretical findings to devise a more efficient FBBT-like approach to solving MINLPs (and MILPs) using sBB. Our methodologies allowed us to significantly improve the CPU time taken to find global optima of many publically available MINLP and MILP instances.

1.1 Literature review

FBBT and its relatives have been rediscovered many times from the '70s onwards: within the AI (Artificial Intelligence) community first, and later within the CP, MILP and GO (Global Optimization) communities, more or less at the same time.

1.1.1 Concepts from AI and CP

FBBT is a specialization of a broad concept known in AI and CP as *constraint propagation*, which aims at removing sets of infeasible solutions iteratively until no more change occurs. The ancestor of such techniques is the Waltz algorithm [47], cited in [16] as a propagation operator in a constraint network, applied until stability is reached. Constraint propagation techniques akin to FBBT, also known under the names "domain filtering", "domain reduction", "bound reduction", "range reduction" and "constraint propagation" are widely used in AI and CP, mostly in connection with integer variables [21], but sometimes in connection with continuous variables too [9].

1.1.2 Range reduction in optimization

FBBT was discussed in the context of MILP presolving in [40] as a bounds improvement technique, and in [4] in order to detect whether the original range of a variable appearing in only one constraint is as tight as possible with respect to the constraints. Limited to linear constraints only, FBBT was adopted as a GO technique in [43] (Sect. 9.6.1 — also see [44]) and [42] (also see [38]) under the name of "poor man's LP". General FBBT, incorporating nonlinear functions, was introduced to the GO community in F. Messine's Ph.D. thesis (see [31] and also [32]) in 1997, and recently discussed in more depth in [41, 46]. To the best of our knowledge, the earliest mention in the CP literature of a backward interval propagation on expression trees including nonlinear functions seems to be [9]. See also Sect. 3.3.2 in [21].

1.1.3 Nonconvergence

Because range reduction methods might fail to converge (with $\varepsilon = 0$) in a finite number of steps, the detection of instances where this happens is interesting. Because such methods have a unique greatest fixed point (see Sect. 3), we shall henceforth refer to the lack of finite convergence as *nonconvergence*.

A formal necessary condition for nonconvergence, cited time and again, is that there exist cycles in the variable/constraint incidence graph [20, 18]. We have found no mention of sufficient conditions in the literature; we propose one for FBBT in Sect. 3.4, though we emphasize that it may not be practical to exploit it, and it is far from being necessary.

1.1.4 Worst-case complexity

As concerns the worst-case theoretical complexity of range reduction methods, it is interesting to note that Sect. 3.2.2 in [22] shows that essentially nothing was known in 1992 aside from the empirical observation that such methods were fast in practice. We quote "Problem formulation 1.2" in [22] as the first formal statement of a problem relating to range reduction methods:

> ICSP (Interval Constraint Satisfaction Problem). "Given a set E of equations relating a set of variables associated with interval domains. Refine the domains as far as possible without losing possible exact solutions of E, i.e., determine for each variable the minimal *consistent* subinterval within its domain."

When E consists of linear equations, linear inequalities and range constraints, then a new polynomial-time algorithm for solving the ICSP is briefly discussed in Sect. 2.1 (see Lemma 2.1).

The ICSP only concerns a consistency property. In [11], three formal problem statements are made with respect to convergence properties: the computation of a non-empty interval fixed point (Problem 1, [11]), the computation of a non-empty greatest interval fixed point (Problem 2, [11]), and the existence of a non-empty interval fixed point (Problem 3, [11]). These three problems are analyzed from the point of view of theoretical complexity and found to be NP-hard (NP-complete in the case of Problem 3) *under the assumption that all variables are constrained to be integer*. Again, under the integrality assumption, a very general range-reduction method is shown in [11] to run in pseudo-polynomial time whenever E contains strict linear inequalities involving two variables, quadratic equations of the form $x_i = x_j^2$ for some $i, j \leq n$, and linear equations involving two variables. Naturally, when all variables are constrained to be integer, convergence of range-reduction methods is always attained in a finite number of steps. In this paper, we show that if E consists of linear equations and/or inequalities and there are no integrality restrictions on the variables, the computation of a fixed point for the FBBT can be carried out in polynomial time by simply solving an LP (nonconvergence of FBBT notwithstanding).

1.2 Motivation and sBB

Our motivation for studying the FBBT algorithm is that it is a crucial step of the sBB algorithm for GO, which solves MPs of the form:

$$\left.\begin{array}{rccc} \min_x & x_n & & \\ g^{0L} \leq & g(x) & \leq & g^{0U} \\ x^{0L} \leq & x & \leq & x^{0U} \\ \forall i \in \mathcal{Z} & x_i & \in & \mathbb{Z}, \end{array}\right\} \tag{2}$$

where $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$, $X^0 = [x^{0L}, x^{0U}]$, $G^0 = [g^{0L}, g^{0U}]$, $g : \mathbb{R}^n \to \mathbb{R}^m$ are continuous functions, and $\mathcal{Z} \subseteq \mathcal{N} = \{1, \ldots, n\}$. We remark that a MINLP having an arbitrary objective function $f(x)$ can be easily reformulated to have the form (2), simply by adjoining a constraint $x_n \geq f(x)$. sBB is a Branch-and-Bound variant that performs a recursive search for a global optimum of (2) over the variable bounding box X^0, splitting the current box along a *branching variable* at a certain *branching point*. This search generates a tree, the nodes of which represent the original problem limited to the current box. An upper bound to the objective function value is computed at each node by solving (2) locally [14, 28]. A lower bound to the objective function value is computed at each node by solving a convex relaxation of (2) globally. If the bounds are sufficiently close (say to within a given δ), a global optimum was found in the

current box: it is stored if it improves the incumbent (i.e. the currently best known solution). Otherwise the box is split in two or more sub-boxes. Should a bound be worse than the current incumbent during the search, the domain is discarded (or *pruned by bound*) without branching. It can be shown that if g_i are continuous and $\delta > 0$, then the recursion terminates (exploiting particular problem structure, it is sometimes possible to show termination even with $\delta = 0$, see e.g. [3]). Branch-and-Bound has been used on combinatorial-optimization problems since the 1960s (see [24]); its first application to nonconvex NLPs is [17]. Some MINLP-specific sBB approaches are [37, 5, 1, 2, 44, 45, 26, 8]. The motivation for the present work is that sBB performance is greatly improved if the boxes at each node are reduced; FBBT and other range reduction strategies are discussed in [8].

1.3 Contents

The rest of this paper is organized as follows. We describe some bounds-reduction techniques and the FBBT in detail in Sect. 2. We define an interval MP whose solution is the limit point of FBBT in Sect. 3, giving also a sufficient condition for nonfinite convergence of FBBT in Sect. 3.4. In Sect. 4 we reformulate the interval MP into an LP. We discuss computational results in Sect. 5. Sect. 6 concludes the paper.

2 The FBBT algorithm

In this section, we describe the main bounds tightening techniques, focusing on FBBT. Notationwise, we denote vectors and arrays of real intervals (also called *boxes*) by the roman capital letters X, Y, indexed and emphasized in various ways. For a vector $X = [x^L, x^U]$, the i-th component of X is $X_i = [x_i^L, x_i^U]$. Occasionally, roman capital letters may also denote single intervals instead of vectors/arrays thereof.

2.1 Bounds tightening

During the sBB execution, the initial variable ranges X^0 are restricted to smaller ranges $X = [x^L, x^U] \in \mathscr{I}^n \subseteq X^0$ depending on the current sBB node. Moreover, it often happens that X are not as tight as possible, in the sense that the feasible set given by $g(x) \in G^0$ might be contained in an interval vector much smaller than X. Let $\mathscr{X}$ be the feasible region of (2). The tightest variable bounds $X^* = [x^{*L}, x^{*U}]$ to (2) are defined as follows:

$$\forall i \leq n \quad x_i^{*L} = \min_{x \in \mathscr{X}} x_i \tag{3}$$

$$\forall i \leq n \quad x_i^{*U} = \max_{x \in \mathscr{X}} x_i. \tag{4}$$

However, solving (3) and (4) globally is as difficult as solving the original problem (2). There are two common and well established heuristic techniques for tightening the original bounds X: *feasibility-based bounds tightening* (FBBT), and *optimality-based bounds tightening* (OBBT) [26]. FBBT tightens the bounds by interval arithmetic [35] on each of the functions $g_i(x)$ $(i \leq m)$ and by exploiting the restriction $g(x) \in G^0$. It propagates bounds on the *expression trees* used to represent the functional forms in the problem constraints (see subsection 2.2). The leaf nodes of expression trees represent problem variables and constants, and their non-leaf nodes represent the operators appearing in the function expressions [26]. FBBT propagates bounds from lower to upper nodes; the interval thus obtained for the root node is intersected with the constraint range $[g^L, g^U]$; the tightened root node interval is then propagated down to the leaf nodes, tightening the intervals associated to the nodes and repeating this propagation until convergence to a limit interval vector.

In OBBT, one employs a convex or linear relaxation $\mathscr{R}$ of $\mathscr{X}$ to solve

$$\forall i \leq n \qquad \min_{x \in \mathscr{R}} x_i \tag{5}$$

$$\forall i \leq n \qquad \max_{x \in \mathscr{R}} x_i \tag{6}$$

in order to tighten the variable bounds. The worst-case complexity of OBBT is polynomial, since it needs to solve $2n$ LPs. One might also consider recomputing $\mathscr{R}$ based on the tightened bounds then re-running OBBT: we shall call this two-step procedure OBBT*. Since X (and $\mathscr{R}$) may vary after tightening a variable bound, FBBT and OBBT* can be applied in an iterative way until no more tightening occurs; in this sense, both techniques can be seen as heuristic searches to find good solutions to (3) and (4). Because FBBT is much faster than OBBT*, the former is applied at each sBB node, whilst the latter is usually applied only at the root node or rarely [8]. Both FBBT and OBBT* may fail to converge to the respective limit points: in practice, termination is enforced by stopping the procedures when progress is too slow.

