Feasibility-based bounds tightening via fixed points*

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

[1] Dept. of Mathematical Sciences, Clemson University, Clemson SC 29634, USA, pbelott@clemson.edu
[2] Dept. Mathématiques et Informatique, ENAC, 7 av. E. Belin, 31055 Toulouse, France, cafieri@recherche.enac.fr
[3] Dept. of Mathematical Sciences, IBM T.J. Watson Research Center, PO Box 218, Yorktown Heights, NY 10598, USA, jonlee@us.ibm.com
[4] LIX, École Polytechnique, 91128 Palaiseau, France liberti@lix.polytechnique.fr

Abstract. The search tree size of the spatial Branch-and-Bound algorithm for Mixed-Integer Nonlinear Programming depends on many factors, one of which is the width of the variable ranges at every tree node. A range reduction technique often employed is called Feasibility Based Bounds Tightening, which is known to be practically fast, and is thus deployed at every node of the search tree. From time to time, however, this technique fails to converge to its limit point in finite time, thereby slowing the whole Branch-and-Bound search considerably. In this paper we propose a polynomial time method, based on solving a linear program, for computing the limit point of the Feasibility Based Bounds Tightening algorithm applied to linear equality and inequality constraints.
Keywords: global optimization, MINLP, spatial Branch-and-Bound, range reduction, constraint programming.

1 Introduction

In this paper we discuss an important sub-step, called Feasibility Based Bounds Tightening (FBBT) of the spatial Branch-and-Bound (sBB) method for solving Mixed-Integer Nonlinear Programs (MINLP) of the form:

$$\left.\begin{array}{rr} \min & x_n \\ & g(x) \in G^0 \\ & x \in X^0 \\ \forall i \in Z & x_i \in \mathbb{Z}, \end{array}\right\} \tag{1}$$

where $x \in \mathbb{R}^n$ are decision variables, $\mathscr{I}$ is the set of all real intervals, $G^0 = [g^{0L}, g^{0U}] \in \mathscr{I}^m$ and $X^0 = [x^{0L}, x^{0U}] \in \mathscr{I}^n$ are vectors of real intervals, also

* Partially supported by grants ANR 07-JCJC-0151 "ARS", ANR 08-SEGI-023 "AsOpt", Digiteo Emergence "PASO", Digiteo Chair 2009-14D "RMNCCO", Digiteo Emergence 2009-55D "ARM", System@tic "EDONA".

called *boxes*, Z is a given subset of $\{1, \ldots, n\}$ encoding the integrality constraints on some of the variables, and $g : \mathbb{R}^n \to \mathbb{R}^m$ are continuous functions. Let $\mathscr{X}$ be the feasible region of (1). We remark that every MINLP involving a general objective function $\min f(x)$ can be reformulated exactly to the formulation (1) at the cost of adjoining the constraint $x_n \geq f(x)$ to the constraints $g(x) \in G^0$.

The sBB is a ε-approximation algorithm (with a given $\varepsilon > 0$) for problems (1) which works by generating a sequence of upper bounds x'_n and lower bounds $\bar{x}_n$ to the optimal objective function value x^*_n. The upper bounding solution x' is found by solving (1) locally with MINLP heuristics [1, 2], whilst the lower bound is computed by automatically constructing and solving a convex relaxation of (1). If $x'_n - \bar{x}_n \leq \varepsilon$ then x' is feasible and at most ε-suboptimal; if x' is the best optimum so far, it is stored as the *incumbent*. Otherwise X^0 is partitioned into two boxes X', X'' along a direction x_i at a branch point p_i, and the algorithm is called recursively on (1) with X^0 replaced by each of the two boxes X', X'' in turn. This generates a binary search tree: nodes can be pruned if the convex relaxation is infeasible or if $\bar{x}_n$ is greater than the objective function value at the incumbent. The sBB converges if the lower bound is guaranteed to increase strictly whenever the box X of ranges of x at the current node decreases strictly. In general, the sBB might fail to converge in finite time if $\varepsilon = 0$, although some exceptions exist [3, 4].

An important step of the sBB algorithm is the reduction of the variable ranges X at each node. There are two commonly used range reduction techniques in Global Optimization (GO): Optimization Based Bounds Tightening (OBBT) and FBBT. The former is slower and more effective, involves the solution of $2n$ Linear Programs (LP) and is used either rarely or just at the root node of the sBB search tree [18]. The latter is an iterative procedure based on propagating the effect of the constraints $g(x) \in G^0$ on the variable ranges using interval arithmetic; FBBT is known to be practically efficient and is normally used at each sBB node. Practical efficiency notwithstanding, the FBBT sometimes converges to its limit point in infinite time, as the example of Eq. (3.11) in [5] shows. The same example also shows that an artificial termination condition enforced when the range reduction extent becomes smaller than a given tolerance might yield arbitrarily large execution times.

