Mathematical Programming manuscript No.
(will be inserted by the editor)

Reformulations in mathematical programming: Automatic symmetry detection and exploitation

Leo Liberti

March 7, 2010

Abstract If a mathematical program has many symmetric optima, solving it via Branch-and-Bound techniques often yields search trees of disproportionate sizes; thus, finding and exploiting symmetries is an important task. We propose a method for automatically finding the formulation group of any given Mixed-Integer Nonlinear Program, and for reformulating the problem by means of static symmetry breaking constraints. The reformulated problem — which is likely to have fewer symmetric optima — can then be solved via standard Branch-and-Bound codes such as CPLEX (for linear programs) and COUENNE (for nonlinear programs). Our computational results include formulation group tables for the MIPLib3, MIPLib2003, GlobalLib and MINLPLib instance libraries and solution tables for some instances in the aforementioned libraries.

Keywords group · symmetry · mixed integer nonlinear programming · branch and bound

1 Introduction

We consider Mixed-Integer Nonlinear Programs (MINLPs) in the following general form:

$$\left.\begin{array}{rrcl} \min\limits_{x\in\mathbb{R}^n} & f(x) & & \\ & g(x) & \le & b \\ & x & \in & [x^L,x^U] \\ \forall i\in Z & x_i & \in & \mathbb{Z}, \end{array}\right\} \tag{1}$$

where $f:\mathbb{R}^n\to\mathbb{R}$, $g:\mathbb{R}^n\to\mathbb{R}^m$, $b\in\mathbb{R}^m$, $x^L,x^U\in\mathbb{R}^n$ and $Z\subseteq\{1,\dots,n\}$. Throughout the paper, elements of groups are represented by means of permutations of either the column or the row space; permutations on the row space are denoted by left multiplication, and permutations on the column space by right multiplication. For a mathematical program P we let $\mathscr{F}(P)$ be its feasible region and $\mathscr{G}(P)$ be the set of its global optima. For $x\in\mathbb{R}^n$ and $B\subseteq\{1,\dots,n\}$, we let $x[B]=(x_j\mid j\in B)$ be the partial vector of x restricted to the components in B. If $X\subseteq\mathbb{R}^n$, then $X[B]=\{x[B]\in\mathbb{R}^{|B|}\mid x\in X\}$.

Leo Liberti
LIX, École Polytechnique, 91128 Palaiseau, France
E-mail: `liberti@lix.polytechnique.fr`

Problems (1), be they linear or nonlinear, may be solved either heuristically or exactly. The most widely used technique for solving (1) exactly is the Branch-and-Bound (BB) algorithm. BB is a tree-based search in the variable space where each node represents a subproblem of (1) whose feasible region is a subset of the feasible region of (1). A node is *pruned* when one of the following holds: (a) a global optimum for the node was found; (b) the node was proved to be infeasible; (c) a lower bound for the problem at the node has higher value than the value of the objective function evaluated at the current best solution (the *incumbent*). In all other cases, the node is *branched* into two or more subnodes the union of whose feasible regions is the same as the feasible region of the parent node. For Mixed-Integer Linear Programs (MILPs), branching occurs on the integer variables only, and BB terminates finitely [54]. Finite termination also occurs with some nonlinear problems [1,9], although in general BB applied to MINLPs — called spatial BB (sBB) — can only terminate finitely with an ε-approximate optimum for a given $\varepsilon > 0$.

BB usually converges slowly on problems (1) whose solution set has many symmetries because many leaf nodes in the BB tree may contain (symmetric) global optima: hence, no node in the paths leading from the root to these leaf nodes can ever be pruned. So, in general, we expect symmetric problems to yield larger BB trees. It is worth pointing out, however, that we carried out a few experiments using other solution methods than BB: these provided evidence to the effect that local-search based heuristics usually find optima faster if there are many of them — so it may not be worth breaking symmetries when using a heuristic method.

In this paper we describe methods to speed up the BB solution process applied to symmetric MILPs and MINLPs via a reformulation of the narrowing type [31].

Definition 1 Given a problem P, a *narrowing* Q of P is such that (a) there is a function $\eta : \mathscr{F}(Q) \to \mathscr{F}(P)$ for which $\eta(\mathscr{G}(Q)) \subseteq \mathscr{G}(P)$, and (b) Q is infeasible only if P is.

The proposed narrowing rests on adjoining some static symmetry breaking inequalities (SSBIs) [38] to the original formulation, i.e. inequalities that are designed to cut some of the symmetric solutions while keeping at least one optimal one. The reformulated problem is then solved by standard software packages such as CPLEX [22] (for MILPs) and COUENNE [4] (for MINLPs, replaced sometimes by BARON [47] on COUENNE's failures). In the same spirit as [31], our reformulation is completely *automatic*, in the sense that given the original problem we automatically compute the formulation group as well as the narrowing.

With respect to the existing literature about symmetry in mathematical programming, the main contribution of this paper is that of being able to deal with symmetric MINLPs and NLPs, and not just MILPs and Semidefinite Programs (SDPs) as was previously the case [34,23,19,43,52]. Moreover, whereas many existing works assume that the formulation group is known in advance, we propose a method for computing the formulation group of a MINLP automatically. The SSBIs we employ for constructing narrowing reformulations hold for every possible group and are well-behaved numerically. We provide computational validation of our ideas by (a) supplying formulation group tables for most of the instances in the MIPLib3 [6], MIPLib2003 [39], GlobalLib [10] and MINLPLib [11] (which also contains MacMINLP [26]); (b) evaluating BB performance on the symmetric instances in the aforementioned libraries, with and without SSBIs.

The rest of this paper is organized as follows. In Sect. 3 we perform a literature review concerning the use of group theoretical methods in mathematical programming. We define several groups linked to a mathematical program in Sect. 4. In Sect. 5 we introduce expression trees and DAGs for representing mathematical functions. We explain in Sect. 6 how to compute a formulation group automatically. Sect. 7 introduces several types of SSBIs and some ways to combine them. Computational results validating the proposed approach are given in Sect. 8: these include formulation group tables (Sect. 8.3) as well as results tables (Sect. 8.4).

2 Notation

Most of the groups considered in this paper act on vectors in $\mathbb{R}^n$ by permuting the components. Permutations act on sets of vectors by acting on each vector in the set. We denote the identity permutation by e. We employ standard group nomenclature: S_n, C_n are the symmetric and cyclic groups of order n, and D_{2n} is the dihedral group of order n (i.e. the group of rotations and reflections of a regular n-polygon in the plane). If G is a *subgroup* of H, we write $G \le H$. If G, H are groups, then the cartesian (set) product $G \times H$ can be endowed with a group structure by defining $(\pi,\sigma)(\pi',\sigma') = (\pi\pi',\sigma\sigma')$ for all $(\pi,\sigma),(\pi',\sigma') \in G \times H$. Two groups G,H are *isomorphic* (denoted $G \cong H$) if there is a *group homomorphism* $\phi : G \to H$ (i.e. ϕ is such that $\phi(\pi\pi') = \phi(\pi)\phi(\pi')$ for all $\pi,\pi' \in G$) which is both injective and surjective.

For a group $G \le S_n$ and a set X of row vectors, $XG = \{xg \mid x \in X \wedge g \in G\}$; if Y is a set of column vectors, $GY = \{gy \mid y \in Y \wedge g \in G\}$. If $X = \{x\}$, we denote XG by xG (and similarly GY by Gy if $Y = \{y\}$). We say that G fixes X setwise if $XG = X$, and pointwise if $\forall x \in X\ xg = x$ (similarly for Y — if not otherwise specified, we shall refer to setwise rather than pointwise fixing). We refer to xG as the *orbit* of x in G (similarly for Gy). In computational group theory literature the notation $\mathrm{orb}(x,G)$ is sometimes employed instead of the more algebraic xG. The (setwise) *stabilizer* $\mathrm{stab}(X,G)$ of a set X with respect to a group G is the largest subgroup H of G that fixes X (i.e. such that $XH = X$). For any permutation $\pi \in S_n$, let $\Gamma(\pi)$ be the set of its disjoint cycles, so that

$$\pi = \prod_{\tau \in \Gamma(\pi)} \tau.$$

For a group G and $\pi \in G$ let $\langle\pi\rangle$ be the smallest subgroup of G containing π, and for a subset $S \subseteq G$ let $\langle S\rangle$ be the smallest subgroup of G containing all elements of S (we also use the terminology *subgroup generated by* π and, respectively, S), in which case S is a set of *generators* of $\langle S\rangle$. For any $\pi \in S_n$, let $o(\pi) = |\langle\pi\rangle|$ denote the *order* of π. If π is expressed as a product of disjoint cycles, $o(\pi)$ turns out to be the least common multiple of all the cycle lengths.

Given $B \subseteq \{1,\ldots,n\}$, $\mathrm{Sym}(B)$ is the symmetric group of all the permutations of elements in B. A permutation $\pi \in S_n$ is limited to B if it fixes every element outside B; π acts on $B \subseteq \{1,\ldots,n\}$ as a permutation $\rho \in \mathrm{Sym}(B)$ if

$$\prod_{\tau \in \Gamma(\pi) \cap \mathrm{Sym}(B)} \tau = \rho,$$

in which case we denote ρ by $\pi[B]$, and call ρ the *restriction* of π to B. Because disjoint cycles commute, it follows from the definition that for all $k \in \mathbb{N}$, $\pi^k[B] = (\pi[B])^k$. A group G of permutations of S_n with generators $\{g_1,\ldots,g_s\}$ acts on $B \subseteq \{1,\ldots,n\}$ as H if $\langle g_i[B] \mid i \le s\rangle = H$; in this case we denote H by $G[B]$. If B is an orbit of the natural action of G on the integers, then it is easy to show that $G[B]$ is a *transitive constituent* of G, defined [21] as the set of restrictions to B of the elements of G whenever B is an orbit. In general, though, $G[B]$ may not even need to be a subgroup of G: take $G = \langle(1,2)(3,4),(1,3),(4,2)\rangle$ and $B = \{1,2\}$, then $G[B] = \langle(1,2)\rangle \not\le G$. Let $B,D \subseteq \{1,\ldots,n\}$ with $B \cap D = \emptyset$; if $\pi \in S_n$ fixes both B,D setwise, it is easy to show that $\pi[B \cup D] = \pi[B]\pi[D]$.

3 Literature review

We provide here an essential review of group-based methods in mathematical programming, with the notable exceptions of SDP-related results [52] and Constraint Programming (CP) [13] because mostly out of scope — see [38] for more information. Notwithstanding, a technique for automatic symmetry detection in CP bearing some similarity to the one proposed here can be found in [45]. The existing work may be classified in three broad categories: (a) the abelian group approach proposed by Gomory to write integer feasibility conditions for Integer Linear Programs (ILPs), not reviewed here because out of scope (see [30] for details); (b) symmetry-breaking techniques for specific problems, whose symmetry group can be computed in advance; (c) general-purpose symmetry group computations and symmetry-breaking techniques to be used in BB-type solution algorithms.

Category (b) is possibly the richest in terms of number of published papers. Many types of combinatorial problems exhibit a certain amount of symmetry. Symmetries are usually broken by means of specific branching techniques (e.g. [36]), appropriate global cuts (e.g. [50]) or special formulations [25,8] based on the problem structure. The main limitation of the methods in this category is that they are difficult to generalize and/or to be made automatic.

Category (c) contains three main research streams. The first was established by Margot in the early 2000s [34,35], and is applicable to problems in general form (1) where $x^L = 0, x^U = 1$, i.e. Binary Linear Programs (BLPs). Margot [34,38] defines the *relaxation group* $G^{\mathrm{LP}}(P)$ of a BLP P as:

$$G^{\mathrm{LP}}(P) = \{\pi \in S_n \mid c\pi = c \wedge \exists \sigma \in S_n \; (\sigma b = b \wedge \sigma A \pi = A)\}, \tag{2}$$

or, in other words, all relabellings of problem variables for which the objective function and constraints are the same. The relaxation group (2) is used to derive effective BB pruning strategies by means of isomorphism pruning and isomorphism cuts local to some selected BB tree nodes (Margot extended his work to general integer variables in [37]). Further results along the same lines, where branching on symmetric nodes at the same level is carried out implicitly (*orbital branching*), can be obtained for covering and packing problems [43, 44]: if O is an orbit of a certain subgroup of the relaxation group, at each BB node the disjunction $(\bigvee_{i\in O} x_i = 1) \vee \sum_{i\in O} x_i = 0$ induces a feasible division of the search space; orbital branching restricts this disjunction to $x_h = 1 \vee \sum_{i\in O} x_i$ where h is an arbitrary index in O. A method for finding the MILP relaxation group (2), based on solving an auxiliary MILP encoding the conditions $\sigma A \pi = A$, $c\pi = c$ and $\sigma b = b$ in the constraints, was proposed in [29].

The second stream was established by Kaibel et al. in 2007 [23] (also see [16]). Symmetries in the column space (i.e. permutations of decision variables) of binary ILPs having 0-1 constraint matrices are shown to affect the quality of the linear programming bound. Limited only to permutations in cyclic and symmetric group, complete descriptions of *orbitopes* are provided by means of linear inequalities. Let x' be a point in $\{0,1\}^n$ (the solution space), with $n = pq$, so that we can arrange the components of x' in a matrix C. Given a group G and $\pi \in G$, for all 0-1 $p \times q$ matrices C let πC be the matrix obtained by permuting the columns of C according to π. Let GC be the orbit of C under the action of all $\pi \in G$, $\overline{GC}$ be the lexicographically maximal matrix in GC (ordering matrices by rows first) and $\mathscr{M}^{\max}_{pq}(G)$ be the set of all $\overline{GC}$. Then the *full orbitope* associated with G is $\mathrm{conv}(\mathscr{M}^{\max}_{pq}(G))$. An automatic symmetry detection method for certain orbitopal symmetries that works in linear time is described in [5]. Inspired by the work on orbitopes, E. Friedman recently proposed a similar but extended approach [19] leading to the application of fundamental domains (see [38] for a definition of this well-known concept) to symmetry reduction: given a feasible polytope $X \subseteq [0,1]^n$ with integral extreme points and a group G acting as an affine transformation on X (i.e. for all $\pi \in G$ there is a matrix $A \in GL(n)$ and an n-vector d such that $\pi x = Ax + d$ for all $x \in X$), a fundamental domain is a subset $F \subset X$ such that $GF = X$.