We remark that the box X^{OBBT} obtained by OBBT might be strictly contained in the limit point X^* of FBBT, as examples 3f on p. 23 in [42] and (3.11) on p. 125 in [21] show. We observe that if $\mathscr{R}$ is defined by linear constraints, X^{OBBT} can also be obtained by performing Fourier-Motzkin [48] elimination on $\mathscr{R}$ to project $\mathscr{R}$ on each variable in turn. This shows that OBBT yields a bound-consistent interval vector, as Lemma 2.1 shows, thus providing a polynomial-time algorithm to solve the ICSP.

2.1 Lemma
Let $\bar{X}$ be the vector of ranges obtained by performing Fourier-Motzkin elimination on $\mathscr{R}$ for each variable x_i in turn ($i \leq n$). We have $\bar{X} = X^{\mathsf{OBBT}}$.

Proof. Consider $\bar{x} \in \bar{X}$; then by definition of Fourier-Motzkin reduction, for all $i \leq n$ each $\bar{x}_i$ can be lifted to a $\hat{x} \in \mathscr{R}$. Thus $\min_{x \in \mathscr{R}} x_i \leq \bar{x}_i \leq \max_{x \in \mathscr{R}} x_i$ as claimed. Now take a point $x_i \in X_i^{\mathsf{OBBT}}$ and suppose $x_i \notin \bar{X}_i = [\bar{x}_i^L, \bar{x}_i^U]$ for some $i \leq n$. Either $x_i < \bar{x}_i^L$ or $x_i > \bar{x}_i^U$. In the former case, by definition of projection, there is no $x' \in \mathscr{R}$ such that $x'_i = x_i$, but this contradicts (5); therefore $x_i \in \bar{X}_i$ as claimed. The other case is symmetric. □ □

2.2 Expression trees

Given a constraint $g(x) \in G^0$, where $g : \mathbb{R}^n \to \mathbb{R}$ and $G^0 \in \mathscr{I}$, we associate to the function g its *expression tree* (G, λ), defined as follows:

1. G is a directed tree (V, A) rooted at a distinguished vertex denoted by $\mathsf{root(g)}$
2. λ is a mapping $V \to \mathscr{L}$, where $\mathscr{L} = \mathscr{O} \cup \mathscr{V} \cup \mathbb{R}$, $\mathscr{O}$ is a finite set of operator symbols (for example let $\mathscr{O} = \{+, -, \times, \div, \hat{}, \exp, \log, \sin, \cos\}$) and $\mathscr{V}$ a set of n variable symbols (for example $\{x_1, \ldots, x_n\}$)
3. the following graph properties hold:

$$\forall\, v \in V \quad (\lambda_v \notin \mathscr{O} \quad \to \quad \delta^+(v) = \emptyset) \tag{7}$$

$$\forall\, v \in V \quad (\lambda_v \in \mathscr{O} \quad \to \quad \delta^+(v) = \mathsf{arity}(v)), \tag{8}$$

   where $\delta^+ : V \to \mathcal{P}(V)$, with $\mathcal{P}(V)$ the power set of V, denotes forward vertex stars and $\mathsf{arity}(v)$ is the number of arguments of the operator λ_v;
4. letting $\eta : V \times \mathbb{R}^n \to \mathbb{R}$ be the *evaluation function*, defined recursively as follows:

$$\forall v \in V\ (\lambda_v \in \mathbb{R} \quad \to \quad \eta_v(x^*) = \lambda_v) \tag{9}$$

$$\forall v \in V\ (\lambda_v \in \mathscr{V} \quad \to \quad \eta_v(x^*) = \mathsf{val}(\lambda_v, x^*)\ ) \tag{10}$$

$$\forall v \in V\ (\lambda_v \in \mathscr{O} \quad \to \quad \eta_v(x^*) = \lambda_v(\eta_u(x^*) \mid u \in \delta^+(v))\ ), \tag{11}$$

where $\mathsf{val} : \mathscr{V} \times \mathbb{R}^n \to \mathbb{R}$ is such that $\mathsf{val}(x_i, x^*) = x_i^*$ for all $i \leq n$, the following property holds:

$$\forall x^* \in \mathbb{R}^n \quad \eta_{\mathsf{root}(g)}(x^*) = g(x^*). \tag{12}$$

Since G is an arborescence by definition, for each $v \in V \setminus \{\mathsf{root}(g)\}$ there is a unique element $u \in V$ such that $(u, v) \in A$. We denote u by $\mathsf{parent}(v)$.

2.3 Interval arithmetics

We recall some very basic interval arithmetics notions [35]. Let $\mathscr{I}$ be the set of all intervals with endpoints in $\mathbb{R} \cup \{-\infty, \infty\}$. For two intervals $Y' = [a, b], Y'' = [c, d] \in \mathscr{I}$, where $a \leq b$ and $c \leq d$, and for some $\alpha \geq 0$ and $q \in \mathbb{N}$, the following operations are defined on $\mathscr{I}$:

$$\begin{array}{rcl}
Y' + Y'' & = & [a + c, b + d] \\
\alpha Y' & = & [\alpha a, \alpha b] \\
-Y' & = & [-b, -a] \\
Y' - Y'' & = & Y' + (-Y'') \\
Y' \times Y'' & = & [\min(ac, bc, ad, bd), \max(ac, bc, ad, bd)] \\
1/Y'' & = & \begin{cases} [1/d, 1/c] & 0 < c \leq d \vee c \leq d < 0 \\ [-\infty, \infty] & c < 0 < d \\ [-\infty, 1/c] & c < 0, d = 0 \\ [1/d, \infty] & c = 0, d > 0 \end{cases} \\
Y'/Y'' & = & Y' \times (1/Y'') \\
(Y')^q & = & \begin{cases} [1, 1] & q = 0 \\ [a^q, b^q] & (a \geq 0) \vee (a \leq 0 \leq b \wedge 2 \not| q) \\ [b^q, a^q] & b \leq 0 \\ [0, \max(a, b)^q] & (a \leq 0 \leq b) \wedge 2 | q. \end{cases}
\end{array}$$

2.4 Propagation operators

Whenever $\lambda_v \in \mathscr{O}$, we write $\lambda_v(Y_1, \ldots, Y_p)$ to mean the application of interval arithmetic rules to the intervals $Y_1, \ldots, Y_p$. For all $v \in V$ such that $\lambda_v = x_j$ for some $j \leq n$, we let $\mathsf{index}(v) = j$ (below, when we generalize to sequences of expression trees indexed by i, we denote this by $\mathsf{index}(i, v) = j$). We associate an interval $Y_v = [x_v^L, x_v^U]$ to each node $v \in V$ of an expression tree. Initially, the ranges of the interval vector $Y = (Y_v \mid v \in V)$ are set in function of the value of the original ranges X as follows:

$$\forall v \in V \quad (\lambda_v \in \mathbb{R} \quad \to \quad Y_v = [\lambda_v, \lambda_v]) \tag{13}$$

$$\forall v \in V \quad (\lambda_v \in \mathscr{V} \quad \to \quad Y_v = X_{\mathsf{index}(v)}) \tag{14}$$

$$\forall v \in V \quad (\lambda_v \in \mathscr{O} \quad \to \quad Y_v = [-\infty, \infty]). \tag{15}$$

We write (13)-(15) by $Y = \mathsf{varlift}_V(X)$, and the initial interval vector $\mathsf{varlift}_V(X^0)$ by Y^0. We now define the $\mathsf{up}, \mathsf{down} : V \times \mathscr{I}^{|V|} \to \mathscr{I}$ propagation operators. Informally, the up operator computes the interval of an operator given the intervals of its arguments; more precisely, it is a recursive application of interval analysis to each operator node in $\mathscr{O}$ of the expression tree using the intervals of the subnodes (i.e. nodes in the forward star):

$$\forall v \in V \quad (\lambda_v \notin \mathscr{O} \quad \to \quad \mathsf{up}(v, Y) = Y_v) \tag{16}$$

$$\forall v \in V \quad (\lambda_v \in \mathscr{O} \quad \to \quad \mathsf{up}(v, Y) = Y_v \cap \lambda_v(Y_u \mid u \in \delta^+(v)) \,) \tag{17}$$

The function down is more involved and only modifies the bounding box of operators having a well-defined inverse. It takes each subnode in turn and attempts to tighten its interval by considering interval

arithmetic rules on the inverse operator applied to the other subnodes and the node itself. For all $v \in V \setminus \{\mathsf{root}(g)\}$, let $\mathsf{siblings}(v) = \delta^+(\mathsf{parent}(v)) \setminus \{v\}$ and $\mathsf{family}(v) = \{\mathsf{parent}(v)\} \cup \mathsf{siblings}(v)$.

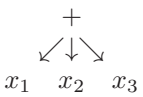
$$\mathsf{down}(\mathsf{root}(g), Y) = Y_{\mathsf{root}(g)} \cap \mathsf{range}(g) \tag{18}$$

$$\forall v \in V \setminus \{\mathsf{root}(g)\} \quad (\ \mathsf{down}(v, Y) = Y_v \ \cap\ \lambda^{-1}_{\mathsf{parent}(v)}(Y_u \mid u \in \mathsf{family}(v))\ ), \tag{19}$$

where $\lambda^{-1}_{\mathsf{parent}(v)}$ denotes the inverse operator of $\lambda_{\mathsf{parent}(v)}$ if it exists and is well defined, and the constant operator mapping each interval vector to $[-\infty, \infty]$ otherwise. For example, the scalar interval multiplication by $\alpha \neq 0$ is inverse to scalar interval multiplication by $\frac{1}{\alpha}$. The operators $\times$ and $\div$ are not always inverse (this depends nontrivially on the signs of the interval bounds).

Finally, we define the function $\mathsf{varproj} : \mathscr{I}^{|V|} \to \mathscr{I}^n$ as $\mathsf{varproj}(Y) = (Y_v \mid \lambda_v \in \mathscr{V})$ as the tool to map the bounding box Y w.r.t. all tree nodes to the bounding box X of the variable nodes. To this aim, we also assume $\mathsf{varproj}(Y)$ to have the same ordering as $\mathscr{V} = (x_1, \ldots, x_n)$. The action of $\mathsf{varproj}$ is inverse with respect to that of $\mathsf{varlift}_V$.