The FBBT was borrowed from Artificial Intelligence (AI) and Constraint Programming (CP) as a bounds filtering technique. Its origins can be traced to [6]; it is known *not* to achieve bound consistency [7] (apart from some special cases [8]), a desirable property for Constraint Satisfaction Problems (CSP): a CSP is bound consistent if every projection of its feasible region on each range X_i is X_i itself [5]. The FBBT was employed as a range reduction technique for Mixed Integer Linear Programs (MILP) in [9, 10] and then for MINLPs in [11]. Within the context of GO, the FBBT was discussed in [12] and recently improved in [13] by considering its effect on common subexpressions of $g(x)$.

The main result of this paper is to show that if $g(x)$ are linear forms, then there exists an LP whose solution is exactly the limit point of the FBBT, which is therefore shown to be computable in polynomial time. If $g(x)$ are nonlinear

functions (as is commonly the case for general MINLPs) we can replace them either by a linear relaxation $\bar{g}(x) \in \bar{G}^0$ of $\mathscr{X}$ or simply consider the largest linear subset $\hat{g}(x) \in \hat{G}^0$ of the constraints $g(x) \in G^0$, according to the usual trade-off between computational effort and result quality (we follow the latter approach in our computational results section).

The rest of this paper is organized as follows. In Sect. 2 we define an FBBT iteration formally as an operator in the interval lattice and show it has a fixed point. In Sect. 3 we show how to construct a linear relaxation of (1). In Sect. 4 we describe an LP the solution of which is the limit point of the FBBT algorithm. In Sect. 5 we discuss computational results on the MINLPLib showing the potential of the LP-based technique.

2 The FBBT algorithm

The FBBT algorithm works by propagating the variable range vector $X^0 \in \mathscr{I}^n$ to the operators in $g(x)$ (using interval arithmetic) in order to derive the interval vector $G \in \mathscr{I}^m$ consisting of lower and upper bounds on $g(x)$ when $x \in X^0$; the vector $G \cap G^0$ is then propagated back to a variable range vector X using inverse interval arithmetic. The interval vector $X \cap X^0$ therefore contains valid and potentially tighter variable ranges for x. If $X \cap X^0 \subsetneq X^0$, then the procedure can be repeated with X^0 replaced by $X \cap X^0$ until no more change occurs. The FBBT algorithm can be shown to converge to a fixed point (fp), see Sect. 2.3. As mentioned in the introduction, the FBBT might fail to converge to its fp in finite time. With a slight abuse of terminology justified by Thm. 2.2 we shall refer to "nonconvergence" to mean "convergence to the fp in infinite time".

2.1 Expression graphs

We make the assumption that the functions g appearing in (1) are represented by expressions built recursively as follows:

1. any element of $\{x_1, \ldots, x_n\} \cup \mathbb{R}$ is an expression (such primitive expressions are called *atoms*);
2. if e_1, e_2 are expressions, then $e = e_1 \otimes e_2$ is an expression for all operators $\otimes$ in a given set $\mathscr{O}$; e_1, e_2 are said to be *subexpressions* of e.

Let $\mathcal{E}$ be the set of all expressions built by the repeated applications of the two above rules, and for all $e \in \mathcal{E}$ let $\mathsf{function}(e)$ be the function $f : \mathbb{R}^n \to \mathbb{R}$ which e represents (this correspondence can be made precise using an evaluation function for expressions [19], p. 244); conversely, to all functions in $f \in \mathsf{function}(\mathcal{E})$ we let $\mathsf{expression}(f)$ be the expression e representing f. Each expression $e \in \mathcal{E}$ can also be associated to its recursion directed acyclic graph (DAG) $\mathsf{dag}(e)$ whose root node is e and whose other nodes are subexpressions of e, subexpressions of subexpressions of e and so on. An arc (u, v) in $\mathsf{dag}(e)$ implies a subexpression relation where v is a subexpression of u; with a slight abuse of notation we identify non-leaf nodes of $\mathsf{dag}(e)$ with operators (labelled $\otimes_v$ for an operator

$\otimes \in \mathscr{O}$ at a node v) and leaf nodes with atoms (labelled either x_i if the node is a variable, or the real constant that the node represents). For all nodes u of $\mathsf{dag}(e)$ we indicate by $\delta^+(u)$ the outgoing star of the node u, i.e. all vertices v such that (u, v) is an arc in $\mathsf{dag}(e)$.