4 Groups of a mathematical program

Given a MINLP P as in (1), the *solution group* $G^*(P)$ of P is defined as $\mathrm{stab}(\mathscr{G}(P), S_n)$, i.e. the group of all permutations of variable indices mapping global optima into global optima. When P is a MILP, $G^*(P)$ contains as a subgroup the *symmetry group* of P, defined for MILPs in [38] as the group of permutations mapping feasible solutions into feasible solutions with the same objective function value. Solution groups are hard to compute for a general MINLP (1) because presumably explicit knowledge of $\mathscr{G}(P)$ is needed *a priori*. We consider the group $\bar{G}_P$ that "fixes the formulation" of P:

$$\begin{aligned}\bar{G}_P = \{\pi \in S_n \mid Z\pi = Z \wedge \forall x \in \mathrm{dom}(f)\ f(x\pi) = f(x) \wedge \\ \exists\, \sigma \in S_m\ (\sigma b = b \wedge \forall x \in \mathrm{dom}(g)\ \sigma g(x\pi) = g(x))\}.\end{aligned} \tag{3}$$

It is easy to show that $\bar{G}_P \leq G^*(P)$: let $\pi \in \bar{G}_P$ and $x^* \in \mathscr{G}(P)$; $x^*\pi \in \mathscr{F}(P)$ because $Z\pi = Z$, $g(x^*\pi) = \sigma^{-1} g(x^*)$ and $\sigma^{-1} b = b$; and it has the same function value because $f(x^*\pi) = f(x^*)$ by definition. Thus $\mathscr{G}(P)\pi = \mathscr{G}(P)$ and $\pi \in G^*(P)$.

The two most problematic conditions that need testing to ascertain whether a given permutation π is in $\bar{G}_P$ are:

$$\begin{aligned}\forall x \in \mathrm{dom}(f) \quad f(x\pi) &= f(x) \\ \exists \sigma \in S_m\ \forall x \in \mathrm{dom}(g) \quad \sigma g(x\pi) &= g(x).\end{aligned}$$

Since NONLINEAR EQUATIONS (determining if a set of general nonlinear equations has a solution) is an undecidable problem in general [55], such tests are algorithmically infeasible. We therefore assume that for functions $f_1, f_2 : \mathbb{R}^n \to \mathbb{R}$ we have an oracle `equal`(f_1, f_2) that, if it returns `true`, then $\mathrm{dom}(f_1) = \mathrm{dom}(f_2)$ and $\forall x \in \mathrm{dom}(f_1)\ (f_1(x) = f_2(x))$, in which case we write $f_1 \equiv f_2$. We remark that we do not ask that the `equal` oracle returning `false` should imply $f_1 \neq f_2$: although this will make equality a stricter notion than it need be (so some pairs of equal functions will not belong to the $\equiv$ relation), it will allow us to implement the oracle efficiently by means of expression trees (see Sect. 5). We can now define the *formulation group* of a MINLP P as follows:

$$G_P = \{\pi \in S_n \mid Z\pi = Z \wedge f(x\pi) \equiv f(x) \wedge \exists\, \sigma \in S_m\ (\sigma g(x\pi) \equiv g(x) \wedge \sigma b = b)\}. \tag{4}$$

The structure of G_P depends on the oracle used to implement equality testing. With respect to a trivial oracle always returning `false`, for example, G_P would always only consist of the identity. Because for any function h, $h(x\pi) \equiv h(x)$ implies $h(x\pi) = h(x)$ for all $x \in \mathrm{dom}(h)$, it is clear that $G_P \leq \bar{G}_P$. Thus, it also follows that $G_P \leq G^*(P)$.

Although $\bar{G}_P$ is defined for any MINLP (1), if P is a BLP, then $\bar{G}_P$ is the same group as $G^{\mathrm{LP}}(P)$, as the following result shows.

Proposition 2 *Given a problem P as in (1) such that f, g are linear forms, $Z = \{1, \ldots, n\}$ and $x^L = \mathbf{0}, x^U = \mathbf{1}$, we have $\bar{G}_P = G^{\mathrm{LP}}(P)$.*

Proof Let $\pi \in G^{\mathrm{LP}}(P)$; then (a) $c\pi = c$ and (b) $\exists \sigma \in S_m$ such that $\sigma b = b$ and $\sigma A b = A$. Let $f(x) = cx$ in (1); then $f(x\pi) = c(x\pi) = (c\pi)x$, and by (a) we have $f(x\pi) = cx = f(x)$. Now let $g(x) = Ax$ in (1); then $g(x\pi) = A(x\pi) = (A\pi)x$, and by (b) there is $\sigma \in S_m$ such that $\sigma b = b$ and $\sigma A\pi = A$. Thus $\sigma g(x\pi) = \sigma((A\pi)x) = (\sigma A\pi)x = Ax = g(x)$, and $\pi \in \bar{G}_P$. The implication $\bar{G}_P \leq G^{\mathrm{LP}}(P)$ follows directly from the definition (3) if P is a BLP. $\square$

5 A function equality test oracle

Any mathematical expression consisting of a finite sequence of operator symbols, variable symbols and numerical constants can be represented by an n-ary *expression tree* [15, 14, 3]. We consider a finite set $\mathscr{O}$ of operators ordered according to a given order (for example, lexicographically according to their English names); we remark that operators can be unary (such as logarithm, exponential, sine, cosine, etc.), binary (such as fraction, difference, power) or k-ary (such as sum and product) for some positive integer k. The usual operator precedences, modified by parentheses, apply. Given a function $h(x)$, its expression tree is a directed tree $h = (V_h, A_h)$ where V_h is partitioned in leaf nodes (labelled with variable symbols from $x_1, \ldots, x_n$ and numerical constants) and non-leaf nodes (labelled with operator symbols from $\mathscr{O}$), and an arc (u, v) is in A_h if v is an argument of the operator node u. The tree h is constructed recursively as follows: the root of h is the lexicographically smallest operator $\otimes$ of lowest precedence in $h(x)$. Let $h_1(x), \ldots, h_K(x)$ be the arguments of $\otimes$. Since each $h_k(x)$ is a mathematical expression, by induction it is represented by a tree $h_k = (V_{h_k}, A_{h_k})$. The vertex set V_h is then defined as $\{\otimes\} \cup \bigcup_{k \leq K} V_{h_k}$ and A_h as $\bigcup_{k \leq K}((\otimes, h_k) \cup A_k)$. In general, expressions need not have unique trees. However, the number of trees corresponding to a given function can be decreased by defining a set of simplification rules ([28], p. 246-247) and an arbitrary argument ordering for each operator (e.g. constants first, then variables in lexicographic ordering, then other operators in the ordered set $\mathscr{O}$ [27]). If $f : \mathbb{R}^n \to \mathbb{R}$, given a vector of values $x' \in \mathbb{R}^n$, the value $f(x')$ can be obtained algorithmically by a simple recursive procedure `eval(`f`,`x'`)` on the expression tree f ([28], p. 243), which returns the symbol `NaN` (Not a Number) whenever $f(x')$ is undefined. The `equal` oracle for two expression trees f, g is defined in Algorithm 1.

Using Algorithm 1, it is easy to show that if `equal(`f_1`,`f_2`)=true`, then $f_1(x) = f_2(x)$ for all x in the domains of f_1, f_2, whereas the converse is not true (e.g. $\sin(x) = \cos(x + \pi/2)$ for all $x \in \mathbb{R}$ but the trees corresponding to $\sin(x)$ and $\cos(x + \pi/2)$ are different). As remarked in Sect. 4, if `equal(`f_1`,`f_2`)=true` we write $f_1(x) \equiv f_2(x)$ (in this expression there is no need to quantify over x, as $\equiv$ is an equality relation between the two trees representing $f_1(x), f_2(x)$).

Restricted to linear forms, the relation $\equiv$ is the same as equality.

Lemma 3 *If f_1, f_2 are linear forms, then $\forall x \in \mathrm{dom}(f_1)$ $f_1(x) = f_2(x)$ (written $f_1 = f_2$) implies $f_1 \equiv f_2$.*

Proof Assume $f_1 = f_2$; let $f_1(x) = cx$ and $f_2(x) = dx$, where $c = (c_1, \ldots, c_n)$, $d = (d_1 \ldots, d_n)$, $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$. We define the *canonical* expression tree for the linear form cx by:

$$\begin{aligned} V_c &= \{+, \times_1, \ldots, \times_n, c_1, \ldots, c_n, x_1, \ldots, x_n\} \\ A_c &= \{(+, \times_i), (\times_i, c_i), (\times_i, x_i) \mid i \leq n\}; \end{aligned}$$

it is easily shown that the canonical tree is unique and that there are finite deterministic algorithms reducing any other expression tree representing cx to the canonical tree (similarly

Algorithm 1 *boolean* `equal`(f_1, f_2)

```
Input expression trees f1, f2
if f1 and f2 are both leaf nodes then
  if f1 and f2 are both variable nodes and represent the same variable then
    return true
  else
    if f1 and f2 are both constant nodes and represent the same constant then
      return true
    else
      return false
    end if
  end if
else
  if f1 and f2 are both operator nodes and have the same arity k then
    r ←true
    for i = 1 to k do
      let f1', f2' be the i-th nodes in the stars of f1, f2
      r ←equal(f1', f2')
      if r =false then
        exit for
      end if
    end for
    return r
  else
    return false
  end if
end if
```

for dx). Now let $f_1(x) = f_2(x)$ for all $x \in \mathbb{R}^n$; then $\forall x\ (cx = dx)$, which implies $c = d$, and thus the canonical expression tree for c is identical to the canonical expression tree for d. This shows that $f_1 \equiv f_2$. $\square$

By Lemma 3 and Prop. 2, given a problem P as in (1) such that f, g are linear forms, $Z = \{1, \dots, n\}$, $x^L = \mathbf{0}$ and $x^U = \mathbf{1}$, we have $G_P = G^{\mathrm{LP}}(P)$.

The functions f, g appearing in (1) have the property that their argument list x is the same, so the trees for $f, g_1, \dots, g_m$ can share the same variable leaf nodes. This yields a Directed Acyclic Graph (DAG) $D_P = (V_P, A_P)$ where $V_P = V_f \cup \bigcup_{i \le m} V_{g_i}$ and $A_P = A_f \cup \bigcup_{i \le m} A_{g_i}$. D_P is a DAG representing the mathematical structure of the functions of P. It is rooted at the smallest operators of lowest precedence in $f, g_1, \dots, g_m$; its leaf nodes are the problem variables and all the problem constants. More comprehensive discussions about expression DAGs and their uses in optimization can be found in [4, 42, 48].

6 Automatic computation of the formulation group

6.1 Fixed subsets of DAG nodes

We emphasize the following subsets of V_P: the set $\mathscr{S}_F$ of all root nodes corresponding to objective functions (in this section we generalize to multi-objective problems although in practice we only consider single objective problems), the set $\mathscr{S}_C$ of all root nodes corresponding to constraints, the set $\mathscr{S}_O$ of all operator nodes, the set $\mathscr{S}_K$ of all constant nodes and the set $\mathscr{S}_V$ of all variable nodes. We remark that $\mathscr{S}_F \cup \mathscr{S}_C \cup \mathscr{S}_O \cup \mathscr{S}_K \cup \mathscr{S}_V = V_P$ but the union is not disjoint as $\mathscr{S}_F \cup \mathscr{S}_C \subseteq \mathscr{S}_O$. For a node $v \in \mathscr{S}_F$, we denote the optimization direction by $d(v)$; for a node $v \in \mathscr{S}_C$, we denote the constraint sense by $s(v)$ and the corresponding constraint RHS constant by $b(v)$. For a node $v \in \mathscr{S}_O$, we let $\ell(v)$ be the level of v in D_P, $\lambda(v)$ be its operator label (operator name) and $\mathrm{o}(v)$ be the rank of v in the argument list of its parent node if the latter represents a noncommutative operator, or 1 otherwise. We remark that for nodes in $\mathscr{S}_O$ the level in D_P is well-defined, as the only nodes in D_P with more than one incoming arc are the leaf nodes, and no operator node can be a leaf. For $v \in \mathscr{S}_K$, we let $\mu(v)$ be the value of v. For $v \in \mathscr{S}_V$ we let $r(v)$ be the 2-vector of lower and upper variable bounds for v and $\zeta(v)$ be 1 if v represents an integral variable or 0 otherwise.

We define the relation $\sim$ on V_P as follows.

$$\begin{aligned}
\forall u,v \in V_P \quad u \sim v \Leftrightarrow\ & (u,v \in \mathscr{S}_F \wedge d(u) = d(v)) \\
\vee\ & (u,v \in \mathscr{S}_C \wedge s(u) = s(v) \wedge b(u) = b(v)) \\
\vee\ & (u,v \in \mathscr{S}_O \wedge \ell(u) = \ell(v) \wedge \lambda(u) = \lambda(v) \wedge \mathrm{o}(u) = \mathrm{o}(v)) \\
\vee\ & (u,v \in \mathscr{S}_K \wedge \mu(u) = \mu(v)) \\
\vee\ & (u,v \in \mathscr{S}_V \wedge r(u) = r(v) \wedge \zeta(u) = \zeta(v)).
\end{aligned}$$

It is easy to show that $\sim$ is an equivalence relation on V_P, and therefore partitions V_P into K disjoint subsets $V_1, \ldots, V_K$.

6.2 The projection homomorphism

Let $G \le S_n$ and ω be a subset of $\{1,\ldots,n\}$. Let $H = \mathrm{Sym}(\omega)$ and define the mapping $\varphi : G \to H$ by $\varphi(\pi) = \pi[\omega]$ for all $\pi \in G$.

Theorem 4 *φ is a group homomorphism if and only if G stabilizes ω setwise.*

Proof ($\Rightarrow$) Assume φ is a group homomorphism and suppose there is $\sigma \in G$ and $i \in \omega$ such that $\sigma(i) = j \notin \omega$. Take any permutation $\pi \in H$ such that $\pi(i) = k \in \omega$, $k \neq i$. Then the action of $\pi\sigma$ is to move i to j first (because of σ), and then fix it to j (because of π), which means that $(\pi\sigma)[\omega]$ simply fixes i; on the other hand, the action of $\pi[\omega]\sigma[\omega]$ on i is to fix it first (because of $\sigma[\omega]$) and then move it to k (because of $\pi[\omega]$), hence $\varphi(\pi\sigma) \neq \varphi(\pi)\varphi(\sigma)$, against the assumption. Thus $\sigma(i) \in \omega$ for all $i \in \omega$ and $\sigma \in G$, which implies $G\omega = \omega$.