2.2 Example (Inverse of a sum)
For a linear form $x_1 + x_2 + x_3$, represented by the tree



we have $Y_{x_1} = Y_+ - Y_{x_2} - Y_{x_3}$, $Y_{x_2} = Y_+ - Y_{x_1} - Y_{x_3}$, $Y_{x_3} = Y_+ - Y_{x_1} - Y_{x_2}$. Supposing $Y_+ = [0, 0]$, $Y_{x_i} = [0, 1]$ for all $i \leq 3$, we obtain $\mathsf{down}(x_i, Y) = Y_{x_1} \cap ([0, 0] - [0, 1] - [0, 1]) = [0, 1] \cap [-2, 0] = [0, 0]$ for all $i \leq 3$. □

2.3 Example (Inverse of a difference)
For a linear form $x_1 - x_2$, represented by the tree

we have $Y_{x_1} = Y_- + Y_{x_2}$ and $Y_{x_2} = Y_{x_1} - Y_-$. Supposing $Y_- = [1, 1]$ and $Y_{x_i} = [0, 2]$ for $i \leq 2$, we obtain $\mathsf{down}(x_1, Y) = Y_{x_1} \cap ([1, 1] + [0, 2]) = [0, 2] \cap [1, 3] = [1, 2]$ and $\mathsf{down}(x_2, Y) = Y_{x_2} \cap ([0, 2] - [1, 1]) = [0, 2] \cap [-1, 1] = [0, 1]$. □

2.5 Iterating up and down the tree

Each iteration of FBBT alternately applies the up function and the down function. In theory, FBBT should repeat this iteration until no further change to X occurs. This, however, as Ex. 2.4 shows, could result in an infinitely convergent process, an occurrence that is far from rare. In practice, FBBT repeats the iteration until the change to X is "small enough", i.e. until the change in the width sum over all intervals in X falls under a given $\varepsilon > 0$ threshold.

FBBT can be applied in the same way to propagate the effect of sets of constraints: let $g : \mathbb{R}^n \to \mathbb{R}^m$ as in (2) such that $g = (g_i \mid i \leq m)$; let $G_i = (V_i, A_i)$ be the expression tree for g_i, and assume without loss of generality that all variables of $\mathscr{V}$ occur in each g_i. For all $i \leq m$ let $m_i = |V_i|$, $\bar{m} = \sum_{i \leq m} m_i$ and $Y \in \mathscr{I}^{\bar{m}}$ be the vector of intervals whose component Y_{iv} is the range of node $v \in V_i$ (Y^0 is trivially extended to all operators nodes in $\bigcup_{i \leq m} V_i$). Then FBBT can be applied to each constraint g_i of the sequence g in turn, as shown formally in Alg. 1. We remark that changing X at Line 6 has an effect on the next iteration of the loop at Lines 4-7 in particular at Line 5, since $\mathsf{varproj}(Y_i) = X$ for all $i \leq m$.

Because the definition of up and down in Eqs. (17) and (19) involves intersecting the interval operator with the original interval, Alg. 1 at Line 6 always replaces X with an interval which is (not necessarily

Algorithm 1 The FBBT algorithm.

Require: X
1: $Y \leftarrow Y^0$
2: **repeat**
3: $Z \leftarrow X$
4: **for all** $i \leq m$ **do**
5: $Y_i \leftarrow \mathsf{down}(\mathsf{root}(g_i), \mathsf{up}(\mathsf{root}(g_i), Y_i))$
6: $X \leftarrow \mathsf{varproj}(Y_i)$
7: **end for**
8: **until** $X = Z$
9: **return** X

strictly) contained in X. For the same reason, the test at Line 8 can be replaced by $X \subsetneq Z$. Let X^k denote the vector X of variable ranges at the k-th iteration (Lines 3-7) of Alg. 1. Correctness of FBBT follows because for all $k \in \mathbb{N}$, we have $X^k \supseteq X^*$.

2.6 Running time

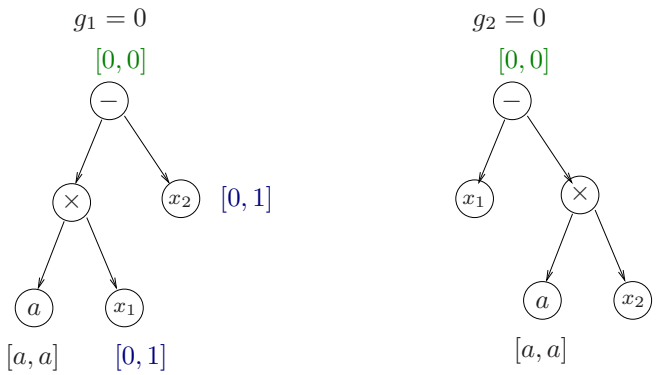
Since the test $X = Z$ in Line 8 of Alg. 1 involves verifying whether two real intervals are equal, the worst-case time complexity of FBBT is ill-defined with respect to the complexity classes P and NP-complete. Furthermore, FBBT is not an exact algorithm, in the sense that, in general, it does not converge to its limit point in finite time (as shown in Example 2.4 below).

Even replacing the test in Line 8 with $\mu(X \triangle Z) \leq \varepsilon$, where μ is the Lebesgue measure in the real line, yields an algorithm whose worst-case time complexity is not polynomial (see Example 2.4): given $\varepsilon > 0$ and an iteration bound k^*, there always exists an instance (a set of constraints g_i, initial ranges X^0) for which FBBT takes a time larger than k^* to converge to within ε.

2.4 Example (Nonconvergence of FBBT)
Consider the following example:

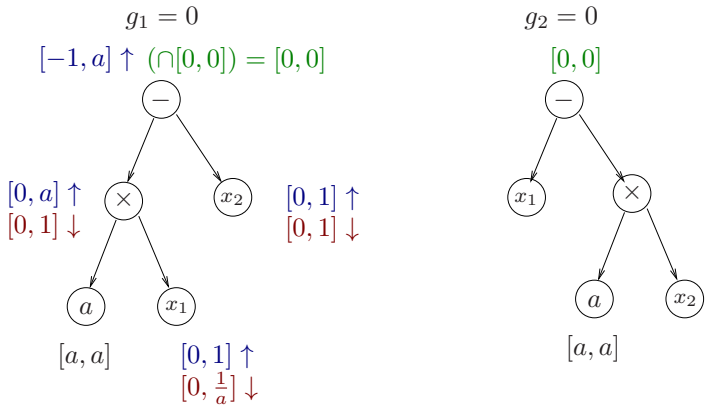
$$\begin{aligned} g_1 \equiv ax_1 - x_2 &= 0 \\ g_2 \equiv x_1 - ax_2 &= 0 \\ x_1 &\in [0,1] \\ x_2 &\in [0,1], \end{aligned}$$

where $a > 1$, represented by the two arborescences below. To each node v of g_i we associate the interval Y_{iv} (for $i \leq 2$).

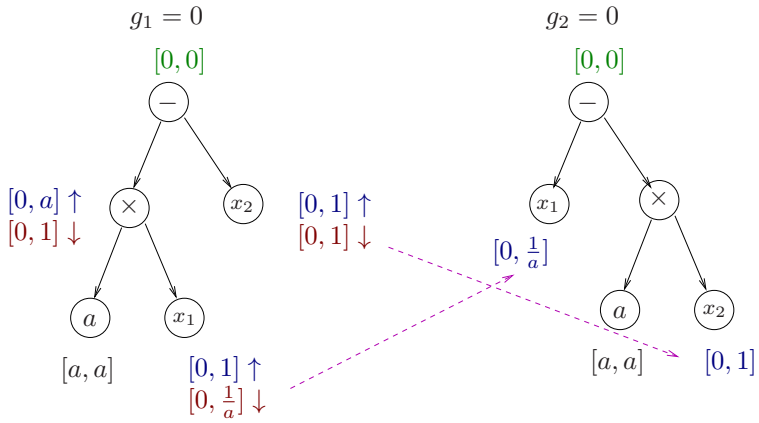


The first step is to apply up *to the leaves of the expression tree encoding g_1: $Y_{1,\times} = [a,a] \times [0,1] = [0,a]$, $Y_{1,-} = [0,a] - [0,1] = [-1,a]$. We then intersect the root node interval $[-1,a]$ with $[0,0]$ (the bounds*

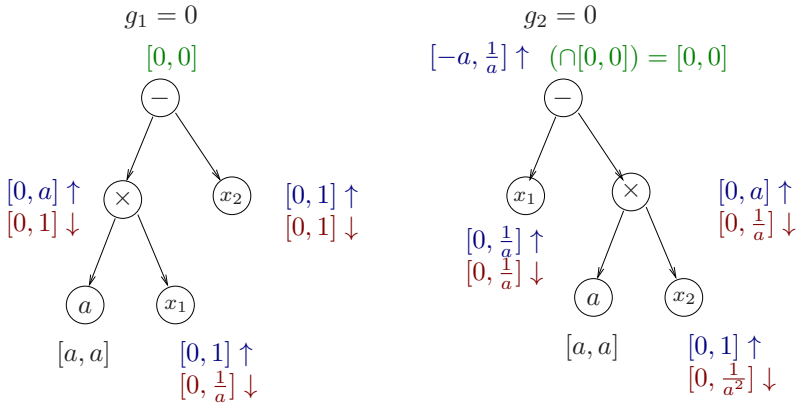
associated to the constraint g_1), then apply down*: $Y_{1\times} = [0,0] + [0,1] = [0,1]$, and since $a > 1$ we tighten to $[0,1] \cap [0,a] = [0,1]$; $Y_{1,x_2} = [0,0] + [0,1] = [0,1]$; $Y_{1,x_1} = \frac{1}{a}[0,1] = [0,\frac{1}{a}]$, which, again since $a > 1$, results in a tightening $[0,\frac{1}{a}] \cup [0,1] = [0,\frac{1}{a}]$.*



We now project the tightened Y_1 onto the variable intervals X and copy these to the variable nodes of the arborescence of g_2.