We assume that $\mathscr{O}$ contains: the infix n-ary sum $+$, the infix binary difference $-$, left multiplication operators $a\times$ for each nonzero real constant $a \in \mathbb{R} \smallsetminus \{0\}$, the infix n-ary product $\times$, the infix binary division $\div$, right raising operators $(\cdot)^q$ for each rational constant $q \in \mathbb{Q}$, the left unary exponential operator exp, its inverse log and the left unary trigonometric operators sin, cos, tan. We also let $\mathscr{O}' = \{+, -, a\times\}$ be the set of linear operators. Endowed with the operator set $\mathscr{O}$, the MINLP formulation (1) can express all practically interesting MP problems from Linear Programming (LP), where $\mathscr{O}$ is replaced by $\mathscr{O}'$ and $Z = \emptyset$, Mixed-Integer Linear Programming (LP), where $\mathscr{O}$ is replaced by $\mathscr{O}'$ and $Z \neq \emptyset$, Nonlinear Programming (NLP), where $Z = \emptyset$, to MINLPs, where $Z \neq \emptyset$. By letting atoms range over matrices, formulation (1) can also encode Semidefinite Programming (SDP) [14]. Black-box optimization problems, however, cannot be described by formulation (1).

The expression representation for functions is well known in computer science [15], engineering [16, 17] and GO [18–20] where it is used for two substeps of the sBB algorithm: lower bound computation and FBBT. More formal constructions of $\mathcal{E}$ can be found in [21], Sect. 3 and [22], Sect. 2.

2.2 The problem DAG

We can also associate a DAG to the whole problem (1) to describe its symbolic nonlinear structure:

$$D' = \bigcup_{i \leq m} \mathsf{dag}(\mathsf{expression}(g_i)) \tag{2}$$

$$D = \mathsf{contract}(D', \{x_1, \ldots, x_n\}), \tag{3}$$

where $\mathsf{contract}(G, L)$ is the result of contracting all vertices of G labelled by ℓ for all $\ell \in L$, i.e. of replacing each subgraph of G induced by all vertices labelled by $\ell \in L$ by a single node labelled by ℓ. The difference between D' and D is that in D' some variable nodes are repeated; more precisely, if x_j occurs in both g_i and g_h, x_j will appear as a leaf node in both $\mathsf{dag}(\mathsf{expression}(g_i))$ and $\mathsf{dag}(\mathsf{expression}(g_h))$.

For all v in D (resp. D') we let $\mathsf{index}(v)$ be i if v is the root node for g_i, j if v is the node for x_j, and 0 otherwise. If v is a constant atom, we let c_v be the value of the constant.

2.3 Formal definition of the FBBT operator

An iteration of the FBBT consists of an upward phase, propagating variable ranges X to an interval vector G such that $g(x) \in G$, and of a downward phase, propagating $G \cap G^0$ down again to updated variable ranges X. Propagation

occurs along the arcs of D. To each node v of D we associate an interval Y_v; initially we set $Y = Y^0$ where for all v representing variable nodes we have $Y_v^0 = X^0_{\mathsf{index}(v)}$, for root nodes representing constraints we have $Y_v^0 = G^0_{\mathsf{index}(v)}$, and $Y_v^0 = [-\infty, \infty]$ otherwise. Operators $\otimes_v$ act on intervals by means of interval arithmetic [23]. The upward propagation occurs along the arcs in the opposite direction, as shown in Alg. 1.

Algorithm 1 $\mathsf{up}(v, Y)$

Require: v (node of D), Y
Ensure: Y
1: **if** v is not an atom **then**
2: **for all** $u \in \delta^+(v)$ **do**
3: $Y = \mathsf{up}(u, Y)$
4: **end for**
5: Let $Y_v = Y_v \cap \otimes_v(Y_u \mid u \in \delta^+(v))$
6: **else if** v is a constant atom **then**
7: $Y_v = [c_v, c_v]$
8: **end if**
9: **return** Y

The downward propagation is somewhat more involved and follows the arcs of D' in the natural direction. For each non-root node v in D' we define $\mathsf{parent}(v)$ as the unique node u such that (u, v) is an arc in D', the set $\mathsf{siblings}(v) = \delta^+(v) \setminus \{v\}$ and the set $\mathsf{family}(v) = \{\mathsf{parent}(v)\} \cup \mathsf{siblings}(v)$, i.e. the siblings and the parent of v. Let $z = \mathsf{parent}(v)$; then $\otimes_z$ induces a function $\mathbb{R}^{|\delta^+(z)|} \to \mathbb{R}$ such that:

$$w_z = \otimes_z(w_u \mid u \in \delta^+(z)). \tag{4}$$

We now define the operator $\otimes_v^{-1}$ as an "inverse" of $\otimes_z$ in the v coordinate. If there is an expression $e \in \mathcal{E}$ such that $w_v = \mathsf{function}(e)(w_u \mid u \in \mathsf{family}(v))$ if and only if (4) holds, let $\omega_v^{-1} = \mathsf{function}(e)$; otherwise, let $\otimes_v^{-1}$ map every argument tuple to the constant interval $[-\infty, \infty]$. The downward propagation, shown in Alg. 2, is based on applying $\otimes_v^{-1}$ recursively to D'.