($\Leftarrow$) Assume $G\omega = \omega$ and let $\pi, \sigma \in G$. First, for a single cycle γ fixing ω pointwise, we obviously have $\gamma[\omega] = e$. Now consider two single cycles β, γ appearing in the disjoint cycle product representation of some permutations of G. Since G fixes ω setwise, either: (1) both $\beta, \gamma \in H$, or (2) one is in H and the other is in $S_n \smallsetminus H$, or (3) both are in $S_n \smallsetminus H$. For case (1), $\beta, \gamma \in H$ implies $\beta[\omega] = \beta$ and $\gamma[\omega] = \gamma$, which yields $(\beta\gamma)[\omega] = \beta\gamma = \beta[\omega]\gamma[\omega]$. For (2), assuming without loss of generality $\beta \in H$ and $\gamma \notin H$, then $(\beta\gamma)[\omega] = \beta[\omega] = \beta[\omega]e = \beta[\omega]\gamma[\omega]$. For (3), $(\beta\gamma)[\omega] = e = ee = \beta[\omega]\gamma[\omega]$. Thus $\varphi(\beta\gamma) = \varphi(\beta)\varphi(\gamma)$. Next, notice that:

$$\pi\sigma = \left(\prod_{\tau \in \Gamma(\pi)} \tau\right)\left(\prod_{\tau \in \Gamma(\sigma)} \tau\right) = \prod_{\tau \in \Gamma(\pi)\cup\Gamma(\sigma)} \tau.$$

Hence,

$$\begin{aligned}
\varphi(\pi\sigma) = \varphi\left(\prod_{\tau \in \Gamma(\pi)\cup\Gamma(\sigma)} \tau\right) &= \prod_{\tau \in \Gamma(\pi)\cup\Gamma(\sigma)} \varphi(\tau) = \prod_{\tau \in (\Gamma(\pi)\cup\Gamma(\sigma))\cap H} \tau \\
&= \left(\prod_{\tau \in \Gamma(\pi)\cap H} \tau\right)\left(\prod_{\tau \in \Gamma(\sigma)\cap H} \tau\right) = \varphi(\pi)\varphi(\sigma),
\end{aligned}$$

which completes the proof. $\square$

6.3 Mapping graph automorphisms onto the formulation group

For a digraph $D = (V,A)$, its automorphism group $\mathrm{Aut}(D)$ is the group of vertex permutations γ such that $(\gamma(u),\gamma(v)) \in A$ for all $(u,v) \in A$ [46]. Let $G^{\mathrm{DAG}}(P)$ be the largest subgroup of $\mathrm{Aut}(D_P)$ fixing V_k for all $k \le K$ (i.e. containing only vertex permutations γ such that $\gamma V_k = V_k$ for all $i \le K$). For ease of notation, assume without loss of generality that the vertices of D_P are ordered so that for all $j \le n$, the j-th vertex corresponds to the leaf node for variable x_j (the rest of the order is not important), i.e. $\mathscr{S}_V = \{1,\ldots,n\}$.

Lemma 5 *$G^{\mathrm{DAG}}(P)$ fixes $\mathscr{S}_V$ setwise.*

Proof By definition, all permutations of $G^{\mathrm{DAG}}(P)$ fix all V_k's (setwise). In particular, since $(u,v \in \mathscr{S}_V \wedge r(u) = r(v) \wedge \zeta(u) = \zeta(v))$ implies $u \sim v$, there will be a subset $\mathscr{K}$ of $\{1,\ldots,K\}$ such that $\mathscr{S}_V = \bigcup_{k\in\mathscr{K}} V_k$. Hence $G^{\mathrm{DAG}}(P)\mathscr{S}_V = \mathscr{S}_V$ as claimed. $\square$

Corollary 6 *The map $\varphi : G^{\mathrm{DAG}}(P) \to Sym(\mathscr{S}_V)$ given by $\varphi(\gamma) = \gamma[\mathscr{S}_V]$ is a group homomorphism.*

Proof By Lemma 5 and Thm. 4. $\square$

Theorem 7 *$\mathrm{Im}\varphi = G_P$ groupwise.*

Proof We first remark that by Cor. 6, $\mathrm{Im}\varphi$ is endowed with a group structure, because $G^{\mathrm{DAG}}(P)/\mathrm{Ker}\varphi \cong \mathrm{Im}\varphi$. In particular, $\mathrm{Im}\varphi$ is a subgroup of S_n. Now let $\psi : G^{\mathrm{DAG}}(P) \to \mathrm{Sym}(\mathscr{S}_C)$ be given by $\psi(\gamma) = \gamma[\mathscr{S}_C]$. By an argument similar to that of Lemma 5, $G^{\mathrm{DAG}}(P)$ fixes $\mathscr{S}_C$ setwise, which implies that ψ is a group homomorphism by Thm. 4. Let $\sigma = \psi(\gamma)$ and $\pi = \varphi(\gamma)$. Because γ fixes each equivalence class V_k, we have $Z\pi = Z$, $f(x\pi) \equiv f(x)$, $\sigma b = b$ and $\sigma g(x\pi) \equiv g(x)$. Conversely, suppose $\pi \in G_P$ and there is no automorphism γ of D_P fixing all V_k's and such that $\varphi(\gamma) = \pi$. Then either $f(x\pi) \not\equiv f(x)$, or there is no $\sigma \in S_m$ such that $\sigma g(x\pi) \equiv g(x)$, or $\psi(\gamma)b \neq b$, contradicting the hypothesis. Thus, $\mathrm{Im}\varphi = G_P$ setwise. Since both are subgroups of S_n, the identity isomorphism shows that $\mathrm{Im}\varphi = G_P$ groupwise too. $\square$

By Thm. 7, we can automatically generate G_P by looking for the largest subgroup of $\mathrm{Aut}(D_P)$ fixing all V_k's. Thus, the problem of computing G_P has been reduced to computing the (generators of the) automorphism group of a certain vertex-coloured DAG. This is in turn equivalent to the GRAPH ISOMORPHISM (GI) problem [2]. GI is in **NP**, but it is not known whether it is in **P** or **NP**-complete. A notion of GI-completeness has therefore been introduced for those graph classes for which solving the GI problem is as hard as solving it on general graphs [51]. Rooted DAGs are GI-complete [7] but there is an $O(N)$ algorithm for solving the GI problem on trees ([46], Ch. 8.5.2).

Corollary 8 *If T' is a set of group generators of $G^{DAG}(P)$, then $T = \{\pi[\mathscr{S}_V] \mid \pi \in T'\}$ is a set of generators for G_P.*

7 Symmetry Breaking Constraints

In this section we shall discuss the automatic generation of two types of SSBIs, one of which is valid for symmetries in *any* group G_P, and the other only holds for the full symmetric group S_n. Because of their generality and of the usual trade-off between generality and efficacy, the general-purpose SSBIs we propose are not the tightest possible; however, it is their generality that makes their automatic generation feasible (and easy) for all MINLPs. We also propose tighter SSBIs that only hold for S_n, so that we can only generate them automatically for those instances displaying at least one orbit whose stabilizer is the symmetric group. Some works in the literature [50] suggest using very tight and rather general-purpose SSBIs based on interpreting a 0-1 vector as a base-k expansion of an integer number, with the constraints acting on the latter (also see [38], p. 667). Quite apart from the fact that these SSBIs only hold for integer variables with values in $\{0,\ldots,k\}$ (so they would not be applicable to continuous NLPs), it is well known that such SSBIs are badly scaled; so that although the corresponding narrowing is formally well-defined and symmetry-free, it is often much more difficult to solve correctly, in practice, than the original problem. We therefore limit our attention to SBCs that are numerically well behaved.

We first give a formal definition of SSBIs that makes them depend on a group rather than a set of solutions.

Definition 9 Given a permutation $\pi \in S_n$ acting on the component indices of the vectors in a given set $X \subseteq \mathbb{R}^n$, the constraints $g(x) \le 0$ (that is, $\{g_1(x) \le 0,\ldots,g_q(x) \le 0\}$) are *symmetry breaking constraints* (SBCs) with respect to π and X if there is $y \in X$ such that $g(y\pi) \le 0$. Given a group G, $g(x) \le 0$ are SBCs w.r.t G and X if there is $y \in XG$ such that $g(y) \le 0$.

If there are no ambiguities as regards X, we simply say "SBCs with respect to π" (respectively, G). In most cases, $X = \mathscr{G}(P)$. The following facts are easy to prove.

1. For any $\pi \in S_n$, if $g(x) \le 0$ are SBCs with respect to π, X then they are also SBCs with respect to $\langle \pi \rangle, X$.
2. For any $H \le G$, if $g(x) \le 0$ are SBCs with respect to H, X then they are also SBCs with respect to G, X.
3. Let $g(x) \le 0$ be SBCs with respect to $\pi \in S_n, X \subseteq \mathbb{R}^n$ and let $B \subseteq \{1,\ldots,n\}$. If $g(x) \equiv g(x[B])$ (i.e., if the constraints g only involve variable indices in B) then $g(x) \le 0$ are also SBCs with respect to $\pi[B], X[B]$.

As regards Fact 3, if $g(x) \equiv g(x[B])$ we denote the SBCs $g(x) \le 0$ by $g[B](x) \le 0$; if B is the domain of a permutation $\alpha \in \mathrm{Sym}(B)$, we also use the notation $g[\alpha](x) \le 0$.

Example 10 Let $y = (1,1,-1), X = \{y\}$ and $\pi = (1,2,3)$; then $\{x_1 \le x_2, x_1 \le x_3\}$ are SBCs with respect to π and X because $y\pi$ satisfies the constraints. The inequalities $\{x_1 \le x_2, x_2 \le x_3\}$ are SBCs with respect to S_3 and X because $(-1,1,1) = y(1,2,3) \in XS_n$, but not with respect to to $\langle (2,3) \rangle$ and X because $X\langle (2,3) \rangle = \{y, y(2,3)\} = \{(1,1,-1),(1,-1,1)\}$ and neither vector satisfies the constraints.

We use SBCs to yield narrowings of the original problem P.

Theorem 11 *If $g(x) \le 0$ are SBCs for any subgroup G of G_P and $\mathscr{G}(P)$, then the problem Q obtained by adjoining $g(x) \le 0$ to the constraints of P is a narrowing of P.*

Proof By Defn. 1, we must provide a map $\mathscr{G}(Q) \to \mathscr{G}(P)$ and show that if P is feasible then Q is feasible. Assume $\mathscr{F}(P) \neq \emptyset$; then $\mathscr{G}(P) \neq \emptyset$. By definition of SBCs, there is $y \in \mathscr{G}(P)G$

such that $g(y) \le 0$. Since $G \le G_P \le G^*(P) = \mathrm{stab}(\mathscr{G}(P), S_n)$, it follows that $\mathscr{G}(P)G = \mathscr{G}(P)$, so that $y \in \mathscr{G}(P)$. Thus, y satisfies the constraints of P and also $g(y) \le 0$, which means that $y \in \mathscr{F}(Q)$, as required. Now, let η be the identity map; since $\mathscr{F}(Q) \ne 0$ it follows that $\mathscr{G}(Q)$ contains at least one element x. Since $\mathscr{F}(Q) \subseteq \mathscr{F}(P)$ (because Q is as P with additional constraints) and the objective functions of P, Q are equal, $\eta(x) = x \in \mathscr{G}(P)$. □

We now describe a way to combine SBCs. Since adjoining more constraints to a formulation results into a smaller feasible region and fewer optima, combined SBCs should yield better narrowings.

Theorem 12 *Let $\omega, \theta \subseteq \{1, \dots, n\}$ be such that $\omega \cap \theta = \emptyset$. Consider $\rho, \sigma \in G_P$, and let $g[\omega](x) \le 0$ be SBCs w.r.t. $\rho, \mathscr{G}(P)$ and $h[\theta](x) \le 0$ be SBCs w.r.t. $\sigma, \mathscr{G}(P)$. If $\rho[\omega], \sigma[\theta] \in G_P[\omega \cup \theta]$ then the system of constraints $c(x) \le 0$ consisting of $g[\omega](x) \le 0$ and $h[\theta](x) \le 0$ is an SBC system for $\rho\sigma$.*

Proof Let $y \in \mathscr{G}(P)$. Since $g[\omega](x)$ only depends on variable indices in ω, $g[\omega](y\rho[\omega]) \le 0$. Likewise, $h[\theta](y\sigma[\theta]) \le 0$. The fact that $\rho[\omega], \sigma[\theta] \in G_P[\omega \cup \theta]$ implies that $\rho[\omega \cup \theta] = \rho[\omega]$ and $\sigma[\omega \cup \theta] = \sigma[\theta]$, and in turn that $\rho[\theta] = \sigma[\omega] = e$. Since σ fixes ω pointwise, the action of $\rho\sigma$ on ω reduces to the action of ρ on ω, and similarly for ρ and θ, i.e. $(\rho\sigma)[\omega] = \rho[\omega]$ and $(\rho\sigma)[\theta] = \sigma[\theta]$. Thus, $g[\omega](y\rho\sigma) = g[\omega](y(\rho\sigma)[\omega]) = g[\omega](y\rho[\omega]) \le 0$ and $h[\theta](y\rho\sigma) = h[\theta](y(\rho\sigma)[\theta]) = h[\theta](y\sigma[\theta]) \le 0$, hence $c(y\rho\sigma) \le 0$ as claimed. □

Thm. 12 can easily be extended to sets of subsets of $\{1, \dots, n\}$ where the required conditions hold pairwise.

7.1 SBCs from orbits

Consider the set Ω of (nontrivial) orbits of the natural action of G_P on $\{1, \dots, n\}$. We pave the way for applying Thm. 12 to adjoin SBCs arising from different orbits. Since G_P acts transitively on each orbit $\omega \in \Omega$, for all $i \ne j \in \omega$ there is at least one permutation in G_P mapping i to j. Let $M^{ij} \subseteq G_P$ be the set of all such permutations. Let $\omega, \theta \in \Omega$ be such that:

1. $\forall i \ne j \in \omega \; \exists \rho \in M^{ij}$ s.t. $\gcd(o(\rho[\omega]), o(\rho[\theta])) = 1$; let $\tilde{R}$ be the set of all such ρ
2. $\forall i \ne j \in \theta \; \exists \sigma \in M^{ij}$ s.t. $\gcd(o(\sigma[\omega]), o(\sigma[\theta])) = 1$; let $\tilde{S}$ be the set of all such σ.

Lemma 13 *For all $\rho \in \tilde{R}$ and $\sigma \in \tilde{S}$, $\rho[\omega], \sigma[\theta] \in G_P[\omega \cup \theta]$.*

Proof Let $r = o(\rho[\theta])$.

$$\begin{aligned} \rho^r[\omega \cup \theta] &= \rho^r[\omega]\rho^r[\theta] && \text{as } \omega, \theta \text{ are (setwise fixed) orbits of } G_P \\ &= \rho[\omega]^r \rho[\theta]^r && \text{by definition (Sect. 2)} \\ &= \rho[\omega]^r && \text{because } r \text{ is the order of } \rho[\theta]. \end{aligned}$$

Now $\langle \rho[\omega]^r \rangle = \langle \rho[\omega] \rangle$ because $\gcd(o(\rho[\omega]), r) = 1$ by definition. Thus there is a positive integer t such that $\rho[\omega]^{rt} = \rho[\omega]$, which means that $\rho[\omega] = \rho[\omega]^{rt} = \rho^r[\omega]^t = \rho^r[\omega \cup \theta]^t \in G_P[\omega \cup \theta]$. The argument for $\sigma[\theta]$ is similar. □

Lemma 13 and Thm. 12 establish the following.