We apply up *to g_2 and tighten the root node range from $[-a,\frac{1}{a}]$ to $[0,0]$ by intersection with the bounds of g_2. Finally, we apply the* down *operator to g_2. Because X_1 is now $[0,\frac{1}{a}]$ instead of $[0,1]$ as in the case of g_1 above, the* down *application yields a different output, i.e. the range for X_2 becomes $[0,\frac{1}{a^2}]$.*



Continuing in this fashion, it is easy to see that at iteration k we have $X_1 = [0,\frac{1}{a^{2k-1}}]$ and $X_2 = [0,\frac{1}{a^{2k}}]$. It is evident that the limit point $[0,0]$ of FBBT applied to this example is reached in infinite number of

iterations. Supposing now we terminate FBBT when the width sum of X^k differs from that of X^{k-1} by less than a given $\varepsilon > 0$ threshold, we have $|\frac{1}{a^{2k}} - \frac{1}{a^{2k-2}}| < \varepsilon$, which yields $k > \frac{1}{2}(\log_a(a^2-1) - \log_a \varepsilon)$, i.e. for each $a > 1$ there is a $\varepsilon > 0$ that makes the expression on the right hand side an increasing lower bound on k (e.g. take $\varepsilon < (a^2-1)^h$ with $h > 1$ constant). □

3 Semantics of FBBT

3.1 Operators on lattices

A *lattice* is a set Λ partially ordered by the relation $\sqsubseteq$ endowed with two operations $\sqcup$ (*join*), $\sqcap$ (*meet*) such that $x \sqsubseteq x \sqcup y$, $y \sqsubseteq x \sqcup y$ and $x \sqcap y \sqsubseteq x$, $x \sqcap y \sqsubseteq y$. A lattice is *complete* if for all $L \subseteq \Lambda$ both $\sqcup_{x \in L} x$ and $\sqcap_{x \in L} x$ are in Λ (for the interval lattice used in this paper, it suffices to include $\bot, \top$ elements in Λ such that $\bot \sqsubseteq x \sqsubseteq \top$ for all $x \in \Lambda$). An operator $f : \Lambda \to \Lambda$ is *monotone* if $x \sqsubseteq y$ implies $f(x) \sqsubseteq f(y)$. An operator $f : \Lambda \to \Lambda$ is *deflationary* if $f(x) \sqsubseteq x$ for all $x \in \Lambda$. A *fixed point* (fp) of f is an $x \in \Lambda$ such that $x = f(x)$. A fixed point x is *greatest* (denoted gfp(f)) if for all other fixed points x' of f we have $x' \sqsubseteq x$.

3.1 Theorem (Thm. 12.9 in [36] and 8.22 in [15])
Let Λ be a complete partial order and $f : \Lambda \to \Lambda$ a monotone deflationary operator. The sequence $\{f^k(\top) \mid k \in \mathbb{N}\}$ converges to gfp(f).

We now frame FBBT as a monotone deflationary operator in a lattice of interval vectors, and its limit point as its greatest fixed point. This is in accordance with the definition of X^* given in Eq. (3)-(4) as the *largest* box $[x^{*L}, x^{*U}]$ such that for all $j \leq n$ there is at least one feasible point $x \in \mathscr{X}$ with $x_j = x_j^L$ or $x_j = x_j^U$, as well as with the general discussion of Sect. 2.2 in [25].

3.2 The Smith standard form

We reformulate (2) to its Smith standard form [44, 8, 27]: for all $i \leq m$, $v \in V_i$ such that $\lambda_{iv} \in \mathbb{R}$ we replace v by the added variable w_{iv} and adjoin the constraint $w_{iv} = \lambda_{iv}$; for all $i \leq m$, $v \in V_i$ such that $\lambda_{iv} \in \mathscr{V}$ we replace v by the added variable w_{iv} and adjoin the constraint $w_{iv} = x_{\mathsf{index}(i,v)}$; for all $i \leq m$, $v \in V_i$ such that $\lambda_{iv} \in \mathscr{O}$ we (recursively and depth-first) replace the subtree rooted at v by an added variable w_{iv}; we then adjoin a *defining constraint* $w_{iv} = \lambda_v(\delta^+(v))$ to the formulation.

Observe that different w variables might refer to the same term or original variable appearing in different constraints (see the occurrence of x_1, x_2 in the two trees of Ex. 2.4). We therefore define an equivalence relation on the vector of w variables by which $w_{iv} \sim w_{\ell t}$ if and only if the two variables replace the same term or correspond to the same original variable; we constrain the two variables to have the same value with the first constraint set in (20). This partitions the set of w variables into distinct equivalence classes; we denote the class corresponding to (i, v) by $\mathsf{E}(i, v)$. This yields an exact reformulation of (2) as follows:

$$\left.\begin{array}{rrcl} \min_x & x_n & & \\ \forall (i,v) \sim (\ell,t) & w_{iv} & = & w_{\ell t} \\ \forall i \leq m, v \in V_i : \lambda_{iv} \in \mathbb{R} & w_{iv} & = & \lambda_{iv} \\ \forall i \leq m, v \in V_i : \lambda_{iv} \in \mathscr{V} & w_{iv} & = & x_{\mathsf{index}(i,v)} \\ \forall i \leq m, v \in V_i : \lambda_{iv} \in \mathscr{O} & w_{iv} & = & \lambda_{iv}(w_{iu} \mid u \in \delta^+(v)) \\ & w & \in & W^0 \\ \forall i \in \mathscr{Z} & x_i & \in & \mathbb{Z}, \end{array}\right\} \tag{20}$$

where $\forall i \leq m$ ($W^0_{i,\mathsf{root}(g_i)} = G^0_i$) and $\mathsf{varproj}(W^0) = X^0$.

3.3 The FBBT limit point as an interval MP

This reformulation allows us to dispense with multiple expression tree levels: we can therefore re-define $\mathsf{up} : \mathscr{I}^{\bar{m}} \to \mathscr{I}^{\bar{m}}$ with respect to (20) as $\mathsf{up}(W) = (\alpha_{iv} \mid i \leq m \wedge v \in V_i)$ where $W = [w^L, w^U]$ and, for all $i \leq m$ and $v \in V_i$,

$$\alpha_{iv} = \begin{cases} W_{iv} \cap \lambda_{iv}(W_{iu} \mid u \in \delta^+(v)\,) & \text{if } \lambda_{iv} \in \mathscr{O} \\ [\lambda_{iv}, \lambda_{iv}] & \text{if } \lambda_{iv} \in \mathbb{R} \\ W_{iv} & \text{otherwise.} \end{cases} \tag{21}$$

Similarly, we re-define the $\mathsf{down} : \mathscr{I}^{\bar{m}} \to \mathscr{I}^{\bar{m}}$ operator with respect to (20) as $\mathsf{down}(W) = (\beta_{iv} \mid i \leq m \wedge v \in V_i)$ where, for all $i \leq m$ and $v \in V_i$,

$$\beta_{iv} = \begin{cases} W_{iv} \cap \bigcap\limits_{(\ell,t)\in \mathsf{E}(i,v)} \lambda^{-1}_{\ell,\mathsf{parent}(t)}(W_{\ell u} \mid u \in \mathsf{family}(t)) & \text{if } v \neq \mathsf{root}(g_i) \wedge \lambda_{i,\mathsf{parent}(v)} \in \mathscr{O} \\ [\lambda_{iv}, \lambda_{iv}] & \text{if } \lambda_{iv} \in \mathbb{R} \\ W_{iv} & \text{otherwise.} \end{cases} \tag{22}$$

Finally, we can define the fbbt operator on the interval lattice $\mathscr{I}^{\bar{m}}$ as:

$$\mathsf{fbbt}(W) = \mathsf{down}(\mathsf{up}(W \cap W^0) \cap W^0). \tag{23}$$

We remark that the projection on the X-coordinates of the fixed point of the fbbt operator as defined in (23) is the same as the limit point of the FBBT Alg. 1. This follows by the definition of the Smith reformulation (20).

3.2 Example (The linear case)
In the case where $g(x) \in G^0$ only involves linear constraints, written as $Ax \in G^0$ where $A = (a_{ij})$ is an $m \times n$ matrix, the notation can be simplified considerably: we dispense with the auxiliary variables w and only work in the original x-space. In this case we can define up *as an operator $\mathscr{I}^n \to \mathscr{I}^m$ by:*

$$\mathsf{up}(X) = (G_i \cap \sum_{j\leq n} a_{ij} X_j \mid i \leq m), \tag{24}$$

and $\mathsf{down} : \mathscr{I}^m \to \mathscr{I}^n$ *by:*

$$\mathsf{down}(G) = (X_j \cap \bigcap_{\substack{\ell \leq m \\ a_{\ell j} \neq 0}} \frac{1}{a_{\ell j}}(G_\ell - \sum_{k \neq j} a_{\ell k} X_k) \mid j \leq n). \tag{25}$$

The corresponding fbbt *operator is defined as follows:*

$$\mathsf{fbbt}(X) = \mathsf{down}(\mathsf{up}(X \cap X^0) \cap G^0). \tag{26}$$

As shown in Sect. 4, its fixed point can be computed in polynomial time at the cost of solving an LP. □

The following facts are well-known or easy to establish:

1. Virtually all interval arithmetic operators are monotone [34, 33, 35] — certainly all those in $\mathscr{O}$ are.
2. The composition of monotone operators is monotone [36].
3. The up and down operators are deflationary because each of their interval components is an intersection with the original interval (see (21)-(22)).
4. The composition of deflationary operators is deflationary [36].
5. The interval width sum function $|W| = \sum\limits_{\substack{i\leq m \\ v\in V_i}} (w^U_{iv} - w^L_{iv})$ is monotonic with the lattice order.

From Facts 1-4 above and Thm. 3.1 we conclude that the limit point of FBBT 1 is the same as the greatest fixed point of the fbbt operator. In other words,

$$\text{gfp}(\mathsf{fbbt}) = \sup_{\subseteq}\{W \mid W = \mathsf{fbbt}(W)\}. \tag{27}$$

By Tarski's Fixed Point Theorem, we can replace $=$ with $\subseteq$. By Fact 5, (27) is equivalent to:

$$\max\{|W| \mid W \subseteq W^0 \wedge W \subseteq \mathsf{up}(W) \wedge W \subseteq \mathsf{down}(W)\}. \tag{28}$$

Problem (28) is an interval MP, i.e. an MP where the decision variables are intervals. Although there are no standard methods to directly solve such problems in full generality, we give in Sect. 4 an LP reformulation of (28) for problems involving linear constraints only. This gives a polynomial time algorithm for finding the FBBT limit point.