We remark that down is defined on D' rather than D for a technical reason, i.e. family relies on nodes having a *unique* parent, which is the case for D' since it is the union of several DAGs; however, leaf nodes of D' representing the same variable $j \leq n$ are contracted to a single node in D, which therefore loses the parent uniqueness property at the leaf node level. Notwithstanding, Y is indexed on nodes of D rather than D', so that in Line 3 (Alg. 2), when u is a variable node we have $Y_u = X_{\mathsf{index}(u)}$, so that the update on Y_u is carried out on the interval referring to the same variable independently of which DAG was used in the calling sequence.

If p is the number of vertices in D, up and down are operators $\mathscr{I}^p \to \mathscr{I}^p$. Since the up operator only changes the intervals in Y relating to a single expres-

Algorithm 2 $\mathsf{down}(v, Y)$

Require: v (node of D'), Y
Ensure: Y
1: **if** v is not an atom **then**
2: **for all** $u \in \delta^+(v)$ s.t. u is not a constant atom **do**
3: $Y_u = Y_u \cap \otimes_u^{-1}(\mathsf{family}(u))$
4: **end for**
5: **for all** $u \in \delta^+(v)$ **do**
6: $Y = \mathsf{down}(u, Y)$
7: **end for**
8: **end if**
9: **return** Y

sion DAG, we extend its action to the whole of D:

$$\mathcal{U}(Y) = \bigcap_{i \leq m} \mathsf{up}(\bar{g}_i, Y), \tag{5}$$

where $\bar{g}_i$ denotes the root node of $\mathsf{dag}(\mathsf{expression}(g_i))$. Similarly, since down needs to be applied to each root node of D', we define:

$$\mathcal{D}(Y) = \bigcap_{i \leq m} \mathsf{down}(\bar{g}_i, Y). \tag{6}$$

Finally, we define the FBBT operator:

$$\mathcal{F}(Y) = \mathcal{D}(\mathcal{U}(Y \cap Y^0) \cap Y^0). \tag{7}$$

Lemma 2.1. *The operators $\mathcal{U}$, $\mathcal{D}$, $\mathcal{F}$ are monotone and inflationary in the interval lattice $\mathscr{I}^p$ ordered by reverse inclusion $\supseteq$.*

Proof. Monotonicity follows because all the interval arithmetic operators in $\mathscr{O}$ are monotone [23], the composition of monotone operators is monotone, the functions represented by expressions in $\mathcal{E}$ are compositions of operators in $\mathscr{O}$. Inflationarity follows because of the intersection operators in Lines 5 of Alg. 1 and 3 of Alg. 2 and those in (5)-(7). □

Theorem 2.2. *The operator $\mathcal{F}$ has a unique least fixed point.*

Proof. This follows by Lemma 2.1 and Thm. 12.9 in [24]. □

We remark that since we ordered $\mathscr{I}^p$ by reverse inclusion, the least fixed point (lfp) of Thm. 2.2 is actually the greatest fixed point (gfp) with respect to standard interval inclusion.

3 Linear relaxation of the MINLP

Different implementations of the sBB algorithm construct the lower bound $\bar{x}_n$ to the optimal objective function value x_n^* in different ways. Some are based on the factorability of the functions in $g(x)$ [25–27], whilst others are based on a symbolic reformulation of (1) based on the problem DAG D [18–20]. We employ the latter approach: each non-leaf node z in the vertex set of D is replaced by an added variable w_z and a corresponding constraint (4) is adjoined to to the formulation (usually, two variables w_v, w_u corresponding to identical defining constraints are replaced by one single added variable). The resulting reformulation, sometimes called *Smith standard form* [18], is exact [14]. A linear relaxation of (1) can automatically be obtained by the Smith standard form by replacing each nonlinear defining constraints by lower and upper linear approximations. In order for the sBB convergence property to hold, the coefficients of these linear approximations are functions of the variable ranges $X = [x^L, x^U]$, so that as the width of X_i decreases for some $i \leq n$, the optimal objective function value of the linear relaxation increases.