Corollary 14 *If $g[\omega](x) \le 0$ are SBCs w.r.t. some $\rho \in \tilde{R}$ and $h[\theta](x) \le 0$ are SBCs w.r.t. some $\sigma \in \tilde{S}$, the union of both systems is an SBC system for $\rho\sigma$.*

We now propose general-purpose SBCs, valid for G_P, which can be derived from any of its orbits.

Proposition 15 *Let $\omega \in \Omega$. The constraints*

$$\forall j \in \omega \setminus \{\min \omega\} \quad x_{\min \omega} \leq x_j. \tag{5}$$

are SBCs with respect to G_P.

Proof Let $y \in \mathscr{G}(P)$. Since all groups act transitively on each orbit, there is $\pi \in G_P$ mapping $\min y[\omega]$ to $y_{\min \omega}$. Thus, $y\pi$ satisfies (5). □

If there is $\omega \in \Omega$ such that the action of G_P on it is the symmetric group on ω, stronger SBCs than (5) hold. Let $\omega^- = \omega \setminus \{\max \omega\}$, and for each $j \in \omega^-$ let $j^+ = \min\{h \in \omega \mid h > j\}$ be the successor of j in ω.

Proposition 16 *Provided $G_P[\omega] = \mathrm{Sym}(\omega)$, the following constraints:*

$$\forall\ j \in \omega^- \quad x_j \leq x_{j^+} \tag{6}$$

are SBCs with respect to G_P.

Proof Let $y \in \mathscr{G}(P)$. Since $G_P[\omega] = \mathrm{Sym}(\omega)$, there is $\pi \in G_P$ such that $(y\pi)[\omega]$ is ordered by $\leq$. Therefore, $y\pi$ is feasible with respect to the constraints $\forall\ j \in \omega^-\ x_j \leq x_{j^+}$, which yields the result. □

By Cor. 14, any set of SBC systems with respect to transitive constituents of G_P whose corresponding orbits verify Conditions 1-2 (top of this subsection) pairwise is a system of SBCs w.r.t. G_P.

Proposition 17 *Let $\omega, \theta \in \Omega$ and assume $G_P[\omega \cup \theta]$ contains a subgroup $H \cong C_{|\omega|} \times C_{|\theta|}$ such that $H[\omega] \cong C_{|\omega|}$ and $H[\theta] \cong C_{|\theta|}$. Then ω, θ satisfy Conditions 1-2.*

Proof Let $\rho \in G_P$ such that $\rho[\omega \cup \theta] \in H$ be chosen so that $\langle \rho[\omega] \rangle = H[\omega] \cong C_{|\omega|}$ and $\rho[\theta] = e$. Then for all $i \neq j \in \omega$ there is an integer k such that ρ^k maps i to j and fixes θ, and hence $\rho^k \in M^{ij}$; $\rho[\theta] = e$ ensures $\gcd(o(\rho[\omega]), o(\rho[\theta])) = 1$. The argument for θ is similar.

Proposition 18 *Let $\omega, \theta \in \Omega$ and assume that $G_P[\omega \cup \theta]$ contains a subgroup H such that $H[\omega] \cong C_{|\omega|}$ and $H[\theta] \cong C_{|\theta|}$. If $\gcd(|\omega|, |\theta|) = 1$ then ω, θ satisfy Conditions 1-2.*

Proof Let $\rho \in G_P$ such that $\rho[\omega \cup \theta] \in H$ be chosen so that: (a) $\langle \rho[\omega] \rangle = H[\omega] \cong C_{|\omega|}$; (b) there is a single cycle $\alpha \in H[\theta]$ having length $|\theta|$ and an integer l such that $\rho[\theta] = \alpha^l$. Hence $s = o(\rho[\theta])$ divides $|\theta|$. Since $o(\rho[\omega]) = |\omega|$ and $\gcd(|\omega|, |\theta|) = 1$, $\langle \rho^s[\omega] \rangle \cong \langle \rho[\omega] \rangle$. Thus, for all $i \neq j \in \omega$ there is an integer k such that $(\rho^s)^k = \rho^{sk}$ maps i to j, and $\rho^{sk}[\theta] = \rho[\theta]^{sk} = (\rho[\theta]^s)^k = e^k = e$. Thus $\tau = \rho^{sk}$ is in M^{ij} and $\gcd(o(\tau[\omega]), o(\tau[\theta])) = 1$. The argument for θ is similar.

Since by Sect. 6 we can obtain the set T of generators of G_P, it is possible to compute the set of orbits Ω in time $O(n|T| + n^2)$ [12]. There are polynomial-time algorithms for testing group membership and subgroup inclusion [49]; algorithms for dealing with the transitive constituent homomorphism φ usually rest on the Schreier-Sims method for computing group generators (of which some implementations as a nearly linear-time Monte Carlo algorithm exist). Thus, deriving SBCs as per Prop. 15 and combining them using Prop. 18 are tasks whose algorithmic implementation is practically feasible.

7.2 Generating SBCs automatically

We aim to test two different approaches. In the first one, we simply pick the largest orbit, verify it contains a subgroup $C_{|\omega|}$, and adjoin the corresponding SBCs (5) to the original problem. In the second, we attempt to use Prop. 17 and 18 in order to adjoin SBCs (of type (6) if possible) from several orbits. Since (5) only impose a minimum element within a set of values, whereas (6) imposes a whole total order, the latter should yield a tighter narrowing than the former, and we expect a tight narrowing to be easier to solve by BB than a slack one. This is not always true in practice, however, because narrowing constraints may have some adverse effects too, such as making each BB node relaxation longer to solve and affecting the choice of branching variable and/or branching point.

A set $\omega \subseteq \{1,\dots,n\}$ is a *block* for G if $\forall g \in G\ (\omega g = \omega \vee \omega g \cap \omega = \emptyset)$. A group G is *primitive* if its only blocks are trivial (i.e. $\emptyset$, singletons and $\{1,\dots,n\}$). There are practically fast algorithms for testing groups for primitivity. Let $\omega \in \Omega$ be a nontrivial orbit of G_P, let T be the set of generators of G_P constructed as per Cor. 8, and for any $\omega \subseteq \{1,\dots,n\}$ let $T[\omega] = \{\pi[\omega] \mid \pi \in T\}$. We first remark that if $T[\omega]$ contains a cycle of length $|\omega|$, then $C_{|\omega|} \leq G_P[\omega]$; this provides a practical way of testing the hypotheses of Propositions 17 and 18. The following result can be used for testing the hypothesis of Prop. 16.

Proposition 19 *If $G_P[\omega]$ is primitive and $T[\omega]$ contains a transposition (i.e. a cycle of length 2), $G_P[\omega] = \mathrm{Sym}(\omega)$.*

Proof By [53], Thm. 13.3. □

Naturally, if $G_P[\omega] = \mathrm{Sym}(\omega)$ then $C_{|\omega|} \leq G_P[\omega]$, so Prop. 19 can also be used to test the hypotheses of Prop. 17 and 18.

In practice, we form a subset $\Lambda \subseteq \Omega$ of orbits which satisfy the hypotheses of Prop. 18 pairwise. Then, for each orbit ω in Λ we further verify whether $G_P[\omega]$ satisfies the hypotheses of Prop. 16 or not. Accordingly, for each orbit in Λ, we either output SBCs (6) or (5). We attempt to construct Λ so that it generates as many added constraints as possible, in the hope of yielding a significantly smaller feasible region. We adopt a greedy approach on the orbit length (Alg. 2).

8 Computational results

We report computational results of two kinds. We first attempt to determine a closed form description of G_P for all the considered instances (Tables 2-4). Secondly, we compare BB performances on the original and reformulated problems. We remark that our symmetry breaking efforts are limited to the adjoining of *static* constraints to the formulation (rather than employing *dynamic* symmetry breaking techniques [38]): with static techniques only, it is not so clear that the proposed approach helps in solving general MILPs, although we have interesting results for some selected instances. The performance on NLPs/MINLPs, on the other hand, is much better. Part of the reason for this is that MILP solvers are technically much more advanced than their NLP/MINLP counterparts — and our MILP solver of choice already contains some symmetry exploitation devices. The good results obtained on MILPs using dynamic symmetry breaking techniques [34, 17, 44], however, point to the fact that the type of automatic symmetry detection proposed in this paper might be complemented by dynamic symmetry breaking techniques and applied to MILPs quite successfully. This will make the object of further investigations.

Algorithm 2 A greedy algorithm for constructing SBCs.

Input P
Compute G_P (Sect. 6)
Let L be the list of all nontrivial orbits of the natural action of G_P over $\{1,\ldots,n\}$
Let $\Lambda = \emptyset$
while $|L| > 0$ **do**
  Let ω be the longest orbit in L
  Let $L \leftarrow L \smallsetminus \{\omega\}$
  if $C_{|\omega|} \leq G_P[\omega]$ **then**
    Let $t \leftarrow 1$
    for $\theta \in \Lambda$ **do**
      if $\gcd(|\omega|, |\theta|) > 1$ **then**
        Let $t \leftarrow 0$
        Break
      end if
    end for
    if $t = 1$ **then**
      Let $\Lambda \leftarrow \Lambda \cup \{\omega\}$
    end if
  end if
end while
for $\omega \in \Lambda$ **do**
  if $G_P[\omega] \cong \mathrm{Sym}(\omega)$ **then**
    Output SBCs (6)
  else
    Output SBCs (5)
  end if
end for

We employ two types of reformulations: *Narrowing1* is obtained by adjoining (5) for the longest orbit to the original formulation; *Narrowing2* adjoins the SBCs returned by Alg. 2. The BB solvers employed are: CPLEX 11.01 [22] for the MILP instances and COUENNE [4] for NLP and MINLP instances; since COUENNE is a relatively young solver, and not yet totally stable, BARON [47] was used whenever COUENNE failed. The solution statistics are:

1. the objective function value of the incumbent
2. the seconds of user CPU time taken (meaningful when below the 7200s limit)
3. the gap still open at termination
4. the number of BB nodes closed and those still on the tree at termination.

A first round of tests compares the statistics after two hours of computation time (per instance). In a second round of tests, we perform the same comparison with different termination criteria on a meaningful subset of instances. All results have been obtained on one 2.4GHz Intel Xeon CPU of a computer with 8 GB RAM (shared by 3 other similar CPUs) running Linux.

8.1 Implementation

We implemented two software systems: the first, `symmgroup`, computes an explicit description of the formulation group structure. The second, `reformulate`, implements Alg. 2 and produces a reformulated instance ready to be solved. The algorithm that computes the explicit description of a group structure given its generators has exponential worst-case complexity and is in practice quite slow, whereas reformulating entails computing the orbits from the generators, computing a group action on an orbit, verifying whether a generator has a certain length, and verifying whether a given group is primitive (all polynomial-time and also practically fast algorithms [49]). Thus, we were not always able to find the group description although we were able to reformulate the original problem to the correct narrowing. The implementation of `symmgroup` and `reformulate` is similar up to the stage where the group generators are computed. Both first call AMPL [18] to parse the instance; the ROSE Reformulation/Optimization Software Engine [32] AMPL-hooked solver is then called (with ROSE's `Rsymmgroup` reformulator) to produce a file representation of the problem expression DAG. This is then fed into *nauty*'s [41,40] `dreadnaut` shell to efficiently compute the generators of $\text{Aut}(D_P)$ (see Sect. 6). A system of shell scripts and Unix tools parses the *nauty* output to form a valid GAP [20] input. At this stage, `symmgroup` uses GAP's `StructureDescription` command to output the formulation group description, whereas `reformulate` uses a purpose-built GAP code that simply outputs SBCs (5) relating to the longest orbit (*Narrowing1*) or implementing Alg. 2 (*Narrowing2*).

8.2 Test set

Our test set consists of almost all the instances in the best known mathematical programming instance libraries: MIPLib3 [6], MIPLib2003 [39] (containing MILPs), GlobalLib [10] (containing NLPs) and MINLPLib [11] (containing MINLPs). We have not tested some of the largest instances (listed in Table 5) because of RAM and CPU time limitations. Our test set consists of a grand total of 669 instances partitioned in the different libraries as given in Table 1 — this table also reports the number of instances whose formulation have nontrivial groups. The instance sizes can be found in the online appendix.

Library	*Instances*	*Nontrivial G_P*	*% of library*
`miplib3`	62	22	35.4%
`miplib2003`$\setminus$`miplib3`	20	7	35.0%
`globallib`	390	58	14.8%
`minlplib`	197	32	16.2%
Total	**669**	**112**	**16.7%**

Table 1 Instance libraries statistics.

8.3 Group tables

In Table 2 we report formulation groups for all (MILP) instances of the MIPLib3 and MIPLib2003 libraries. In Table 3 we report formulation groups for all (NLP) instances of the GlobalLib library. In Table 4 we report formulation groups for all (MINLP) instances of the MINLPLib library. We remark that all group tables have been compiled with the AMPL presolver disabled. Since the group depends on the formulation rather than the problem itself, the AMPL presolver has an impact on the group structure. This raises an interesting question for future research: determining the exact reformulation of P yielding the formulation group with tightest associated SBCs (a meaningful simplification might call for the reformulation yielding the largest formulation group). An equally interesting question is that of deciding whether a given problem instance has a formulation whose group is equal to the solution group.

Critical failures were due to excessive RAM or CPU usage on the part of *nauty*. Non-critical failures, due to GAP excessive RAM requirements, imply that an explicit description of the group structure is missing but the group generators are provided (so it is possible to reformulate the problem nonetheless). Computational times are not reported in Tables 2-4 because a large share of the total CPU time taken to compute the group structure was taken by GAP's `StructureDescription` command. Since this was only necessary to compute the tables, but not to reformulate the instances, CPU times at this stage would not be indicative (the CPU time taken to reformulate the instances is reported in Tables 6-9). Just to give a rough idea, compiling all the tables took 7 days of computation, with a significant fraction of the CPU time being taken by all the `arki-` instances.

MIPLib3 1/2

Instance	G_P
`air03`	$(C_2)^{13}$
`arki001`	S_{48}
`blend2`	S_9
`enigma`	C_2
`fiber`	C_2
`gen`	C_2
`mas74`	$(C_2)^2$
`mas76`	$(C_2)^2$
`misc03`	S_3
`misc06`	$(S_5)^3$
`misc07`	S_3
`mitre`	$(C_2)^7$
`mkc`[a]	$\langle$193 generators$\rangle$
`noswot`	C_2
`p0201`	$(C_2)^2$
`p2756`	$(C_2)^{29}$

[a] GAP RAM failure.