3.4 A sufficient condition for nonconvergence

As was mentioned above, several early AI and CP sources cite the presence of "cycles in the constraint network" as a necessary (but not sufficient) condition for slow convergence of constraint propagation operators. Translated to FBBT terminology, this condition requires the presence of cycles in the graph $\bar{G}$ obtained by the union of all expression trees, followed by contracting each set of leaf nodes representing the same variable, and finally by replacing each arc with an (undirected) edge. In this section we provide a sufficient (albeit not necessary) condition for infinite convergence of the FBBT iteration. Specifically, we show that an FBBT iteration cannot reduce the dimension of the affine hull of a bounding box (under mild conditions); thus, if the affine hull of the FBBT limit point has dimension strictly less than the dimension of the bounding box, then FBBT cannot converge to its limit point in finite time.

Let $W^* = \text{gfp}(\mathsf{fbbt})$.

3.3 Theorem
Let $d = \dim \text{aff}(W^0) > \dim \text{aff}(W^*) = d'$ and assume that: (a) without loss of generality all the intervals in W^0 have positive width; (b) the relative interior of W^0 contains a point x' feasible in (20)*; (c) all the operators in $\mathscr{O}$ are C^∞. Then there is no $k \in \mathbb{N}$ such that $\mathsf{fbbt}^k(W^0) = W^*$.*

Proof. Suppose the contrary; then there is a smallest iteration index $h \in \{1, \ldots, k\}$ when the dimension $\dim \text{aff}(\mathsf{fbbt}^h(W^0))$ decreases. Hence, there is $i \leq m$ and $v \in V_i$ with $\lambda_v \notin \mathbb{R}$ such that W_{iv} has positive width but either α_{iv} or β_{iv} (defined in (21) and (22)) contain a single real. If $\lambda_{iv} \in \mathscr{V}$ then this is impossible by hypothesis (a), as $\alpha_{iv} = W_{iv}$ and $\beta_{iv} = W_{iv}$. Therefore, $\lambda_{iv} \in \mathscr{O}$. Suppose $\alpha_{iv} = \{a\}$, i.e. the intersection of the intervals W_{iv} and $\lambda_{iv}(W_{iu} \mid u \in \delta^+(v))$ only contains a. This can only happen for the following reasons: (i) a is a common bound of the intervals W_{iv} and $\lambda_{iv}(W_{iu} \mid u \in \delta^+(v))$ (a lower bound for one and an upper bound for the other); (ii) $W_{iv} = \{a\}$; (iii) $\lambda_{iv}(W_{iu} \mid u \in \delta^+(v)) = \{a\}$. Case (ii) cannot occur because of hypothesis (a); case (iii) cannot occur because λ_{iv} is C^∞; case (i) implies that all feasible points of (20) are contained within the hyperplane $w_{iv} = a$, which, because of (a), goes against the hypothesis (b). The argument for $\beta_{iv} = \{a\}$ is similar, remarking that by definition of down when λ_{iv}^{-1} is not naturally defined, then it is considered by to map every interval argument to $[-\infty, \infty]$. We conclude the proof by noticing that since there is no $k \in \mathbb{N}$ that causes d to decrease, the hypothesis holds whenever W^0 is replaced by $\mathsf{fbbt}^k(W^0)$. □ □

Theorem 3.3 generalizes the nonconvergence result of Example 2.4. Unfortunately, it does not by itself provide a detection device for nonconvergent cases (since it assumes prior knowledge of the dimension of the affine hull of the fixed point W^*).

4 FBBT on the linear MINLP relaxation

The Smith reformulation described in Sect. 3.2 is commonly used to derive a convex or linear relaxation of the original MINLP (2): each of the defining constraints is replaced by a system of convex or linear inequalities approximating the convex hull of $\mathscr{X}$. Optimizing on this relaxation yields a lower bound on the optimal objective function value of (2), as described in Sect. 1.2. We assume in the following that a linear relaxation is available, formulated as the following LP:

$$\left.\begin{array}{lrcl} \min & z_p & & \\ & Az & \in & B^0 \\ & z & \in & Z^0, \end{array}\right\} \tag{29}$$

where $z \in \mathbb{R}^p$, A is $q \times p$, $B^0 \in \mathscr{I}^q$ and $Z^0 \in \mathscr{I}^p$. We remark that the z variables in (29) include all problem variables of (20).

Since (29) is an LP, the FBBT definition given in Example 3.2 applies. It yields an interval LP as follows:

$$\max\{|Z| \mid Z \subseteq Z^0 \wedge B \subseteq B^0 \wedge B \subseteq \mathsf{up}(Z) \wedge Z \subseteq \mathsf{down}(B)\}, \tag{30}$$

where $Z = [z^L, z^U] \in \mathscr{I}^p$ and $B = [b^L, b^U] \in \mathscr{I}^q$ are interval decision variables. Problem 30 can be reformulated as a standard LP as detailed below. We define $S_i^+ = \{j \leq p \mid a_{ij} > 0\}$ and $S_i^- = \{j \leq p \mid a_{ij} < 0\}$ for all $i \leq q$.

$$\max \sum_{j \leq p} (z_j^U - z_j^L) \tag{31}$$

$$\forall j \leq p \quad z_j^L \leq z_j^U \tag{32}$$

$$\forall i \leq q \quad b_i^L \leq b_i^U \tag{33}$$

$$\forall j \leq p \quad z_j^L \geq z_j^{0L} \tag{34}$$

$$\forall j \leq p \quad z_j^U \leq z_j^{0U} \tag{35}$$

$$\forall i \leq q \quad b_i^L \geq b_i^{0L} \tag{36}$$

$$\forall i \leq q \quad b_i^U \leq b_i^{0U} \tag{37}$$

$$\forall i \leq q \quad b_i^L \geq \sum_{j \in S_i^+} y_{ij}^L + \sum_{j \in S_i^-} a_{ij} z_{ij}^U \tag{38}$$

$$\forall i \leq q \quad b_i^U \leq \sum_{j \in S_i^+} y_{ij}^U + \sum_{j \in S_i^-} a_{ij} z_{ij}^L \tag{39}$$

$$\forall i \leq q, j \in S_i^+ \quad z_j^L \geq \frac{1}{a_{ij}} (b_i^L - \sum_{\ell \in S_i^+ \smallsetminus \{j\}} a_{i\ell} z_\ell^U - \sum_{\ell \in S_i^- \smallsetminus \{j\}} a_{i\ell} z_\ell^L) \tag{40}$$

$$\forall i \leq q, j \in S_i^+ \quad z_j^U \leq \frac{1}{a_{ij}} (b_i^U - \sum_{\ell \in S_i^+ \smallsetminus \{j\}} a_{i\ell} z_\ell^L - \sum_{\ell \in S_i^- \smallsetminus \{j\}} a_{i\ell} z_\ell^U) \tag{41}$$

$$\forall i \leq q, j \in S_i^- \quad z_j^L \geq \frac{1}{a_{ij}} (b_i^L - \sum_{\ell \in S_i^+ \smallsetminus \{j\}} a_{i\ell} z_\ell^L - \sum_{\ell \in S_i^- \smallsetminus \{j\}} a_{i\ell} z_\ell^U \tag{42}$$

$$\forall i \leq q, j \in S_i^- \quad z_j^U \leq \frac{1}{a_{ij}} (b_i^U - \sum_{\ell \in S_i^+ \smallsetminus \{j\}} a_{i\ell} z_\ell^U - \sum_{\ell \in S_i^- \smallsetminus \{j\}} a_{i\ell} z_\ell^L. \tag{43}$$

The objective function (31) maximizes the interval width sum. Constraints (32)-(33) provide the definition of an interval decision variable; Constraints (34)-(35) model $Z \subseteq Z^0$; similarly, Constraints (36)-(37) model $B \subseteq B^0$; Constraints (38)-(39) model $B \subseteq \mathsf{up}(Z)$ and Constraints (40)-(43) model $Z \subseteq \mathsf{down}(B)$. We call the formulation (31)-(43) "FBBT-LP".

The above discussion can be summarised in the following theorem.

4.1 Theorem
If the limit point of FBBT is not the empty interval, then the solution of FBBT-LP provides the limit point of FBBT acting on (29) *in polynomial time.*

4.1 Infeasibility

It might sometimes happen that at a given iteration of FBBT, one of the interval components $[z_i^L, z_i^U]$ of the interval vector Z might be updated in such a way that $z_i^L > z_i^U$. In such cases we can establish that $Z = \varnothing$ and hence that (29) is infeasible. The converse does not hold in general: there are infeasible problems (29) for which FBBT does not converge to an empty interval vector. The former case is beneficial in sBB, because it allows us to prune the node without further work. A similar mechanism is found in the FBBT-LP, which includes constraints (32) that explicitly make empty intervals infeasible. Since infeasibility is easy to detect in LPs, we exploit this fact to detect infeasibility in (29).

4.2 Proposition
If the FBBT-LP is infeasible, (29) *is infeasible.*

Proof. Assume (29) is feasible, then it has an optimal solution z^*, which must be contained in some ranges Z'. Thus, the limit point of FBBT must contain Z', which implies that $Z^* = \text{gfp}(\mathsf{fbbt}) \supseteq Z'$, where Z^* is the optimal solution to the FBBT-LP, which is therefore feasible. □

We remark that we are *not* stating that every time FBBT has $z_i^L > z_i^U$ at some iteration for some component i, then the FBBT-LP is infeasible. Although in our computational experiments we found that this statement holds in most cases, we also found one case where this does not hold. Accordingly, Thm. 4.1 only holds when the limit point of FBBT is not empty. Conceiving an LP that also caters explicitly for empty intervals seems a difficult task, and might require the introduction of binary variables, as was done in [29].

4.2 Reducing the formulation size

The size of the FBBT-LP is dependent on the number of variables p of the reformulation (i.e., original and auxiliaries) and on the number of inequalities q of the linearization. Given that several MINLP approaches [8] are of a *Branch-and-Cut* variety, i.e., sBB amended with a cutting-plane algorithm at each sBB node, q can grow very large. The purpose of this section is to demonstrate that a more compact LP formulation can be used to obtain the same result. Specifically, we eliminate the $2q$ variables b^L, b^U.