4 FBBT in the linear case

In this section we assume that the nonlinear part $g(x) \in G^0$ of (1) is either replaced by its linear relaxation $\bar{g}(x) \in \bar{G}^0$ as discussed in Sect. 3 or by the largest subset of linear constraints $\hat{g}(x) \in \hat{G}^0$ in (1), called the *linear part* of (1). If $\bar{\mathscr{X}} = \{x \in \mathbb{R}^n \mid \bar{g}(x) \in \bar{G}^0\}$ and $\hat{\mathscr{X}} = \{x \in \mathbb{R}^n \mid \hat{g}(x) \in \hat{G}^0\}$ are the feasible regions of the relaxation and of the linear part of (1), then $\bar{\mathscr{X}} \subseteq \hat{\mathscr{X}}$. Both $\bar{\mathscr{X}}$ and $\hat{\mathscr{X}}$ can be represented by systems of linear equations and inequalities:

$$Ax \in B^0, \tag{8}$$

where $A = (a_{ij})$ is a $m \times n$ real matrix. A system (8) encoding $\bar{\mathscr{X}}$ is a lifting in the w added variables and has in general many more constraints than a system (8) encoding $\hat{\mathscr{X}}$. Performing the FBBT on the relaxation will generally yield tighter ranges than on the linear part. For the purposes of this section, the construction of the LP yielding the gfp will be exactly the same in both cases.

The constraints in (8) are of the form $b^{0L} \leq a_i \cdot x \leq b^{0U}$, where $B^0 = [b^{0L}, b^{0U}]$ and a_i is the i-th row of A. Letting $J_i = \{j_1, \ldots, j_{k_i}\}$ be such that $a_{ij} \neq 0$ for all $j \in J_i$ and $a_{ij} = 0$ otherwise, we write $g_i(x) = \sum_{j \in J_i} a_{ij} x_j$, $e = \mathsf{expression}(g_i) = a_{ij_1} \times x_{j_1} + \cdots + a_{ij_{k_i}} \times x_{j_{k_i}}$ and hence $\mathsf{dag}(e)$ is the DAG shown in Fig. 1.

Because of their simple structure, the intervals Y_v where v represents operators such as $a_{ij}\times$ can be disposed of, and closed form interval expressions for $\mathcal{U}$, $\mathcal{D}$ can be derived that act on an interval vector (X, B) where $X \in \mathscr{I}^n$ and $B \in \mathscr{I}^m$. Specifically, $\mathcal{U}$ becomes:

$$\mathcal{U}(X, B) = (X, (B_i \cap \sum_{j \in J_i} a_{ij} X_j \mid i \leq m)) \tag{9}$$

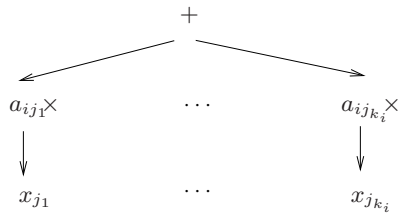

Fig. 1. The expression DAG of a row of (8).

and $\mathcal{D}$ becomes:

$$\mathsf{down}(X, B) = ((X_j \cap \bigcap_{\substack{i \leq m \\ a_{ij} \neq 0}} \frac{1}{a_{ij}}(B_i - \sum_{\ell \neq j} a_{i\ell} X_\ell) \mid j \leq n), B), \tag{10}$$

where the products and the sums are interval operations [23]. The equivalence of (9) with (5) in the linear case follows by simply replacing $a_{ij}X_j$ with an interval Y_v for a node v in D representing the operator $a_{ij}\times$, and similarly for the equivalence of (10) with (6), where we remark that the interval inverse $\otimes_v^{-1}$ of a linear form in each component is simply another linear form. We consequently re-define $\mathcal{F}$ for the linear case as follows:

$$\mathcal{F}(X, B) = \mathcal{D}(\mathcal{U}(X \cap X^0, B \cap B^0)). \tag{11}$$

4.1 Greatest fixed point via Linear Programming

The gfp of $\mathcal{F}$ with respect to interval inclusion is, by definition,

$$\text{gfp}(\mathcal{F}) = \sup_{\subseteq}\{(X, B) \in \mathscr{I}^{n+m} \mid (X, B) = \mathcal{F}(X, B)\}. \tag{12}$$

Consider the interval vector width sum function $|\cdot| : \mathscr{I}^n \to \mathbb{R}_+$ given by $|X| = \sum_{j \leq n}(x_j^U - x_j^L)$. It is easy to see that it is monotonic with the lattice order $\subseteq$ on $\mathscr{I}^n$, in the sense that if $X \subseteq X'$ then $|X| \leq |X'|$ (the converse may not hold). Furthermore, for $(X, B) \in \mathscr{I}^{n+m}$ we have $|(X, B)| = |X| + |B|$.