MIPLib3 2/2

Instance	G_P
`qiu`	$C_2 \times S_4$
`rgn`	S_5
`rout`	S_5
`seymour`[d]	$\langle$216 generators$\rangle$
`stein27`	$((C_3)^3 \ltimes PSL(3,3)) \ltimes C_2$
`swath`[d,]	$\langle$461 generators$\rangle$
All other	1

MIPLib2003∖MIPLib3

Instance	G_P
`glass4`	C_2
`mzzv11`	$(C_2)^{155}$
`mzzv42z`	$(C_2)^{110}$
`opt1217`	C_2
`protfold`	$(C_2)^2$
`timtab1`	C_2
`timtab2`	C_2
All other	1

Table 2 MILP instances and formulation groups. The group labelled $PSL(3,3)$ is the *projective special linear group* of order 3 on $\mathbb{F}_3$.

It is worth mentioning (thanks to one of the referees for pointing this out) that the `stein45` instance in MIPLib3 has a trivial symmetry group due to an input error of its contributor J. Gregory, as verified by our code. The real Steiner triple incidence matrix actually has significant symmetry.

8.4 Results tables

In this section we present comprehensive results tables. Their purpose is to show that breaking symmetry in general helps, specially on NLP/MINLPs. As explained above, we compare the performance of various BB algorithms solving the original problem and two types of narrowings (*Narrowing1*, adjoining SBCs (5) for the longest orbit; and *Narrowing2*, adjoining the SBCs output by Alg. 2).

The kind of pattern we notice from the first round of tests (Tables 6-9 — symmetric instances solved with a 2h CPU time limit) is twofold. Firstly, more instances are solved faster

GlobalLib 1/2

Instance	G_P
arki0002	$(S_6)^2$
arki0003	C_2
arki0008	S_{50}
arki0009	$(S_5)^{10} \times S_9 \times S_{11}$
arki0010	$(S_5)^5 \times S_9 \times S_{11}$
arki0011	$(C_2)^{15} \times S_3 \times (S_9)^3 \times S_{20}$
arki0012	$(C_2)^{15} \times S_3 \times (S_9)^3 \times S_{11}$
arki0013	$(C_2)^{15} \times S_3 \times (S_9)^3 \times S_{20}$
arki0014	$(C_2)^{15} \times S_3 \times (S_9)^3 \times S_{20}$
arki0016	S_5
elec100	S_3
elec25	S_3
elec50	S_3
ex14_1_5	S_4
ex2_1_3	C_2
ex5_2_5	S_3
ex6_1_1	C_2
ex6_1_3	C_2
ex6_2_10	C_2
ex6_2_12	C_2
ex6_2_13	C_2
ex6_2_14	C_2
ex6_2_5	C_2
ex6_2_7	S_3
ex6_2_9	C_2
ex8_1_6	C_2
ex8_2_1	$(S_4)^4 \times S_8$
ex8_2_2[a]	$\langle$465 generators$\rangle$
ex8_2_4	$(S_4)^4 \times S_8$

[a] GAP RAM failure.

GlobalLib 2/2

Instance	G_P
ex8_2_5[a]	$\langle$602 generators$\rangle$
ex8_3_10	S_5
ex8_3_11	S_5
ex8_3_12	S_5
ex8_3_13	S_5
ex8_3_14	S_5
ex8_3_1	S_5
ex8_3_2	S_5
ex8_3_3	S_5
ex8_3_4	S_5
ex8_3_5	S_5
ex8_3_6	S_5
ex8_3_7	S_5
ex8_3_8	S_5
ex8_3_9	S_5
ex8_4_6	S_3
ex9_1_10	C_2
ex9_1_8	C_2
ex9_2_6	$C_2 \times D_8$
ganges	$(C_2)^6 \times (S_3)^2$
gangesx	$(C_2)^6 \times (S_3)^2$
korcge	$(C_2)^2$
maxmin	C_2
st_e18	$(C_2)^2$
st_e39	$(C_2)^2$
st_fp3	$(C_2)^2$
st_rv9	$(C_2)^{10}$
torsion50	C_2
turkey	$(C_2)^4$
All other	1

[a] GAP RAM failure.

Table 3 NLP instances and formulation groups.

MINLPLib 1/2

Instance	G_P
cecil_13	$(C_2)^{30}$
deb7	S_{10}
deb8	S_{10}
deb9	S_{10}
elf	S_3
gastrans	$(C_2)^2$
gear	D_8
gear2	D_8
gear3	D_8
gear4	D_8
hmittelman	C_2
lop97ic	$(C_2)^2$
lop97icx	$(C_2)^7 \times S_{762}$
nuclear14	S_6
nuclear24	S_6
nuclear25	S_5
nuclear49	S_7

MINLPLib 2/2

Instance	G_P
nuclearva	S_3
nuclearvb	S_3
nuclearvc	S_3
nuclearvd	S_3
nuclearve	S_3
nuclearvf	S_3
nvs09	S_{10}
product[a]	$\langle$150 generators$\rangle$
product2	$\langle$561 generators$\rangle$
risk2b	$(C_2)^5 \times (S_3)^{11} \times S_5 \times (S_6)^2 \times (S_{13})^3$
super1	$(C_2)^8 \times (S_3)^4$
super2	$(C_2)^{10} \times (S_3)^2$
super3	$(C_2)^9 \times (S_3)^2$
super3t	$(C_2)^9 \times (S_3)^2$
synheat	S_4
All other	1

[a] GAP RAM failure.

Table 4 MINLP instances and formulation groups.

Library	*Instances*
MIPLib3	-
MIPLib2003	`ds`, `momentum3`, `msc98-ip`, `sp97ar`, `stp3d`
GlobalLib	`arki0005`, `arki0006`, `arki0007`, `arki0018`, `arki0023`, `arki0024`, `elec200`, `ex8_2_3`, `jbearing100`, `minsurf100`, `torsion100`
MINLPLib	`detf1`, `dosemin2d`, `dosemin3d`, `eg_all_s`, `eg_disc_s`, `eg_disc2_s`, `eg_int_s`, `mbtd`, `nuclear104`, `nuclear10b`, `qap`

Table 5 Excessively large instances (*nauty* RAM or CPU failures during reformulation).

in the narrowing SBC reformulations than in the original problem. Secondly, whereas those instances that are solved faster without SBCs only scrape off a minor CPU time advantage, those that are solved faster with SBCs often have a marked CPU time advantage (or, if not run to completion, a noticeable optimality gap or total/unexplored nodes ratio advantage). For MILPs and *Narrowing1*, for example (see Table 6), the cumulated CPU time advantage in favour of the original problem is 275s, whereas that in favour of the SBC narrowing is 9861s. The trend seems to be that the beneficial effect of SBCs is mainly felt for medium to large-sized instances with long BB runs. Even though the optimal solution is often found later on in the BB run when solving SBC narrowings, the BB tree explorations are in general shorter. For those instances not solved to optimality, the ratios of total/unexplored nodes at termination are often larger (fewer unexplored nodes) and the open optimality gaps often smaller. This is what promtped us to run a second round of tests with no time limit for some selected difficult instances (see Table 11), on which these effects are even more remarkable.

Table 6 refers to MILPs (MIPLib3 and MIPLib2003), Table 7 refers to NLPs (GlobalLib) and Table 9 refers to MINLPs (MINLPLib). All tables have the same core structure recording the following indicators at termination:

1. incumbent value (f^*)
2. seconds of user CPU time (*CPU*)
3. open gap (*gap* — we use the CPLEX definition $\left(\frac{100|f^*-\bar{f}|}{|f^*+10^{-10}|}\right)\%$, where f^* is the objective function value of the incumbent and $\bar{f}$ is the best overall lower bound)
4. total nodes (*nodes*)
5. unexplored nodes (*tree*)

for the original problem and each SBC narrowing. The last column (*R.t.*) contains the reformulation time expressed as seconds of user CPU time taken to reformulate the instance (this refers to *Narrowing1*; the values for *Narrowing2* are practically identical, the bottleneck being the computation of the group structure by *nauty*). Tables 7 and 9 also have a column (*Slv*) which indicates the solver name: "C" stands for COUENNE [4], and "B" for BARON [47]. Although COUENNE was our NLP/MINLP global solver of choice, because of its relatively young age it still shows some rough spots, which sometimes hamper the solution process. COUENNE failed on all instances whose results in the table are marked BARON. NLP and MINLP instances where both solvers failed are recorded in Tables 8 and 10: all these are well known to be difficult instances, and most of them are very large in size. We remark that for many of them the reason for failure was the absence of meaningful variable ranges, which makes the construction of the lower bounding problem inherently difficult. In all tables, data marked in boldface signals an advantage: in general, lower values for incumbent, CPU times, open gap, total and unexplored nodes at termination are considered an advantage. However, for those instances not solved to optimality within the 2h CPU time limit, the higher values of the ratios *total/unexplored nodes* marks an advantage (meaning that more of the tree has been explored in the allotted time).

	Original problem			Narrowing1			Narrowing2			
Instance	*CPU*	f^* *gap*	*nodes* *tree*	*CPU*	f^* *gap*	*nodes* *tree*	*CPU*	f^* *gap*	*nodes* *tree*	*R.t.*
MIPLib3										
`air03`	1.14	340160 0%	0 0	1.10	340160 0%	0 0	**0.98**	340160 0%	0 0	161.94
`arki001`	114.31	7.58e+6 0%	93340 0	125.03	7.58e+6 0%	93340 0	**108.5**	7.58e+6 0%	93340 0	78.56
`blend2`	0.94	7.59 0%	**957** 0	**0.87**	7.59 0%	969 0	0.91	7.59 0%	967 0	1.43
`enigma`	**0.00**	0 0%	**0** 0	0.05	0 0%	153 0	0.05	0 0%	153 0	1.22
`fiber`	**0.26**	4.05e+5 0%	**71** 0	0.27	4.05e+5 0%	81 0	**0.26**	4.05e+5 0%	81 0	4.96
`gen`	0.04	1.12e+5 0%	0 0	0.04	1.12e+5 0%	0 0	0.04	1.12e+5 0%	0 0	2.57
`mas74`	529	1.18e+4 0%	**2405600** 0	466	1.18e+4 0%	2558400 0	**426**	1.18e+4 0%	2558400 0	1.41
`mas76`	**43.42**	4e+4 0%	305500 0	43.62	4e+4 0%	305500 0	**41.97**	4e+4 0%	305500 0	1.41
`misc03`	0.30	3360 0%	**160** 0	**0.29**	3360 0%	656 0	1.07	3360 0%	700 0	1.35
`misc06`	0.13	0 0%	17 0	0.13	0 0%	17 0	0.13	0 0%	17 0	11.63
`misc07`	18.41	2810 0%	16211 0	**12.72**	2810 0%	**12317** 0	21.28	2810 0%	**20395** 0	1.57
`mitre`	0.83	115155 0%	0 0	**0.80**	115155 0%	0 0	0.82	115155 0%	0 0	1304.41
`mkc`	7200	-563.732 0.17%	**156803** **75392**	7200	**-563.846** **0.15**%	136200 36654	-	-	-	2712.33
`noswot`	7200	-41 4.88%	8629594 1581600	7200	-41 **3.56**%	**7852302** **817466**	7200	-41 **3.56**%	**7852302** **817466**	1.27
`p0201`	**0.24**	7615 0%	**103** 0	0.35	7615 0%	295 0	-	-	-	3.44
`p2756`	0.40	3124 0%	11 0	0.39	3124 0%	11 0	**0.37**	3124 0%	11 0	25.61
`qiu`	45.01	-132.87 0%	8375 0	**36.38**	-132.87 0%	**5500** 0	-	-	-	6.22
`rgn`	0.13	82.2 0%	561 0	0.15	82.2 0%	555 0	**0.12**	82.2 0%	**505** 0	1.36
`rout`	**11.99**	1077.21 0%	**5800** 0	15.28	1077.56 0%	6700 0	53.86	1077.56 0%	17700 0	2.39
`seymour`	7200	423 1.58%	**103097** **84949**	7200	423 1.58%	101576 83701	7200	423 **1.57**%	105481 86941	5.95
`stein27`	0.27	18 0%	1582 0	**0.07**	18 0%	**637** 0	-	-	-	1.30
`swath`[a]	1534	492.45 14.60%	215100 200293	1773	486.18 **13.83**%	**194400** **165017**	1550	**484.09** 17.55%	196200 182289	325.10
MIPLib2003∖MIPLib3										
`glass4`	7200	1.4e+9 21.43%	2240022 944291	**1180.69**	**1.2e+9** **0**%	**392600** **0**	1186.57	**1.2e+9** **0**%	**392600** **0**	1.62
`mzzv11`	**112.87**	-21718 0%	**543** 0	129.35	-21718 0%	588 0	161	-21718 0%	588 0	213.76
`mzzv42z`	**40.94**	-20540 0%	237 0	46.42	-20540 0%	**223** 0	59.76	-20540 0%	**223** 0	244.76
`opt1217`	**0.09**	-16 0%	0 0	**0.09**	-16 0%	0 0	0.10	-16 0%	0 0	1.37
`protfold`	7200	-19 **90.71**%	**12936** **11563**	7200	-19 91.58%	14050 12760	-	-	-	592.14
`timtab1`	5317.42	7.64e+5 0%	3576200 0	**1554.20**	7.64e+5 0%	**973700** 0	1555.74	7.64e+5 0%	**973700** 0	1.37
`timtab2`	7200	**1.12e+6** **31.74**%	1115428 745016	7200	1.14e+6 33.23%	1002113 647515	7200	1.14e+6 33.32%	**959893** **619402**	1.38

[a] Termination on out of memory.

Table 6 MILP results (MIPLib3 and MIPLib2003 solved by CPLEX 11). Lower values are best in general; in instances not solved to optimality (CPU=7200), higher ratios *nodes/tree* are best. Values marked '-' denote Narrowing2=Narrowing1.

Some of the results for MILPs are very encouraging: the `glass4` instance, for example, is known to be a hard one: [24] reports solving this instance using XPRESS 2006B in 7 hours on a 2 processor Xeon system with the following settings: no cuts, best first node selection, heavy strong branching, and variable selection based on up/down pseudocosts. Although it is hard to compare with our results, what with the solver, hardware and version date difference, solving it in less than 20 minutes on a default configuration is worthy of note; even more so considering that the symmetry group is only C_2. The `timtab1` instance solution time is reduced to less than a third by adjoining the SBCs. It is interesting that `arki001`, `mas76` and `p2756` have the same number of nodes to completion but different CPU times. The CPLEX output log files of original/reformulated instances being equal for all but the partial CPU times, the only cause of this difference lies in the LP being solved at each node: although most of the times an LP with more constraints takes more time to solve, CPLEX employs several preprocessing techniques which might exploit the SBCs present in the reformulation (but absent in the original formulation) to yield the observed improvements.