To ease notation, we rewrite (29) as follows:

$$\begin{array}{ll} \min & z_p \\ s.t. & b_i^{0L} \leq a^i z \leq b_i^{0U} \quad i = 1, 2 \ldots, m, \end{array} \tag{44}$$

with $a^i = (a_{ij} \mid j \leq n)$. For the sake of clarity and conciseness, we introduce the following notation: a^{i+} is the vector whose j-th component is $\max\{0, a_{ij}\}$, for $j \leq p$. Analogously, a^{i-} is defined as the vector whose j-th component is $\min\{0, a_{ij}\}$, for $j \leq p$ (thus $a^i z = a^{i+} z + a^{i-} z$). Also, for each j such that $a_{ij} > 0$ we define $a^{j,i+} = a^{i+} - a_{ij} e^j$, where e^j is the p-vector whose j-th component is one and all remaining components are zero, and similarly, for each j such that $a_{ij} < 0$, we define $a^{j,i-} = a^{i-} - a_{ij} e^j$. In other words, $a^{j,i+}$ (resp. $a^{j,i-}$) is equal to a^{i+} (resp. a^{i-}) except for the j-th element, which is set to 0.

Let us use two vectors z^L and z^U to represent the $2p$ variables z_j^L and z_j^U, for each $j \leq p$, with the same meaning as in the FBBT-LP: lower and upper bounds of the fixed point. Then, if z_j^L is a lower

bound to (3) and z_j^L is an upper bound to (4), the following inequalities are valid (they are simple implied bounds inequalities):

$$\forall i \leq m, \forall j \leq n : a_{ij} > 0 \quad a_{ij} z_j^L \geq \quad b_i^{0L} - (a^{j,i+} z^U + a^{i-} z^L) \tag{45}$$
$$\forall i \leq m, \forall j \leq n : a_{ij} > 0 \quad a_{ij} z_j^U \leq \quad b_i^{0U} - (a^{j,i+} z^L + a^{i-} z^U) \tag{46}$$
$$\forall i \leq m, \forall j \leq n : a_{ij} < 0 \quad -a_{ij} z_j^L \geq \quad -b_i^{0U} + (a^{i+} z^L + a^{j,i-} z^U) \tag{47}$$
$$\forall i \leq m, \forall j \leq n : a_{ij} < 0 \quad -a_{ij} z_j^U \leq \quad -b_i^{0L} + (a^{i+} z^U + a^{j,i-} z^L). \tag{48}$$

Note that these inequalities do not involve variables b_i^L or b_i^U from FBBT-LP. A new, more compact LP formulation can then be defined as follows:

$$\min\{(31) : (32), (34), (35), (45) - (48)\}. \tag{49}$$

It is worth noting that (45)-(48) are equivalent to (40)-(43) except for variables b_i^L and b_i^U being replaced by b_i^{0L} and b_i^{0U}, respectively. We call (49) "FBBT-LPC".

4.3 Proposition
The feasible set of the FBBT-LPC is a projection on the (z^L, z^U)-space of the feasible set of the FBBT-LP.

Proof. We apply the Fourier-Motzkin procedure to project out variables b_i^L and b_i^U for all $i \leq q$: we first isolate all inequalities containing b^L, then analyze and rearrange the term of this subsystem, and finally do the same for b^U.

The inequalities of the FBBT-LP involving b_i^L variables are as follows:

$$\begin{array}{lll} b_i^L \geq b_i^{0L} & & \text{from (36)} \\ b_i^L \geq a^{i+} z^L + a^{i-} z^U & & \text{from (38)} \\ b_i^L \leq b_i^U & & \text{from (33)} \\ b_i^L \leq a^{j,i+} z^U + a^{i-} z^L + a_{ij} z_j^L & \forall j \leq n : a_{ij} > 0 & \text{from (40)} \\ b_i^L \leq a^{i+} z^U + a^{j,i-} z^L + a_{ij} z_j^U & \forall j \leq n : a_{ij} < 0 & \text{from (43).} \end{array}$$

The variables b_i^L (for $i \leq q$) are eliminated by combining the lower and upper bounding expressions for b_i^L:

$$b_i^U \geq b_i^{0L} \tag{50}$$
$$\forall j \in S_i^+ \quad a^{j,i+} z^U + a^{i-} z^L + a_{ij} z_j^L \geq b_i^{0L} \tag{51}$$
$$\forall j \in S_i^- \quad a^{i+} z^U + a^{j,i-} z^L + a_{ij} z_j^U \geq b_i^{0L} \tag{52}$$
$$b_i^U \geq a^{i+} z^L + a^{i-} z^U \tag{53}$$
$$\forall j \in S_i^+ \quad a^{j,i+} z^U + a^{i-} z^L + a_{ij} z_j^L \geq a^{i+} z^L + a^{i-} z^U \tag{54}$$
$$\forall j \in S_i^- \quad a^{i+} z^U + a^{j,i-} z^L + a_{ij} z_j^U \geq a^{i+} z^L + a^{i-} z^U. \tag{55}$$

Inequalities (54) and (55) are always satisfied when (32) holds. We show this for (54) only (the case of (55) is similar): eq. (54) is equivalent to

$$a^{j,i+} z^U + a_{ij} z_j^L + a^{i-} z^L \geq a^{j,i+} z^L + a_{ij} z_j^L + a^{i-} z^U \qquad \forall j \leq n : a_{ij} > 0;$$

the above always holds because $a^{j,i+} z^U \geq a^{j,i+} z^L$, $a_{ij} z_j^L$ cancels out and $a^{i-} z^L \geq a^{i-} z^U$ by definition of a^{i-} and (32).

Moreover, (51) and (52) are rearrangements of (45) and (48), which in turn are simply (40) and (43).

In order to eliminate the variables b_i^U (for $i \leq q$), we list the inequalities involving them:

$$\begin{array}{rcl} b_i^U & \geq & b_i^{0L} \\ b_i^U & \geq & a^{i+}z^L + a^{i-}z^U \\ b_i^U & \leq & b_i^{0U} \\ b_i^U & \leq & a^{i+}z^U + a^{i-}z^L \\ \forall j \in S_i^+ \quad b_i^U & \leq & a^{j,i+}z^U + a^{i-}z^L + a_{ij}z_j^L \\ \forall j \in S_i^- \quad b_i^U & \leq & a^{i+}z^U + a^{j,i-}z^L + a_{ij}z_j^U, \end{array}$$

and apply the same procedure as above:

$$b_i^{0U} \geq b_i^{0L} \tag{56}$$
$$\forall j \in S_i^+ \quad a^{j,i+}z^U + a^{i-}z^L + a_{ij}z_j^L \geq b_i^{0L} \tag{57}$$
$$\forall j \in S_i^- \quad a^{i+}z^U + a^{j,i-}z^L + a_{ij}z_j^U \geq b_i^{0L} \tag{58}$$
$$a^{i+}z^U + a^{i-}z^L \geq b_i^{0L} \tag{59}$$
$$b_i^{0U} \geq a^{i+}z^L + a^{i-}z^U \tag{60}$$
$$\forall j \in S_i^+ \quad a^{j,i+}z^U + a^{i-}z^L + a_{ij}z_j^L \geq a^{i+}z^L + a^{i-}z^U \tag{61}$$
$$\forall j \in S_i^+ \quad a^{i+}z^U + a^{j,i-}z^L + a_{ij}z_j^U \geq a^{i+}z^L + a^{i-}z^U \tag{62}$$
$$a^{i+}z^U + a^{i-}z^L \leq a^{i+}z^L + a^{i-}z^U. \tag{63}$$

Inequality (56) is trivial, while inequalities (61)-(62) and (63) are again satisfied when (32) holds. Inequalities (57) and (58) are rearrangements of (46) and (47) respectively. Inequalities(59)-(60) say that the upper and lower interval bounds to the linear form az contain the constraint RHS interval B_i^0, and are therefore satisfied as long as (44) is feasible.

Hence, for a feasible (44) the only irredundant constraints are (32), (34), (35), (45)-(48), which completes the proof. □ □

4.4 Corollary
The set of optimal solutions of the FBBT-LPC is a projection on the (z^L, z^U)-space of the set of optimal solutions of the FBBT-LP.

Proof. Notice that (31) is a function of z^L, z^U only. □ □

5 Computational results

We provide two types of tests: consistency tests, and field tests. Our consistency tests aim to support the theory by showing that its predictions are verified in computational experiments. Specifically, we show that the FBBT-LPC formulation given in Sect. 4.2 yields interval vectors that are always contained in those obtained by direct application of FBBT. In our field tests we experiment with two strategies using direct FBBT and solution of the FBBT-LPC in Couenne [8], and show how they compare to a pure FBBT application.

All tests were carried out using the `trunk` version of Couenne. Apart from the parameters discussed below that are used to enable FBBT or FBBT-LPC, standard parameter values were used: reliability branching, standard heuristics for finding feasible solutions, and *convexification cuts* for creating a linear relaxation of the MINLP were applied. All tests have been conducted on the Palmetto cluster at Clemson University, which employs several computers equipped with different processors and RAM memory.

The field tests consist of comparing the different variants of Couenne mentioned above applied to the same MINLP or MILP instance: in this case, although two instances may have been solved on different machines, all variants of Couenne used the same machine for each instance.

5.1 Consistency tests

For these tests, we devised a very special class of LP instances, designed on the basis of Thm. 3.3 in order to prevent finite convergence of FBBT. The formulation of these instances, which we call `nonconvergent`, is:

$$\left.\begin{array}{rrcl} \min & \sum\limits_{j\leq n} x_j & & \\ & Ax & \leq & b \\ & Cx & = & d \\ & -M \leq x & \leq & M, \end{array}\right\} \tag{64}$$

where A is a randomly generated $m \times n$ matrix, C has shape $(0|C')$ with C' a random invertible $p \times p$ matrix, b, d are randomly generated vectors in $\mathbb{R}^m$ and $\mathbb{R}^p$ respectively. M is a given real value (set to 10 in our experiments) and each random coefficient is chosen in $[-M, M]$.

Since C' is nonsingular, $x' = (x_{n-p+1}, \ldots, x_n)$ is determined by the equality constraints $C'x' = d$. This makes it more likely to actually find a FBBT limit point X^* such that $\delta' = \dim \mathrm{aff}(X^*) < \dim \mathrm{aff}([-M, M]) = \delta$, as required by Thm. 3.3.