4.1 Proposition
$|gfp(\mathcal{F})| \geq |X| + |B|$ for all fixed points (X, B) of $\mathcal{F}$.

Proof. Since $(X^*, B^*) = \text{gfp}(\mathcal{F})$ is the inclusion-wise greatest of all fixed points of $\mathcal{F}$, it is also maximal with respect to all the other fixed points that are included in it. Suppose, to get a contradiction, that there is a fixed point (X', B') of $\mathcal{F}$ with $|X'| + |B'| > |X^*| + |B^*|$; by assumption, (X', B') is not included in (X^*, B^*). Since the set of fixed points of $\mathcal{F}$ is a complete lattice by Tarski's Fixed Point theorem [28], there must be a fixed point (X'', B'') of $\mathcal{F}$ which includes both (X', B') and (X^*, B^*). By monotonicity of $|\cdot|$, $|X'| + |B'| \leq |X''| + |B''|$, and hence $|X^*| + |B^*| < |X''| + |B''|$, showing that $(X'', B'') \supsetneq (X^*, B^*)$ and hence that (X^*, B^*) is not the gfp of $\mathcal{F}$, against the hypothesis. □

4.2 Theorem
The following interval programming problem:

$$\left.\begin{array}{r} \max |X| + |B| \\ (X,B) \subseteq (X^0, B^0) \\ (X,B) \subseteq \mathcal{U}(X,B) \\ (X,B) \subseteq \mathcal{D}(X,B) \end{array}\right\} \tag{13}$$

has a unique global optimum equal to gfp($\mathcal{F}$).

Proof. By Tarski's Fixed Point Theorem [28], the gfp of $\mathcal{F}$ is the join of all its pre-fixed points, so we can replace $=$ with $\subseteq$ in (12). The equivalence of (13) with (12) then follows by Prop. 4.1 and (11). Uniqueness of solution follows by uniqueness of the gfp. □

We remark that (13) can be reformulated as an ordinary LP by simply replacing each interval $X_j = [x_j^L, x_j^U]$ by pairs of decision variables (x_j^L, x_j^U) with the constraint $x_j^L \leq x_j^U$ (for all $j \leq n$ and similarly for $B = [b^L, b^U]$). The constraints for $\mathcal{U}$ can be trivially reformulated from interval to scalar arithmetic; the equivalent reformulation for constraints in $\mathcal{D}$ is a little more involved but very well known [18, 19, 5]. The LP encoding the gfp of the FBBT is given in (14)-(34). The z variables have been added for clarity — they simply replace products $a_{ij}x_j$ appropriately, depending on the sign of a_{ij}. Let $S^+ = \{i \leq m, j \leq n \mid a_{ij} > 0\}$ and $S^-\{i \leq m, j \leq n \mid a_{ij} < 0\}$.

$$\max_{x,b,z} \sum_{j \leq n} (x_j^U - x_j^L) + \sum_{i \leq m} (b_i^U - b_i^L) \tag{14}$$
$$\forall j \leq n \quad x_j^L \geq x_j^{0L} \tag{15}$$
$$\forall j \leq n \quad x_j^U \leq x_j^{0U} \tag{16}$$
$$\forall i \leq m \quad b_j^L \geq b_i^{0L} \tag{17}$$
$$\forall i \leq m \quad b_j^U \leq b_i^{0U} \tag{18}$$
$$\forall (i,j) \in S^+ \quad z_{ij}^L = a_{ij} x_j^L \tag{19}$$
$$\forall (i,j) \in S^+ \quad z_{ij}^U = a_{ij} x_j^U \tag{20}$$
$$\forall (i,j) \in S^- \quad z_{ij}^L = a_{ij} x_j^U \tag{21}$$
$$\forall (i,j) \in S^- \quad z_{ij}^U = a_{ij} x_j^L \tag{22}$$
$$\forall i \leq m \quad b_i^L \geq \sum_{j \leq n} z_{ij}^L \tag{23}$$
$$\forall i \leq m \quad b_i^U \leq \sum_{j \leq n} z_{ij}^U \tag{24}$$
$$\forall (i,j) \in S^+ \quad x_j^L \geq \frac{1}{a_{ij}} (b_i^L - \sum_{\ell \neq j} z_{i\ell}^U) \tag{25}$$
$$\forall (i,j) \in S^+ \quad x_j^U \leq \frac{1}{a_{ij}} (b_i^U - \sum_{\ell \neq j} z_{i\ell}^L) \tag{26}$$
$$\forall (i,j) \in S^- \quad x_j^L \geq \frac{1}{a_{ij}} (b_i^U - \sum_{\ell \neq j} z_{i\ell}^L) \tag{27}$$
$$\forall (i,j) \in S^+ \quad x_j^U \leq \frac{1}{a_{ij}} (b_i^L - \sum_{\ell \neq j} z_{i\ell}^U) \tag{28}$$
$$\forall j \leq n \quad x_j^L \leq x_j^U \tag{29}$$
$$\forall i \leq m \quad b_i^L \leq b_i^U \tag{30}$$
$$\forall i \leq m, j \leq n \quad z_{ij}^L \leq z_{ij}^U \tag{31}$$
$$x^L, x^U \in \mathbb{R}^n \tag{32}$$
$$b^L, b^U \in \mathbb{R}^m \tag{33}$$
$$z^L, z^U \in \mathbb{R}^{mn}. \tag{34}$$

Constraints (15)-(18) encode $(X,B) \subseteq (X^0, B^)$; constraints (19)-(24) encode $(X,B) \subseteq \mathcal{U}(X,B)$; constraints (25)-(28) encode $(X,B) \subseteq \mathcal{D}(X,B)$; and constraints (29)-(31) encode the fact that the decision variables x^L, x^U, b^L, b^U (as well as the auxiliary variables z^L, z^U) represent intervals.