On average, with a 2h user CPU time limit, it is slightly more advantageous to solve an SBC narrowing than the original problem. We reported total user CPU time, number of times the solution yielded best optima in the set (*Best*), and total number of BB nodes. The total closed gap averaged over original problem and *Narrowing1* and *Narrowing2* reformulations is 22661.35% with a standard deviation of 0.14, which effectively means that within the 2h CPU time limit, symmetry breaking had no effect with respect to the closed gap (without the 2h limit the story is different, see Table 11). It appears evident that, on average, breaking symmetry is beneficial when using BB-type solution algorithms.

The results referring to the second round of tests, involving selected (difficult) instances solved *without* the 2h user CPU time limit, are in Table 11. As before, data marked in boldface signals an advantage. The most meaningful indicators at termination are:

1. the objective function value of the incumbent (the lower the better);
2. the open gap (the lower the better);
3. the amount of explored nodes per second, i.e. $\frac{nodes}{tree \times CPU}$ (the higher the better).

With extended CPU time limits, *Narrowing2* provides a significant computational advantage over the original problem, and a slight advantage over *Narrowing1*.

We remark also that results worthy of note were obtained on the `protfold` (open gap reduced by almost half) and `seymour` (given the problem structure, even a minor reduction in open gap is impressive) MILP instances.

It appears that adding SBCs sometimes has an adverse effect (albeit slighter than the beneficial observed effect). This occurrence may be explained by any one of the following facts: (a) SBCs have an element of arbitrary user choice in them, e.g. the natural variable index order 1,2,3,... (constraints enforcing other orders would also be valid); (b) SBCs may change branching decisions; (c) best choices for breaking symmetries may change during the BB tree exploration, locally to each node (it might be advantageous to change narrowing at select nodes rather than just at the root node). These issues will hopefully be addressed in future works (in particular, it might be a good idea to employ orbital branching [43, 44] instead of a static narrowing as a symmetry-breaking device).

9 Conclusion

This paper discusses methods for automatically exploiting symmetries in MILPs, nonconvex NLPs and MINLPs. We construct the formulation group, then derive static Symmetry-Breaking Constraints from its generators, and finally reformulate the given problem to a narrowing where some of the symmetric solutions are likely to be infeasible. The reformulated problem can then be solved by standard Branch-and-Bound solvers such as CPLEX

		Original problem			Narrowing1			Narrowing2			
Instance	*Slv*	*CPU*	f^* *gap*	*nodes* *tree*	*CPU*	f^* *gap*	*nodes* *tree*	*CPU*	f^* *gap*	*nodes* *tree*	*R.t.*
ex14_1_5	C	0.018	0 0%	0 0	**0.013**	0 0%	0 0	0.020	0 0%	0 0	1.44
ex2_1_3	C	0.013	-15 0%	0 0	**0.010**	-15 0%	0 0	0.018	-15 0%	0 0	1.41
ex5_2_5	C	7200	-3500 0.32%	2040274 503850	7200	-3500 0.27%	**63595** **18733**	7200	-3500 **0.22**%	1580366 402117	1.40
ex6_1_1	C	61	-2.02e-2 0%	14226 0	**37**	-2.02e-02 0%	**10027** 0	45	-2.02e-02 0%	**10027** 0	1.43
ex6_1_3	C	135	-3.53e-01 0%	13660 0	**97**	-3.53e-01 0%	**2659** 0	111	-3.53e-01 0%	**2659** 0	1.40
ex6_2_10[a]	C	4754	-3.052 0.37%	45200 19397	7200	-3.052 **0.01**%	**88300** **16228**	7200	-3.052 0.014%	81700 15898	1.40
ex6_2_12	C	205	2.89e-01 0%	15827 0	**85**	2.89e-01 0%	**3477** 0	**85**	2.89e-01 0%	**3477** 0	1.41
ex6_2_13	C	7200	-2.16e-01 0.09%	65461 27773	7200	-2.16e-01 **0.03**%	**77569** **29202**	7200	-2.16e-01 **0.03**%	75670 28580	1.38
ex6_2_14	C	29	-6.96e-01 0%	195 0	19	-6.96e-01 0%	**92** 0	**17.8**	-6.96e-01 0%	**92** 0	1.37
ex6_2_5[a]	C	3017	-70.75 6.85%	**43500** **18094**	4920	-70.75 **0.90**%	68600 27187	5240	-70.75 **0.90**%	68600 27187	1.43
ex6_2_7[a]	C	1874	-0.161 48.35%	32500 10900	1884	-0.161 39.92%	**17500** **7894**	1323	-0.161 **32.07**%	16600 8008	1.42
ex6_2_9[b]	C	699	-3.51e-02 0%	28711 0	**184**	-3.46e-02 0%	**8522** 0	191	-3.46e-02 0%	**8522** 0	1.42
ex8_1_6	C	0.03	-5.065 0%	0 0	0.03	-5.065 0%	0 0	0.046	-5.065 0%	0 0	1.36
ex8_3_1[c]	B	7200	0 *-10*	20245 12299	7200	0 *-10*	**14191** **8099**	-	-	-	2.27
ex8_3_2	B	7200	-0.4123 2325%	9471 6566	7200	-0.4123 2325%	**8004** **5445**	-	-	-	2.05
ex8_3_3	B	7200	-0.4166 2393%	**9205** **5770**	7200	-0.4166 2393%	7416 5065	-	-	-	1.36
ex8_3_4	B	7200	-3.58 2695%	6597 4484	7200	-3.58 2695%	**4985** **3347**	-	-	-	2.10
ex8_3_5	B	7200	**-0.069** **14371**%	**7843** **4727**	7200	-0.068 14434%	8470 5597	-	-	-	2.07
ex8_3_11[c]	B	7200	0 *-10*	**15197** **8393**	7200	0 *-10*	21532 13344	-	-	-	1.51
ex8_3_12[c]	B	7200	0 *-10*	**21881** **12515**	7200	0 *-10*	20566 13000	-	-	-	1.37
ex8_3_13[c]	B	7200	0 *-10*	13662 9015	7200	0 *-10*	**11038** **7179**	-	-	-	1.91
ex8_4_6	C	**0.08**	0.66 0%	0 0	1.53	0.66 0%	0 0	0.13	0.66 0%	0 0	1.40
ex9_1_10	C	**4.68**	-3.25 0%	0 0	9.3	-3.25 0%	0 0	9.3	-3.25 0%	0 0	1.77
ex9_1_8	C	**4.7**	-3.25 0%	0 0	9.3	-3.25 0%	0 0	9.3	-3.25 0%	0 0	1.43
ex9_2_2	C	0.16	99.99 0%	2 0	**0.15**	99.99 0%	2 0	0.17	99.99 0%	2 0	2.31
ex9_2_6	C	0.1	-1 0%	0 0	0.1	-1 0%	0 0	-	-	-	1.38
maxmin	B	7200	-0.366 157.55%	31655 20122	7200	-0.366 157.38%	**28973** **18300**	7200	-0.366 **156.87**%	31117 19659	1.29
st_e18	C	0.01	-2.83 0%	0 0	0.01	-2.83 0%	0 0	0.01	-2.83 0%	0 0	1.38
st_e39	C	**0.03**	-5.065 0%	0 0	**0.03**	-5.065 0%	0 0	0.04	-5.065 0%	0 0	1.41
st_fp3	C	**0.015**	-15 0%	0 0	**0.015**	-15 0%	0 0	0.017	-15 0%	0 0	2.02
st_rv9	C	10.3	-120.15 0%	214 0	9.5	-120.15 0%	**208** 0	**9.3**	-120.15 0%	**208** 0	1.44
turkey	C	**2749**	-28371.6 0%	**97** 0	3724	-28371.6 0%	125 0	3831	-28371.6 0%	125 0	7.24

[a] *CPU* < 7200 and *gap*$>$ 0% because of COUENNE's segmentation fault during computation.

[b] Some AMPL warnings might be the cause of the objective function value discrepancy (both were certified optimal by COUENNE).

[c] When $f^* = 0$ the open gap is (almost) ill defined, thus the value of the best LP bound is reported instead.

Table 7 NLP results (GlobalLib solved by COUENNE or BARON). Lower values are best in general; in instances not solved to optimality (CPU=7200), higher ratios *nodes/tree* are best. Values marked '-' denote Narrowing2=Narrowing1.

(for linear problems) and COUENNE (for nonlinear problems). We exhibit computational results validating the approach.

Instance	*R.t.*	*Instance*	*R.t.*	*Instance*	*R.t.*	*Instance*	*R.t.*
arki0002	82.36	arki0012	5400.03	elec25	5.32	ex8_3_6	1.45
arki0003	10813	arki0013	5268.54	elec50	228.67	ex8_3_7	1.62
arki0008	92.15	arki0014	5695.04	ex8_2_1	1.43	ex8_3_9	1.41
arki0009	401.10	arki0016	142.34	ex8_2_2	21080	ex8_3_10	1.28
arki0010	65.46	elec100	13082.36	ex8_2_4	1.64	ex8_3_14	1.48
arki0011	5715.02			ex8_2_5	17172	torsion50	6122

Table 8 NLP instances where both COUENNE and BARON failed.

		Original problem			Narrowing1			Narrowing2			
Instance	*Slv*	*CPU*	f^* *gap*	*nodes* *tree*	*CPU*	f^* *gap*	*nodes* *tree*	*CPU*	f^* *gap*	*nodes* *tree*	*R.t.*
cecil_13	C	**6181**	-1.14e+5 0%	106074 0	6248	-1.14e+5 0%	106074 0	7200	-1.14e+5 0%	101138 2084↓	2.64
elf	B	11.86	0.1916 0%	341 0	7.35	0.1916 0%	326 0	**2.9**	0.1916 0%	**87** 0	1.43
gastrans	C	9	89.1 0%	227 0	**5.2**	89.1 0%	**109** 0	5.7	89.1 0%	**109** 0	1.39
gear	C	0.08	0 0%	8 0	**0.01**	0 0%	**0** 0	-	-	-	1.27
gear2	C	**0.34**	0 0%	**6** 0	0.51	0 0%	21 0	-	-	-	1.38
gear3	C	**0.14**	0 0%	26 0	0.19	0 0%	**25** 0	-	-	-	1.30
gear4	B	0.62	1.968 0%	3239 0	**0.48**	1.968 0%	**1739** 0	-	-	-	1.28
hmittelman	C	**0.16**	13 0%	0 0	0.18	13 0%	0 0	0.20	13 0%	0 0	1.25
lop97icx	B	7200	4492.48 40.65%	**9106** **5537**	7200	4493.5 **39.9**%	10146 6296	7200	**4457.55** 40.23%	3254 2026	24.96
nvs09	C	5.1	-43.13 0%	0 0	2.4	-43.13 0%	0 0	**1.7**	-43.13 0%	0 0	1.24
risk2b[a]	C	**13.26**	-55.87 0%	0 0	14.77	-55.87 0%	0 0	14.28	-55.87 0%	0 0	2.48
synheat	B	127	1.5e+5 0%	3316 0	**92**	1.5e+5 0%	**1775** 0	566	1.5e+5 0%	9509 0	1.27

[a] This instance is unbounded [33], so the objective function value is not a meaningful indicator.

Table 9 MINLP results (MINLPLib solved by COUENNE or BARON). Lower values are best in general; in instances not solved to optimality (CPU=7200), higher ratios *nodes/tree* are best. Values marked '-' denote Narrowing2=Narrowing1.

Instance	*R.t.*	*Instance*	*R.t.*	*Instance*	*R.t.*	*Instance*	*R.t.*
deb7	26.0	nuclear24	15.22	nuclearvc	1.88	product2	292.39
deb8	26.1	nuclear25	19.46	nuclearvd	2.26	super1	10.12
deb9	25.9	nuclear49	457.88	nuclearve	2.03	super2	10.60
lop97ic	202.85	nuclearva	1.83	nuclearvf	2.03	super3	10.15
nuclear14	15.49	nuclearvb	1.73	product	13.88	super3t	6.09

Table 10 MINLP instances where both COUENNE and BARON failed (the `deb` instances are reported infeasible).

Symmetry-Breaking Constraints practically help finding exact optima by Branch-and-Bound algorithms: in general, the more symmetry-breaking constraints we adjoin to the original formulation, the fewer nodes we might hope the BB search tree will have. Furthermore, these constraints are generated by the nontrivial orbits of the formulation group action on the set of variable indices. Therefore, in general, the larger the formulation group, the better. Since different exact reformulations of the same problem often yield different formulation groups (all of which are subgroups of the solution group associated to the problem), a very interesting question for future research is that of looking for the exact reformulation maximizing the number (and length) of nontrivial orbits. It must be said, however, that our symmetry-breaking constraints are rather general-purpose, hence they undergo the usual trade-off between generality and efficiency. This suggests that breaking symmetries at the modelling level (*static* symmetry breaking) should also be complemented by breaking sym-

		Original problem			Narrowing1			Narrowing2			
Instance	*Slv*	*CPU*	f^* *gap*	*nodes* *tree*	*CPU*	f^* *gap*	*nodes* *tree*	*CPU*	f^* *gap*	*nodes* *tree*	*R.t.*
MILPLib(s)											
`mkc`[a]		146850	-563.846 0.13%	1945500 1479080	133924	-563.846 0.13%	**2104500** **1449867**	-	-	-	2712.33
`protfold`[b]		300000	-26 30.51%	592000 458813	300000	**-29** **16.54%**	**536100** **353823**	-	-	-	592.14
`seymour`[a]		262817	423 0.9%	3992700 3026077	283311	423 **0.83%**	**4343500** **3038821**	233643	423 1.0%	3960700 3064665	5.95
GlobalLib											
`ex5_2_5`[a]	C	19805	-3500 28.14%	5452500 1259853	82320	-3500 18.5%	**7373700** **747262**	18151	-3500 **17.41%**	4425400 1076927	1.40
`maxmin`[a]	B	58643	-0.366 145%	237100 150803	57762	-0.366 **144%**	**238000** **150355**	-	-	-	1.29
MINLPLib											
`lop97icx`[a]	B	26903	**4391.1** 38.2%	**44858** **27824**	30772	4493.5 39.14%	43948 27189	42926	4412.9 **37.97%**	23416 14708	24.96

[a] Termination on out of memory.