We generated 190 `nonconvergent` instances using values of m in $\{10, 20, 30\}$, values of n in $\{2, 4, \ldots, m\}$ and values of p in $\{2, 4, \ldots, n\}$. Of these instances, 36 turned out to have the required property $\delta' < \delta$; and of these, 20 were found to be feasible and 16 to be infeasible (this suited our purposes as it allowed us to verify Prop. 4.2).

In Fig. 1, we show the CPU time difference and the final interval vector width difference, defined as the sum of the interval width differences over each component, between FBBT terminated when the discrepancy between one iteration and the next decreased to less than 10^{-4} and the FBBT limit point found using the FBBT-LPC formulation (on the feasible instances only). The results are consistent

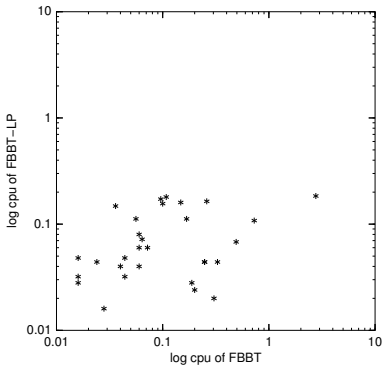


Figure 1: CPU time difference between FBBT and FBBT-LPC. Points under the diagonal denote instances where FBBT-LPC is more performant.

with the theoretical predictions: the output interval width difference between FBBT and the FBBT-LPC is always positive. The values are very small because of the FBBT termination condition. As for the infeasible instances, FBBT was able to identify 6 instances out of 16; the corresponding score for FBBT-LPC was 5 out of 16.

5.2 Field tests

We ran extensive tests with a modified version of the Couenne [8] solver. Our test set includes the GlobalLib [12], a set of pooling and blending problems from [19], the MINLPLib [13], and the union of MIPLib3 [10] and MIPLib2003 [30] (denoted as MIPLib in the following), for a total of 706 instances.

5.2.1 Tested versions of the Couenne solver

We benchmark two modified versions of Couenne with the standard version. Each version is described by a pair of integers (f, ℓ): f gives the maximum number of iterations that FBBT is permitted to take at each sBB node; ℓ can be either 0 (the FBBT-LPC is disabled), 1 (the FBBT-LPC is solved at each node) or a negative integer (the FBBT-LPC is solved every ℓ-th node if successful at the root node, otherwise never). The standard Couenne version is $(5, 0)$. The other versions we use are a $(0, 1)$, a pure FBBT-LPC where FBBT is disabled, and a hybrid approach $(3, -5)$.

5.2.2 CPU time

The first performance measure is the user CPU time (in seconds) taken to solve an instance (or deem it infeasible). We only compare this measure over a subset of 415 instances for which all tested versions of Couenne terminate before the 2h time limit. The aggregated statistics are reported in Table 1. We

Library	Standard $(5, 0)$	FBBT-LPC $(0, 1)$	Hybrid $(3, -5)$
GlobalLib	22551	11891	**9847**
Pooling problems	5133	13329	**46**
MINLPLib	27395	11668	**9395**
MIPLib	17016	**13186**	25592
Total	72095	50074	**44880**

Table 1: CPU time comparison of Couenne $(5, 0)$, $(0, 1)$ and $(3, -5)$.

remark that the hybrid Couenne is more effective for nonlinear problems, whereas the pure FBBT-LPC version is more effective for linear ones. It is interesting to remark that the standard FBBT approach, which is what is currently implemented in most solvers (Couenne, BARON [39], CPLEX [23] as part of the presolver), seems to be at a loss overall.

We also compare the FBBT-LPC and hybrid Couenne versions against the standard one using scatter plots (Fig. 2-4) where the x-axis is $\log\log$ CPU of the standard version and the y-axis is $\log\log$ CPU of the FBBT-LPC and hybrid versions. Points under the diagonal denote instances for which the FBBT-LPC and hybrid versions are faster. The hybrid Couenne definitely has a CPU time advantage with respect to the standard one.

5.2.3 Optimality gap closed

The second performance measure is the optimality gap closed at termination on the 60 instances where all Couenne versions terminate because of a reached user CPU time limit of 2h. The *closed gap* is measured as:

$$\frac{LB \text{ at termination} - LB \text{ at root node}}{\text{smallest } UB \text{ over Couenne versions} - LB \text{ at root node}},$$

as detailed in [8]. The aggregated statistics are reported in Table 2. It turns out that the standard Couenne closes slightly more gap on nonterminating instances. The standard deviation of the aggregated

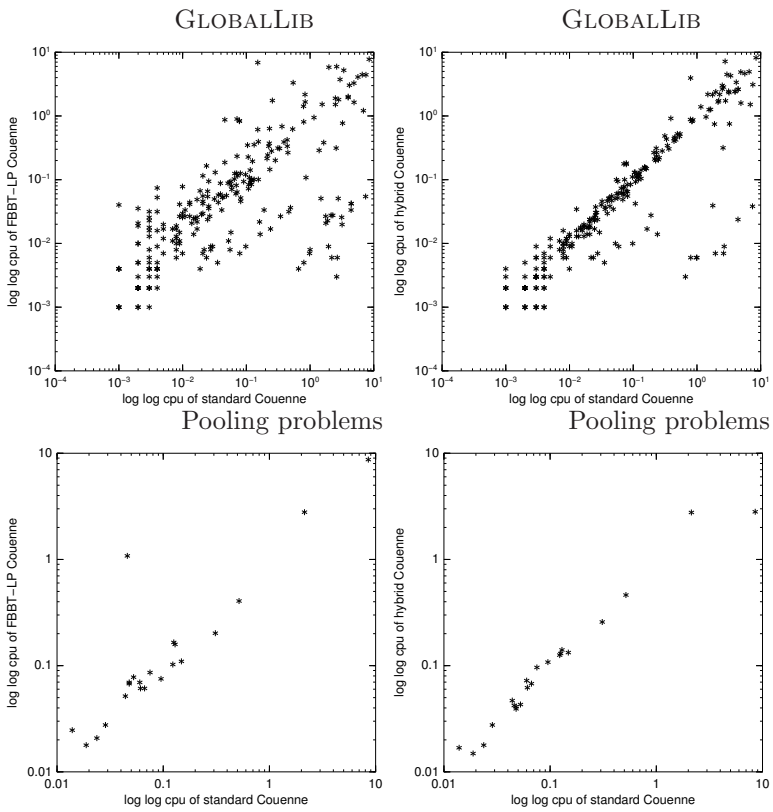


Figure 2: User CPU time scatter plots of FBBT-LPC (left) and hybrid (right) versus standard Couenne versions on continuous NLP instances from GlobalLib (top) and the pooling problems (bottom).

Library	Standard $(5,0)$	FBBT-LPC $(0,1)$	Hybrid $(3,-5)$
GlobalLib	**6.01**	5.43	5.85
Pooling problems	0.84	0.45	**0.86**
MINLPLib	**0.40**	0.37	0.20
MIPLib2003	5.15	**5.54**	5.21
Total	**12.41**	11.80	12.12

Table 2: Closed optimality gap comparison of Couenne $(5,0)$, $(0,1)$ and $(3,-5)$.

statistic over all Couenne versions is very low, however. It is worth mentioning that, on MILPs, the FBBT-LPC version fares better than the others.

5.2.4 Categorical comparison

In this section, we draw a comparison on the set of 202 instances on which the Couenne versions are discordant as regards the natural or artificial (CPU time limit) termination. Table 3 reports the number of discordant instances on which the different versions of Couenne terminated naturally. It appears clear that the hybrid version of Couenne is able to solve more instances within the 2h CPU time limit.

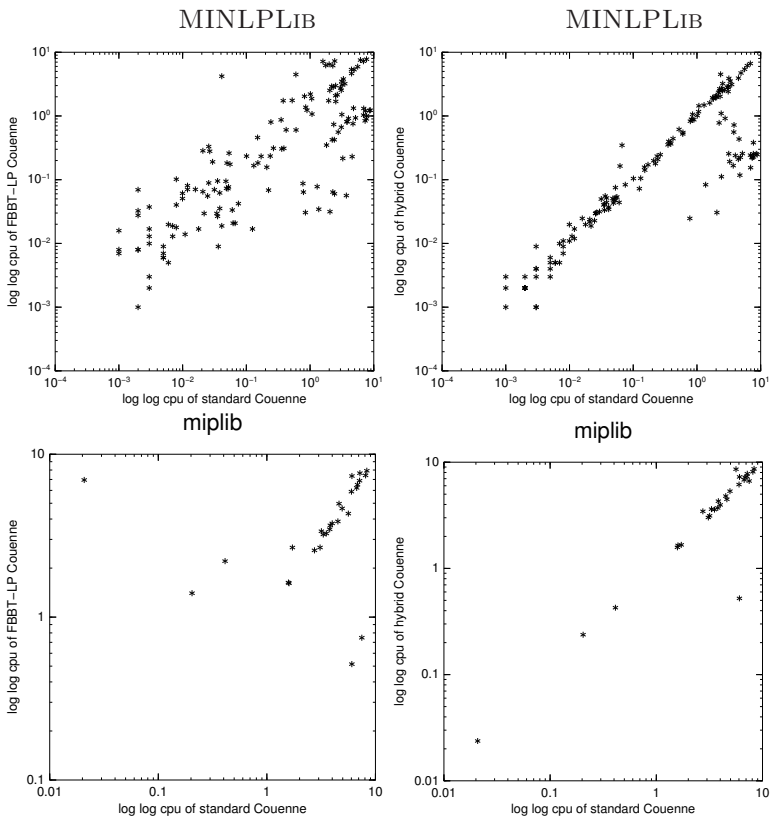


Figure 3: User CPU time scatter plots of FBBT-LPC (left) and hybrid (right) versus standard COUENNE versions on integer instances from MINLPLIB (top) and MIPLIB (bottom).