If we formalize the problem of finding the gfp of the FBBT operator as a decision problem (for example deciding if the gfp width sum is smaller than

the original bounds X^0 by at least a given constant $\gamma > 0$), then Thm. 4.2 shows that this problem is in **P**. It is interesting to remark that [29] proves that essentially the same problem (with a few more requirements on the type of allowed constraints in $Ax \in G^0$) is **NP**-complete as long as variable integrality constraints are enforced.

5 Computational results

Our testbed consists of the MINLPLib [30] instance library. We first artificially restricted each instance to $X \in \{-10^4, 10^4\}$ in order for $|\cdot|$ to be bounded. Secondly, we ran the FBBT on the linear constraints $\hat{g}(x) \leq \hat{G}^0$, with a termination condition set at $|X^k \triangle X^{k-1}| \leq 10^{-6}$ on slow progress at iteration k. We then computed the fixed point by solving the LP (14)-(34) applied to $\hat{g}(x) \leq \hat{G}^0$. The FBBT, as well as the automatic construction of the LP (14)-(34), were implemented in ROSE [31, 32]. LPs were solved using CPLEX 11 [33]. All results were obtained on one core of an Intel Core 2 Duo at 1.4GHz with 3GB RAM running Linux.

Out of 194 MINLPs in the MINLPLib we obtained results for 172 (the remaining ones failing on some AMPL [34] error). For each successful instance we computed the width sum $|X|$ of the obtained solution and the user CPU time taken by the traditional FBBT method and by the LP based one. As these methods would be typically used in a sBB algorithm, we ignored the LP construction time, since this would be performed just once at the root node and then simply updated with the current node interval bounds X^0. The full table can be accessed at `http://www.lix.polytechnique.fr/~liberti/fbbtlp_table-1007.csv`. Table 1 only reports the totals, averages and standard deviations of the sample.

	FBBT		LP-based	
Statistic	$\|X\|$	CPU	$\|X\|$	CPU
Total	9.38×10^7	394.37	9.16×10^7	9.07
Average	5.4×10^5	2.29	5.3×10^5	0.05
Std. dev.	1.365×10^6	17.18	1.362×10^6	0.18

Table 1. Totals, averages, standard deviations of the width sum and CPU times taken by FBBT and LP on 172 MINLPLib instances.

Table 1 is consistent with what was empirically observed about the FBBT: it often works well but it occasionally takes a long time converging to the fixed point. The LP-based method addresses this weakness perfectly, as shown by the markedly better CPU time statistics. Since the LP finds the guaranteed gfp, it also produces somewhat tighter interval bounds, although the savings in terms of $|X|$ are not spectacular. The traditional FBBT was strictly faster than the LP-based method in 43% of the instances. The bulk of the CPU time savings of the LP-based method is due to twelve FBBT CPU time outliers taking $> 1s$ (the results are shown in Table 2). For nine of these the large CPU time is actually due to problem size, i.e. the number of FBBT iterations is low (< 5). Three of the instances displayed clear signs of nonconvergence.

Name	FBBT CPU time	FBBT iterations
cecil_13	27.6	29
nuclear14b	16.25	2
nuclear14	5.96	1
nuclear24b	16.19	2
nuclear24	5.89	1
nuclear25b	18.97	2
nuclear25	6.78	1
product	11.21	4
risk2b	2.28	2
space960	218.41	2
super3t	39.96	15
util	12.78	321

Table 2. FBBT CPU time outlier instances.