[b] Termination after 5000 minutes

Table 11 Some results without the 2h CPU time limit. Lower values are best in general; in instances not solved to optimality, higher ratios *nodes/(tree×CPU)* are best.

metries at the branching level of the BB algorithm (*dynamic* symmetry breaking). This will make the object of further investigations.

The tabulation of the formulation groups for all instances in the best known mathematical programming libraries suggests that although symmetry is not all-encompassing, it is nonetheless pervasive enough to merit more attention than is currently attributed to it by the mathematical programming community. Current efforts are limited to Mixed-Integer Linear and Semidefinite Programming only (this is the first work reaching into Mixed-Integer Nonlinear Programming) and often assume prior knowledge of (subgroups of) the solution group. If efficient symmetry detection and breaking devices are to make their way into mainstream MINLP solvers, more techniques are needed to address the issues arising in treating symmetry in mathematical programming.

Acknowledgements

I wish to thank François Margot for many useful discussions and suggestions, and for carefully checking the theoretical part of this paper, and two anonymous referees for insightful comments. Financial support by ANR under grant 07-JCJC-0151, by EU under grant NEST "Morphex" and by the Digiteo RMNCCO Chair is gratefully acknowledged.

References

1. F.A. Al-Khayyal and H.D. Sherali. On finitely terminating branch-and-bound algorithms for some global optimization problems. *SIAM Journal of Optimization*, 10(4):1049–1057, 2000.
2. L. Babai. Automorphism groups, isomorphism, reconstruction. In R. Graham, M. Grötschel, and L. Lovász, editors, *Handbook of Combinatorics, vol. 2*, pages 1447–1540. MIT Press, Cambridge, MA, 1996.
3. C. Bauer, A. Frink, and R. Kreckel. Introduction to the GiNaC framework for symbolic computation within the C++ programming language. *Journal of Symbolic Computation*, 33(1):1–12, 2002.
4. P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter. Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods and Software*, 24(4):597–634, 2009.

5. T. Berthold and M. Pfetsch. Detecting orbitopal symmetries. In B. Fleischmann, K.-H. Borgwardt, R. Klein, and A. Tuma, editors, *Operations Research Proceedings 2008*, pages 433–438, Berlin, 2009. Springer.
6. R. Bixby, S. Ceria, C. McZeal, and M. Savelsbergh. An updated mixed integer programming library: Miplib 3. Technical Report TR98-03, Rice University, 1998.
7. K. Booth and C. Colbourn. Problems polynomially equivalent to graph isomorphism. Technical Report CS-77-04, University of Waterloo, 1979.
8. M. Boulle. Compact mathematical formulation for graph partitioning. *Optimization and Engineering*, 5:315–333, 2004.
9. M. Bruglieri and L. Liberti. Optimal running and planning of a biomass-based energy production process. *Energy Policy*, 36:2430–2438, 2008.
10. M. Bussieck. Globallib — a collection of nonlinear programming problems, 2004. (`http://www.gamsworld.org/global/globallib.htm`).
11. M. Bussieck, A. Drud, and A. Meeraus. MINLPLib — A collection of test models for mixed-integer nonlinear programming. *INFORMS Journal on Computing*, 15(1), 2003.
12. G. Butler. *Fundamental Algorithms for Permutation Groups*, volume 559 of *LNCS*. Springer, 1991.
13. D. Cohen, P. Jeavons, C. Jefferson, K. Petrie, and B. Smith. Symmetry definitions for constraint satisfaction problems. In P. van Beek, editor, *CP*, volume 3709 of *LNCS*. Springer, 2005.
14. J.S. Cohen. *Computer Algebra and Symbolic Computation: Mathematical Methods*. AK Peters, Natick, Massachusetts, 2000.
15. J.S. Cohen. *Computer Algebra and Symbolic Computation: Elementary Algorithms*. AK Peters, Natick, Massachusetts, 2002.
16. Y. Faenza and V. Kaibel. Extended formulations for packing and partitioning orbitopes. *Mathematics of Operations Research*, 34(3):686–697, 2009.
17. M. Fischetti and D. Salvagnin. A local dominance procedure for mixed-integer linear programming. Technical report, ARRIVAL project, 2007.
18. R. Fourer and D. Gay. *The AMPL Book*. Duxbury Press, Pacific Grove, 2002.
19. E.J. Friedman. Fundamental domains for integer programs with symmetries. In A. Dress, Y. Xu, and B. Zhu, editors, *COCOA Proceedings*, volume 4616 of *LNCS*, pages 146–153. Springer, 2007.
20. The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.4.10*, 2007.
21. M. Hall. *Theory of Groups*. Chelsea Publishing Company, New York, 2nd edition, 1976.
22. ILOG. *ILOG CPLEX 11.0 User's Manual*. ILOG S.A., Gentilly, France, 2008.
23. V. Kaibel and M. Pfetsch. Packing and partitioning orbitopes. *Mathematical Programming*, 114(1):1–36, 2008.
24. R. Laundy, M. Perregaard, G. Tavares, H. Tipi, and A. Vazacopoulos. Solving hard mixed-integer programming problems with Xpress-MP: A MIPLIB 2003 case study. *INFORMS Journal on Computing*, 21(2):304–313, 2009.
25. J. Lee and F. Margot. On a binary-encoded ILP coloring formulation. *INFORMS Journal on Computing*, 19(3):406–415, 2007.
26. S. Leyffer. MacMINLP — AMPL collection of mixed integer nonlinear programs, 2000. (`http://www.mcs.anl.gov/~leyffer/macminlp/`).
27. L. Liberti. Framework for symbolic computation in C++ using n-ary trees. Technical report, CPSE, Imperial College London, 2001.
28. L. Liberti. Writing global optimization software. In L. Liberti and N. Maculan, editors, *Global Optimization: from Theory to Implementation*, pages 211–262. Springer, Berlin, 2006.
29. L. Liberti. Automatic generation of symmetry-breaking constraints. In B. Yang, D.-Z. Du, and C.A. Wang, editors, *COCOA Proceedings*, volume 5165 of *LNCS*, pages 328–338, Berlin, 2008. Springer.
30. L. Liberti. Reformulations in mathematical programming: Symmetry. Technical Report 2165, Optimization Online, 2008.
31. L. Liberti. Reformulations in mathematical programming: Definitions and systematics. *RAIRO-RO*, 43(1):55–86, 2009.
32. L. Liberti, S. Cafieri, and F. Tarissan. Reformulations in mathematical programming: A computational approach. In A. Abraham, A.-E. Hassanien, P. Siarry, and A. Engelbrecht, editors, *Foundations of Computational Intelligence Vol. 3*, number 203 in Studies in Computational Intelligence, pages 153–234. Springer, Berlin, 2009.
33. L. Liberti, N. Mladenović, and G. Nannicini. A good recipe for solving MINLPs. In V. Maniezzo, T. Stützle, and S. Voß, editors, *Hybridizing metaheuristics and mathematical programming*, volume 10 of *Annals of Information Systems*, pages 231–244, New York, 2009. Springer.
34. F. Margot. Pruning by isomorphism in branch-and-cut. *Mathematical Programming*, 94:71–90, 2002.
35. F. Margot. Exploiting orbits in symmetric ILP. *Mathematical Programming B*, 98:3–21, 2003.
36. F. Margot. Small covering designs by branch-and-cut. *Mathematical Programming B*, 94:207–220, 2003.

37. F. Margot. Symmetric ILP: coloring and small integers. *Discrete Optimization*, 4:40–62, 2007.
38. F. Margot. Symmetry in integer linear programming. In M. Jünger, T. Liebling, D. Naddef, G. Nemhauser, W. Pulleyblank, G. Reinelt, G. Rinaldi, and L. Wolsey, editors, *50 Years of Integer Programming*, pages 647–681. Springer, Berlin, 2010.
39. A. Martin, T. Achterberg, and T. Koch. Miplib 2003. Technical Report 05-28, ZIB, 2005.
40. B. McKay. Practical graph isomorphism. *Congressus Numerantium*, 30:45–87, 1981.
41. B. McKay. *nauty User's Guide (Version 2.4)*. Computer Science Dept. , Australian National University, 2007.
42. A. Neumaier. Complete search in continuous global optimization and constraint satisfaction. *Acta Numerica*, 13:271–369, 2004.
43. J. Ostrowski, J. Linderoth, F. Rossi, and S. Smriglio. Orbital branching. In M. Fischetti and D.P. Williamson, editors, *IPCO*, volume 4513 of *LNCS*, pages 104–118. Springer, 2007.
44. J. Ostrowski, J. Linderoth, F. Rossi, and S. Smriglio. Constraint orbital branching. In A. Lodi, A. Panconesi, and G. Rinaldi, editors, *IPCO*, volume 5035 of *LNCS*, pages 225–239. Springer, 2008.
45. A. Ramani and I. Markov. Automatically exploiting symmetries in constraint programming. In B. Faltings, A. Petcu, F. Fages, and F. Rossi, editors, *Constraint Solving and Constraint Logic Programming*, volume 3419 of *LNAI*, pages 98–112, Berlin, 2005. Springer.
46. K.H. Rosen, editor. *Handbook of Discrete and Combinatorial Mathematics*. CRC Press, New York, 2000.
47. N.V. Sahinidis and M. Tawarmalani. *BARON 7.2.5: Global Optimization of Mixed-Integer Nonlinear Programs,* User's Manual, 2005.
48. H. Schichl and A. Neumaier. Interval analysis on directed acyclic graphs for global optimization. *Journal of Global Optimization*, 33(4):541–562, 2005.
49. A. Seress. *Permutation Group Algorithms*. Cambridge University Press, Cambridge, 2003.
50. H. Sherali and C. Smith. Improving discrete model representations via symmetry considerations. *Management Science*, 47(10):1396–1407, 2001.
51. R. Uehara, S. Toda, and T. Nagoya. Graph isomorphism completeness for chordal bipartite graphs and strongly chordal graphs. *Discrete Applied Mathematics*, 145:479–482, 2005.
52. F. Vallentin. Symmetry in semidefinite programs. Technical Report 1702, Optimization Online, 2007.
53. H. Wielandt. *Finite permutation groups*. Academic Press, New York, 1964.
54. L.A. Wolsey. *Integer Programming*. Wiley, New York, 1998.
55. W. Zhu. Unsolvability of some optimization problems. *Applied Mathematics and Computation*, 174:921–926, 2006.

Online appendix: Instance size tables

MIPLib3 1/2

Instance	*n*	*Bin.*	*Int.*	*m*
10teams	2025	1800	0	230
air03	10757	10757	0	124
air04	8904	8904	0	823
air05	7195	7195	0	426
arki001	1388	415	123	1048
bell3a	133	39	32	123
bell5	104	30	28	91
blend2	353	231	33	274
cap6000	6000	6000	0	2176
dano3mip	13873	552	0	3202
danoint	521	56	0	664
dcmulti	548	75	0	290
dsbmip	1877	160	0	1182
egout	141	55	0	98
enigma	100	100	0	21
fiber	1298	1254	0	363
fixnet6	878	378	0	478
flugpl	18	0	11	18
gen	870	144	6	780
gesa2	1224	240	168	1392
gesa2_o	1224	384	336	1248
gesa3	1152	216	168	1368
gesa3_o	1152	336	336	1224
gt2	188	24	164	29
harp2	2993	2993	0	112
khb05250	1350	24	0	101
l152lav	1989	1989	0	97
lseu	89	89	0	28
markshare1	62	50	0	6
markshare2	74	60	0	7
mas74	151	150	0	13
mas76	151	150	0	12
misc03	160	159	0	96
misc06	1808	112	0	820
misc07	260	259	0	212
mitre	10724	10724	0	2054
mkc	5325	5323	0	3411
mod008	319	319	0	6
mod010	2655	2655	0	146
modglob	422	98	0	291
noswot	128	75	25	182
nw04	87482	87482	0	36
p0033	33	33	0	15

MIPLib3 2/2

Instance	*n*	*Bin.*	*Int.*	*m*
p0201	201	201	0	133
p0282	282	282	0	241
p0548	548	548	0	176
p2756	2756	2756	0	755
pk1	86	55	0	45
pp08aCUTS	240	64	0	246
pp08a	240	64	0	136
qnet1	1541	1288	129	503
qnet1_o	1541	1288	129	456
rgn	180	100	0	24
rentacar	9557	55	0	6803
rout	556	300	15	291
set1ch	712	240	0	492
seymour	1372	1372	0	4944
stein27	27	27	0	118
stein45	45	45	0	331
swath	6805	6724	0	884
vpm1	378	168	0	234
vpm2	378	168	0	234

MIPLib2003 $\setminus$ MIPLib3

Instance	*n*	*Bin.*	*Int.*	*m*
a1c1s1	3648	192	0	3312
aflow30a	842	421	0	479
aflow40b	2728	1364	0	1442
atlanta-ip	48738	46667	106	21731
disctom	10000	10000	0	399
glass4	321	302	0	396
liu	1154	1087	0	2178
manna81	3321	18	3303	6480
momentum1	5174	2349	0	42680
momentum2	3732	1808	1	24237
mzzv11	10240	9989	251	9499
mzzv42z	11717	11482	235	10460
net12	14115	1603	0	14021
nsrand-ipx	6621	6620	0	735
opt1217	769	768	0	64
protfold	1835	1835	0	2112
roll3000	1166	246	492	2294
timtab1	397	64	107	171
timtab2	675	113	181	294
tr12-30	1080	360	0	750

Table 12 MILP instance statistics.