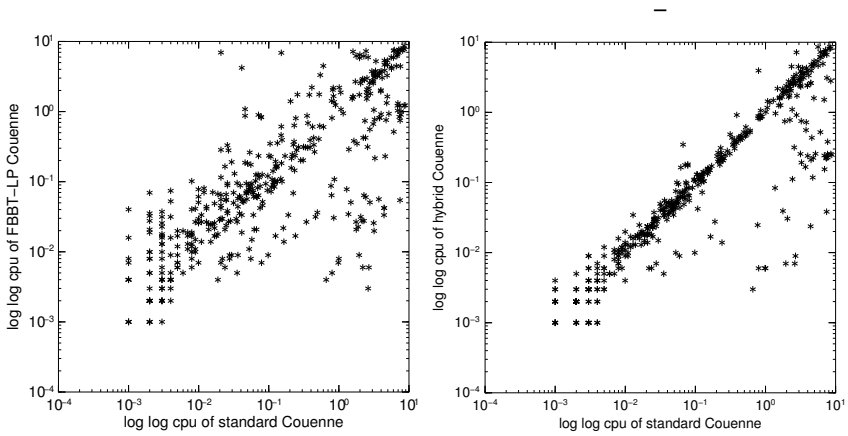


Figure 4: User CPU time scatter plots of FBBT-LPC (left) and hybrid (right) versus standard COUENNE versions on the whole test set.

6 Conclusion

Range reduction techniques have paramount importance in sBB solvers both for linear and nonlinear programming. We provide an in-depth study of the FBBT operator (over continuous variables), showing

Library	Standard $(5, 0)$	FBBT-LPC $(0, 1)$	Hybrid $(3, -5)$
GlobalLib	34	69	**78**
Pooling problems	**2**	0	**2**
MINLPLib	10	48	**54**
MIPLib	4	**12**	9
Total	50	129	**143**

Table 3: Couenne performance on discordant instances.

that its limit point can be determined in polynomial time by solving a linear program. We apply this theoretical result to the sBB algorithm by replacing FBBT with its limit-finding FBBT-LPC formulation. Lastly, we test a hybrid sBB approach where FBBT is run for a short time at each node, while the linear program for finding the limit point is run every five nodes: this approach turns out to improve performance with respect to the standard sBB. Interestingly, the FBBT-LPC makes a definite impact in MILP instances from MIPLib3 and MIPLib2003.

Acknowledgments

The last author (LL) is partly supported by grants Digiteo Chair 2009-14D "RMNCCO", Digiteo 2009-55D "ARM" and the Microsoft-CNRS "OSD" Chair.

References

[1] Adjiman, C., Dallwig, S., Floudas, C., Neumaier, A.: A global optimization method, αBB, for general twice-differentiable constrained NLPs: I. Theoretical advances. Computers & Chemical Engineering **22**(9), 1137–1158 (1998)

[2] Adjiman, C.S., Androulakis, I.P., Floudas, C.A.: A global optimization method, αBB, for general twice-differentiable constrained NLPs: II. Implementation and computational results. Computers & Chemical Engineering **22**(9), 1159–1179 (1998)

[3] Al-Khayyal, F., Sherali, H.: On finitely terminating branch-and-bound algorithms for some global optimization problems. SIAM Journal of Optimization **10**(4), 1049–1057 (2000)

[4] Andersen, D., Andersen, K.: Presolving in linear programming. Mathematical Programming **71**, 221–245 (1995)

[5] Androulakis, I.P., Maranas, C.D., Floudas, C.A.: αBB: A global optimization method for general constrained nonconvex problems. Journal of Global Optimization **7**(4), 337–363 (1995)

[6] Apt, K.: Principles of Constraint Programming. Cambridge University Press, Cambridge (2003)

[7] Belotti, P., Cafieri, S., Lee, J., Liberti, L.: Feasibility-based bounds tightening via fixed points. In: D.Z. Du, P. Pardalos, B. Thuraisingham (eds.) Combinatorial Optimization, Constraints and Applications (COCOA10), *LNCS*, vol. 6508, pp. 65–76. Springer, New York (2010)

[8] Belotti, P., Lee, J., Liberti, L., Margot, F., Wächter, A.: Branching and bounds tightening techniques for non-convex MINLP. Optimization Methods and Software **24**(4), 597–634 (2009)

[9] Benhamou, F., Goualard, F., Granvilliers, L., Puget, J.F.: Revising hull and box consistency. In: Proceedings of the 1999 international conference on Logic programming, pp. 230–244. Massachusetts Institute of Technology, Cambridge (1999)

[10] Bixby, R., Ceria, S., McZeal, C., Savelsbergh, M.: An updated mixed integer programming library: Miplib 3. Tech. Rep. TR98-03, Rice University (1998)

[11] Bordeaux, L., Hamadi, Y., Vardi, M.: An analysis of slow convergence in interval propagation. In: C. Bessière (ed.) Principles and Practice of Constraint Programming, *LNCS*, vol. 4741, pp. 790–797. Springer (2007)

[12] Bussieck, M.: Globallib — a collection of nonlinear programming problems (2004). (`http://www.gamsworld.org/global/globallib.htm`)

[13] Bussieck, M., Drud, A., Meeraus, A.: MINLPLib — A collection of test models for mixed-integer nonlinear programming. INFORMS Journal on Computing **15**(1) (2003)

[14] D'Ambrosio, C., Frangioni, A., Liberti, L., Lodi, A.: Experiments with a feasibility pump approach for nonconvex MINLPs. In: P. Festa (ed.) Symposium on Experimental Algorithms, *LNCS*, vol. 6049. Springer, Heidelberg (2010)

[15] Davey, B., Priestley, H.: Introduction to Lattices and Order, 2nd edn. Cambridge University Press, Cambridge (2002)

[16] Davis, E.: Constraint propagation with interval labels. Artificial Intelligence **32**, 281–331 (1987)

[17] Falk, J., Soland, R.: An algorithm for separable nonconvex programming problems. Management Science **15**, 550–569 (1969)

[18] Faltings, B.: Arc-consistency for continuous variables. Artificial Intelligence **65**, 363–376 (1994)

[19] Foulds, L., Haughland, D., Jornsten, K.: A bilinear approach to the pooling problem. Optimization **24**, 165–180 (1992)

[20] Freuder, E.: A sufficient condition for backtrack-free search. Journal of the ACM **29**(1), 24–32 (1982)

[21] Hooker, J.: Integrated methods for optimization. Springer, New York (2007)

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

[23] IBM: ILOG CPLEX 12.2 User's Manual. IBM (2010)

[24] Land, A., Doig, A.: An automatic method of solving discrete programming problems. Econometrica **28**(3), 497–520 (1960)

[25] Lebbah, Y., Lhomme, O.: Accelerating filtering techniques for numeric CSPs. Artificial Intelligence **139**, 109–132 (2002)

[26] Liberti, L.: Writing global optimization software. In: L. Liberti, N. Maculan (eds.) Global Optimization: from Theory to Implementation, pp. 211–262. Springer, Berlin (2006)

[27] Liberti, L.: Reformulations in mathematical programming: Definitions and systematics. RAIRO-RO **43**(1), 55–86 (2009)

[28] Liberti, L., Mladenović, N., Nannicini, G.: A good recipe for solving MINLPs. In: V. Maniezzo, T. Stützle, S. Voß (eds.) Hybridizing metaheuristics and mathematical programming, *Annals of Information Systems*, vol. 10, pp. 231–244. Springer, New York (2009)

[29] Liberti, L., Roux, S.L., Leconte, J., Marinelli, F.: Mathematical programming based debugging. In: R. Mahjoub (ed.) Proceedings of the International Symposium on Combinatorial Optimization, *Electronic Notes in Discrete Mathematics*, vol. 36, pp. 1311–1318. Elsevier, Amsterdam (2010)

[30] Martin, A., Achterberg, T., Koch, T.: Miplib 2003. Tech. Rep. 05-28, ZIB (2005)

[31] Messine, F.: Méthodes d'optimisation globale basées sur l'analyse d'intervalle pour la résolution de problèmes avec contraintes (in French). Ph.D. thesis, Institut National Polytechnique de Toulouse (1997)

[32] Messine, F.: Deterministic global optimization using interval constraint propagation techniques. RAIRO-RO **38**(4), 277–294 (2004)

[33] Moore, R.: Interval Analysis. Prentice Hall, Englewood Cliffs (1966)

[34] Moore, R.: Methods and Applications of Interval Analysis. SIAM, Philadelphia (1979)

[35] Moore, R., Kearfott, R., Cloud, M.: Introduction to Interval Analysis. SIAM, Philadelphia (2009)

[36] Roman, S.: Lattices and Ordered Sets. Springer, New York (2008)

[37] Ryoo, H., Sahinidis, N.: Global optimization of nonconvex NLPs and MINLPs with applications in process design. Computers & Chemical Engineering **19**(5), 551–566 (1995)

[38] Sahinidis, N.: Global optimization and constraint satisfaction: the Branch-and-Reduce approach. In: C. Bliek, C. Jermann, A. Neumaier (eds.) Global Optimization and Constraint Satisfaction, *LNCS*, vol. 2861, pp. 1–16. Springer (2003)

[39] Sahinidis, N., Tawarmalani, M.: BARON 7.2.5: Global Optimization of Mixed-Integer Nonlinear Programs, *User's Manual* (2005)

[40] Savelsbergh, M.: Preprocessing and probing techniques for mixed integer programming problems. INFORMS Journal on Computing **6**(4), 445–454 (1994)

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

[42] Shectman, J., Sahinidis, N.: A finite algorithm for global minimization of separable concave programs. Journal of Global Optimization **12**, 1–36 (1998)

[43] Smith, E.: On the optimal design of continuous processes. Ph.D. thesis, Imperial College of Science, Technology and Medicine, University of London (1996)

[44] Smith, E., Pantelides, C.: A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs. Computers & Chemical Engineering **23**, 457–478 (1999)

[45] Tawarmalani, M., Sahinidis, N.: Global optimization of mixed integer nonlinear programs: A theoretical and computational study. Mathematical Programming **99**, 563–591 (2004)

[46] Vu, X.H., Schichl, H., Sam-Haroud, D.: Interval propagation and search on directed acyclic graphs for numerical constraint solving. Journal of Global Optimization **45**, 499–531 (2009)

[47] Waltz, D.: Understanding the line drawings of scenes with shadows. In: P. Winston (ed.) The Psychology of Computer Vision, pp. 19–91. McGraw-Hill, New York (1975)

[48] Ziegler, G.: Lectures on Polytopes. Springer, Berlin (1995)