References

1. D'Ambrosio, C., Frangioni, A., Liberti, L., Lodi, A.: Experiments with a feasibility pump approach for nonconvex MINLPs. In Festa, P., ed.: Symposium on Experimental Algorithms. Volume 6049 of LNCS., Heidelberg, Springer (2010)
2. Liberti, L., Mladenović, N., Nannicini, G.: A good recipe for solving MINLPs. In Maniezzo, V., Stützle, T., Voß, S., eds.: Hybridizing metaheuristics and mathematical programming. Volume 10 of Annals of Information Systems., New York, Springer (2009) 231–244
3. Al-Khayyal, F., Sherali, H.: On finitely terminating branch-and-bound algorithms for some global optimization problems. SIAM Journal of Optimization **10**(4) (2000) 1049–1057
4. Bruglieri, M., Liberti, L.: Optimal running and planning of a biomass-based energy production process. Energy Policy **36** (2008) 2430–2438
5. Hooker, J.: Integrated methods for optimization. Springer, New York (2007)
6. Waltz, D.: Understanding the line drawings of scenes with shadows. In Winston, P., ed.: The Psychology of Computer Vision. McGraw-Hill, New York (1975) 19–91
7. Davis, E.: Constraint propagation with interval labels. Artificial Intelligence **32** (1987) 281–331
8. Faltings, B.: Arc-consistency for continuous variables. Artificial Intelligence **65** (1994) 363–376
9. Savelsbergh, M.: Preprocessing and probing techniques for mixed integer programming problems. INFORMS Journal on Computing **6**(4) (1994) 445–454
10. Andersen, D., Andersen, K.: Presolving in linear programming. Mathematical Programming **71** (1995) 221–245
11. Shectman, J., Sahinidis, N.: A finite algorithm for global minimization of separable concave programs. Journal of Global Optimization **12** (1998) 1–36
12. Schichl, H., Neumaier, A.: Interval analysis on directed acyclic graphs for global optimization. Journal of Global Optimization **33**(4) (2005) 541–562
13. 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** (2009) 499–531
14. Liberti, L.: Reformulations in mathematical programming: Definitions and systematics. RAIRO-RO **43**(1) (2009) 55–86
15. Knuth, D.: The Art of Computer Programming, Part II: Seminumerical Algorithms. Addison-Wesley, Reading, MA (1981)

16. Bogle, I., Pantelides, C.: Sparse nonlinear systems in chemical process simulation. In Osiadacz, A., ed.: Simulation and Optimization of Large Systems, Oxford, Clarendon Press (1988)
17. Keeping, B., Pantelides, C.: Novel methods for the efficient evaluation of stored mathematical expressions on scalar and vector computers. AIChE Annual Meeting, Paper #204b (nov 1997)
18. Smith, E., Pantelides, C.: A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs. Computers & Chemical Engineering **23** (1999) 457–478
19. Liberti, L.: Writing global optimization software. In Liberti, L., Maculan, N., eds.: Global Optimization: from Theory to Implementation. Springer, Berlin (2006) 211–262
20. 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) (2009) 597–634
21. Cafieri, S., Lee, J., Liberti, L.: On convex relaxations of quadrilinear terms. Journal of Global Optimization **47** (2010) 661–685
22. Costa, A., Hansen, P., Liberti, L.: Formulation symmetries in circle packing. In Mahjoub, R., ed.: Proceedings of the International Symposium on Combinatorial Optimization. Electronic Notes in Discrete Mathematics, Amsterdam, Elsevier (accepted)
23. Moore, R., Kearfott, R., Cloud, M.: Introduction to Interval Analysis. SIAM, Philadelphia (2009)
24. Roman, S.: Lattices and Ordered Sets. Springer, New York (2008)
25. McCormick, G.: Computability of global solutions to factorable nonconvex programs: Part I — Convex underestimating problems. Mathematical Programming **10** (1976) 146–175
26. 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) (1998) 1137–1158
27. Tawarmalani, M., Sahinidis, N.: Global optimization of mixed integer nonlinear programs: A theoretical and computational study. Mathematical Programming **99** (2004) 563–591
28. Tarski, A.: A lattice-theoretical fixpoint theorem and its applications. Pacific Journal of Mathematics **5**(2) (1955) 285–309
29. Bordeaux, L., Hamadi, Y., Vardi, M.: An analysis of slow convergence in interval propagation. In Bessiere, C., ed.: Principles and Practice of Constraint Programming. Volume 4741 of Lecture Notes in Computer Science., Springer (2007) 790–797
30. Bussieck, M., Drud, A., Meeraus, A.: MINLPLib — A collection of test models for mixed-integer nonlinear programming. INFORMS Journal on Computing **15**(1) (2003)
31. Liberti, L., Cafieri, S., Savourey, D.: Reformulation optimization software engine. In: Mathematical Software. LNCS, New York, Springer (to appear)
32. Liberti, L., Cafieri, S., Tarissan, F.: Reformulations in mathematical programming: A computational approach. In Abraham, A., Hassanien, A.E., Siarry, P., Engelbrecht, A., eds.: Foundations of Computational Intelligence Vol. 3. Number 203 in Studies in Computational Intelligence. Springer, Berlin (2009) 153–234
33. ILOG: ILOG CPLEX 11.0 User's Manual. ILOG S.A., Gentilly, France. (2008)
34. Fourer, R., Gay, D.: The AMPL Book. Duxbury Press, Pacific Grove (2002)