GlobalLib 1/3

Instance	*n*	*m*	*NLT*
abel	30	14	76
alkylation	10	11	19
alkyl	14	7	17
arki0001	1030	513	512
arki0002	2456	1976	1976
arki0003	2283	2583	33180
arki0008	5072	5020	17646
arki0009	7714	6707	1299
arki0010	4144	3427	654
arki0011	19161	17430	2979
arki0012	19170	17574	2979
arki0013	19161	17430	2979
arki0014	19265	17525	2979
arki0015	2093	1496	4600
arki0016	5047	2946	7301
arki0017	4332	2572	6312
arki0019	510	2	4693
arki0020	1262	2	11930
arki0021	3187	2	27540
arki0022	4152	2	51502
bayes2_10	86	77	440
bayes2_20	86	77	440
bayes2_30	86	77	440
bayes2_50	86	77	440
bearing	13	12	36
btest14	135	93	551
camcge	279	242	867
camshape100	199	200	396
camshape200	399	400	796
camshape400	799	800	1596
camshape800	1599	1600	3196
catmix100	303	200	1600
catmix200	603	400	3200
catmix400	1203	800	6400
catmix800	2403	1600	12800
chain100	202	101	1400
chain200	402	201	2800
chain25	102	51	700
chain400	802	401	5600
chain50	102	51	700
chakra	62	41	82
chance	4	3	10
chem	11	4	30
chenery	43	38	54
circle	3	10	30
demo7	70	57	24
dispatch	4	2	33
etamac	97	70	79
ex14_1_1	3	4	24
ex14_1_2	6	9	67
ex14_1_3	3	4	8
ex14_1_4	3	4	12
ex14_1_5	6	6	8
ex14_1_6	9	15	34
ex14_1_7	10	17	114
ex14_1_8	3	4	26
ex14_1_9	2	2	14
ex14_2_1	5	7	90
ex14_2_2	4	5	32
ex14_2_3	6	9	192
ex14_2_4	5	7	222
ex14_2_5	4	5	68
ex14_2_6	5	7	250
ex14_2_7	6	9	496
ex14_2_8	4	5	128
ex14_2_9	4	5	164
ex2_1_10	20	10	40
ex2_1_1	5	1	15
ex2_1_2	6	2	10
ex2_1_3	13	9	12
ex2_1_4	6	5	2
ex2_1_5	10	11	21
ex2_1_6	10	5	30
ex2_1_7	20	10	39
ex2_1_8	24	10	48
ex2_1_9	10	1	22
ex3_1_1	8	6	5
ex3_1_2	5	6	40
ex3_1_3	6	6	9
ex3_1_4	3	3	10
ex4_1_1	1	0	9
ex4_1_2	1	0	96
ex4_1_3	1	0	8
ex4_1_4	1	0	5
ex4_1_5	2	0	8
ex4_1_6	1	0	5
ex4_1_7	1	0	5
ex4_1_8	2	1	3
ex4_1_9	2	2	12
ex5_2_2_case1	9	6	4
ex5_2_2_case2	9	6	4
ex5_2_2_case3	9	6	4
ex5_2_4	7	6	14
ex5_2_5	32	19	195
ex5_3_2	22	16	12
ex5_3_3	62	53	110
ex5_4_2	8	6	5
ex5_4_3	16	13	22
ex5_4_4	27	19	36
ex6_1_1	8	6	28
ex6_1_2	4	3	12
ex6_1_3	12	9	54
ex6_1_4	6	4	24
ex6_2_10	6	3	176
ex6_2_11	3	1	88
ex6_2_12	4	2	76
ex6_2_13	6	3	118
ex6_2_14	4	2	72
ex6_2_5	9	3	221
ex6_2_6	3	1	106
ex6_2_7	9	3	318
ex6_2_8	3	1	106
ex6_2_9	4	2	120
ex7_2_1	7	14	59
ex7_2_2	6	5	10
ex7_2_3	8	6	19
ex7_2_4	8	4	28
ex7_3_1	4	7	44
ex7_3_2	4	7	6
ex7_3_3	5	8	8
ex7_3_4	12	17	50
ex7_3_5	13	15	59
ex7_3_6	17	17	466
ex8_1_1	2	0	5
ex8_1_2	1	0	24
ex8_1_3	2	0	23
ex8_1_4	2	0	9
ex8_1_5	2	0	11
ex8_1_6	2	0	9
ex8_1_7	5	5	15

GlobalLib 2/3

Instance	*n*	*m*	*NLT*
ex8_1_8	6	5	10
ex8_2_1a	55	31	109
ex8_2_1b	57	33	61
ex8_2_1	55	31	109
ex8_2_2a	7510	1947	15018
ex8_2_2b	7522	1959	7530
ex8_2_2	7510	1947	15018
ex8_2_3a	15636	3155	31268
ex8_2_3b	15641	3160	15648
ex8_2_4a	55	81	309
ex8_2_4b	61	87	165
ex8_2_4	55	81	309
ex8_2_5a	2510	3774	30018
ex8_2_5b	2534	3798	15042
ex8_2_5	2510	3774	30018
ex8_3_10	141	108	291
ex8_3_11	115	76	274
ex8_3_12	120	81	284
ex8_3_13	115	72	269
ex8_3_14	110	71	294
ex8_3_1	115	76	274
ex8_3_2	110	76	219
ex8_3_3	110	76	214
ex8_3_4	110	76	219
ex8_3_5	110	76	219
ex8_3_6	110	76	215
ex8_3_7	126	92	276
ex8_3_8	126	93	270
ex8_3_9	78	45	107
ex8_4_1	22	10	30
ex8_4_2	24	10	70
ex8_4_3	52	25	75
ex8_4_4	17	12	36
ex8_4_5	15	11	53
ex8_4_6	14	8	88
ex8_4_7	62	40	160
ex8_4_8_bnd	42	30	170
ex8_4_8	42	30	170
ex8_5_1	6	4	33
ex8_5_2	6	4	33
ex8_5_3	5	4	22
ex8_5_4	5	4	22
ex8_5_5	5	4	30
ex8_5_6	6	4	42
ex8_6_1	75	45	315
ex8_6_2	30	0	315
ex9_1_10	14	12	5
ex9_1_1	13	12	5
ex9_1_2	10	9	4
ex9_1_4	10	9	4
ex9_1_5	13	12	5
ex9_1_8	14	12	5
ex9_2_1	10	9	8
ex9_2_2	10	11	6
ex9_2_3	16	15	6
ex9_2_4	8	7	6
ex9_2_5	8	7	5
ex9_2_6	16	12	10
ex9_2_7	10	9	8
ex9_2_8	6	5	4
filter	2	1	11
flowchan100	2400	2398	1600
flowchan200	4800	4798	3200
flowchan400	9600	9598	6400
flowchan50	1200	1198	800
ganges	356	273	1035
gangesx	356	273	1035
gasoil100	2603	2598	2202
gasoil200	5203	5198	4202
gasoil400	10403	10398	8202
gasoil50	1303	1298	1202
glider100	1315	1209	3527
glider200	2615	2409	7027
glider400	5215	4809	14027
glider50	665	609	1777
gsg_0001	78	112	108
gtm	63	24	42
harker	20	7	40
haverly	12	9	3
hhfair	29	25	28
himmel11	9	4	22
himmel16	18	21	54
house	8	8	4
hs62	3	1	14
hydro	31	24	24
immun	21	7	6
jbearing25	1404	0	21216
jbearing50	2704	0	41616
jbearing75	4004	0	62016
korcge	95	77	193
launch	38	28	128
least	3	0	22
like	9	3	625
linear	24	20	20
lnts100	506	400	1600
lnts200	1006	800	3200
lnts400	2006	1600	6400
lnts50	256	200	800
mathopt1	2	2	5
mathopt2	2	4	7
mathopt3	6	7	24
mathopt4	2	2	11
maxmin	27	78	234
meanvar	8	2	145
methanol100	3005	2997	5295
methanol200	6005	5997	10395
methanol400	12005	11997	20595
methanol50	1505	1497	2745
mhw4d	5	3	9
minlphi	64	79	72
minsurf25	1404	0	21216
minsurf50	2704	0	41616
minsurf75	4004	0	62016
nemhaus	5	5	18
otpop	103	76	83
pindyck	116	96	96
pinene100	5005	4995	2560
pinene200	10005	9995	4960
pinene25	2505	2495	1360
pinene50	2505	2495	1360
pollut	42	8	80
polygon25	50	324	1896
popdynm100	5615	5592	3352
popdynm200	11215	11192	6352
popdynm25	1415	1392	1342
popdynm50	2815	2792	2012
prob05	2	2	6
prob06	2	2	8
prob07	15	36	76
prob09	3	1	4

GlobalLib 3/3

Instance	*n*	*m*	*NLT*
process	10	7	13
prolog	20	22	8
qp1	50	2	5000
qp2	50	2	2550
qp3	100	52	50
qp4	79	31	58
qp5	108	31	0
ramsey	33	22	44
rbrock	2	0	4
robot100	1112	802	2307
robot200	2212	1602	4607
robot400	4412	3202	9207
robot50	562	402	1157
rocket100	607	502	1607
rocket200	1207	1002	3207
rocket400	2407	2002	6407
rocket50	307	252	807
sambal	17	10	26
sample	4	2	8
ship	10	16	40
srcpm	39	27	10
st_bpaf1a	10	10	5
st_bpaf1b	10	10	5
st_bpk1	4	6	4
st_bpk2	4	6	4
st_bpv1	4	4	2
st_bpv2	4	5	2
st_bsj2	3	5	3
st_bsj3	6	1	9
st_bsj4	6	4	9
st_cqpf	4	6	8
st_cqpjk1	4	2	8
st_cqpjk2	3	1	6
st_e01	2	1	1
st_e02	3	3	6
st_e03	10	7	14
st_e04	4	2	7
st_e05	5	3	4
st_e06	3	3	4
st_e07	10	7	3
st_e08	2	2	6
st_e09	2	1	4
st_e10	2	1	3
st_e11	3	2	5
st_e12	4	3	2
st_e16	12	9	23
st_e17	2	1	3
st_e18	2	4	4
st_e19	2	2	5
st_e20	6	5	10
st_e21	6	6	3
st_e22	2	5	3
st_e23	2	2	1
st_e24	2	4	1
st_e25	4	8	9
st_e26	2	4	4
st_e28	9	4	22
st_e30	14	15	8
st_e33	9	6	3
st_e34	6	4	20
st_e37	4	1	69
st_e39	2	0	9
st_e41	4	2	24
st_e42	7	2	6
st_fp1	5	1	10
st_fp2	6	2	10
st_fp3	13	10	8
st_fp4	6	5	2
st_fp5	10	11	14
st_fp6	10	5	20
st_fp7a	20	10	40
st_fp7b	20	10	40
st_fp7c	20	10	40
st_fp7d	20	10	40
st_fp7e	20	10	39
st_fp8	24	20	48
st_glmp_fp1	4	8	1
st_glmp_fp2	4	9	1
st_glmp_fp3	4	8	1
st_glmp_kk90	5	7	1
st_glmp_kk92	4	8	1
st_glmp_kky	7	13	2
st_glmp_ss1	5	11	1
st_glmp_ss2	5	8	1
st_ht	2	3	2
st_iqpbk1	8	7	116
st_iqpbk2	8	7	114
st_jcbpaf2	10	13	5
st_jcbpafex	2	2	1
st_kr	2	5	3
st_m1	20	11	36
st_m2	30	21	55
st_pan1	3	4	6
st_pan2	5	1	10
st_ph10	2	4	4
st_ph11	3	4	6
st_ph12	3	4	5
st_ph13	3	10	6
st_ph14	3	10	6
st_ph15	4	4	7
st_ph1	6	5	12
st_ph20	3	9	2
st_ph2	6	5	11
st_ph3	6	5	7
st_phex	2	5	3
st_qpc-m0	2	2	2
st_qpc-m1	5	5	50
st_qpc-m3a	10	10	200
st_qpc-m3b	10	10	200
st_qpc-m3c	10	10	200
st_qpc-m4	10	10	196
st_qpk1	2	4	6
st_qpk2	6	12	26
st_qpk3	11	22	51
st_robot	8	8	17
st_rv1	10	5	20
st_rv2	20	10	40
st_rv3	20	20	40
st_rv7	30	20	60
st_rv8	40	20	80
st_rv9	50	20	100
st_z	3	5	3
torsion25	1408	4	15912
torsion50	2708	4	31212
torsion75	4008	4	46512
trig	1	1	10
turkey	518	287	110
wall	6	6	6
water	41	25	102
weapons	65	12	130

Table 13 NLP instance statistics. *NLT* is the number of nonlinear terms in the problem; AMPL errors on `fct, worst`.

MINLPLib 1/2

Instance	*n*	*Bin.*	*Int.*	*m*	*NLT*
4stufen	149	48	0	98	111
alan	8	4	0	7	8
batchdes	20	9	0	20	13
batch	46	24	0	73	28
beuster	157	52	0	114	155
cecil_13	840	162	18	898	540
contvar	296	87	1	284	799
csched1	77	63	0	23	16
csched2	401	308	0	138	141
deb10	182	11	11	129	940
deb6	475	20	0	507	3108
deb7	813	10	10	897	6144
deb8	823	10	10	897	6144
deb9	813	10	10	917	6144
du-opt5	20	0	13	9	2336
du-opt	20	0	13	9	2336
elf	54	24	0	38	27
eniplac	141	24	0	189	90
enpro48	154	92	0	215	35
enpro56	128	73	0	192	31
ex1221	5	3	0	5	2
ex1222	3	1	0	3	3
ex1223a	7	1	0	9	9
ex1223b	7	2	0	9	17
ex1223	11	1	0	13	17
ex1224	11	5	0	7	5
ex1225	8	6	0	10	3
ex1226	5	3	0	5	8
ex1233	52	12	0	64	68
ex1243	68	16	0	96	32
ex1244	95	23	0	129	46
ex1252a	24	0	6	34	75
ex1252	39	15	0	43	75
ex1263a	24	4	20	35	16
ex1263	92	72	0	55	16
ex1264a	24	4	20	35	16
ex1264	88	68	0	55	16
ex1265a	35	5	30	44	25
ex1265	130	100	0	74	25
ex1266a	48	6	42	53	36
ex1266	180	138	0	95	36
ex3	32	8	0	31	7
ex4	37	25	0	31	254
fac1	22	6	0	18	4
fac2	66	12	0	33	6
fac3	66	12	0	33	6
feedtray2	88	36	0	284	885
feedtray	98	7	0	92	977
fo7_2	114	42	0	211	14
fo7	114	42	0	211	14
fo8	146	56	0	273	16
fo9	182	72	0	343	18
fuel	15	3	0	15	12
fuzzy	896	120	0	1056	72
gasnet	90	10	0	69	163
gastrans	106	21	0	149	45
gbd	4	3	0	4	2
gear2	28	24	0	4	4
gear3	8	0	4	4	4
gear4	6	0	4	1	4
gear	4	0	4	0	4
gkocis	11	3	0	8	3
hda	723	13	0	719	480
hmittelman	16	15	0	7	238
johnall	194	188	0	192	17860
lop97icx	986	67	785	87	556
m3	26	6	0	43	6
m6	86	30	0	157	12
m7	114	42	0	211	14
meanvarx	35	14	0	44	56
nous1	50	0	0	43	66
nous2	50	0	0	43	66
nuclear10a	13010	10920	0	3339	31988
nuclear14a	992	600	0	633	3984
nuclear14b	1568	600	0	1785	3408
nuclear14	1562	576	0	1226	4272
nuclear24a	992	600	0	633	3984
nuclear24b	1568	600	0	1785	3408
nuclear24	1562	576	0	1226	4272
nuclear25a	1058	650	0	659	4215
nuclear25b	1683	650	0	1909	3590
nuclear25	1678	625	0	1303	4590
nuclear49a	3341	2450	0	1431	10826
nuclear49b	5742	2450	0	6233	8425
nuclear49	5735	2401	0	3873	12541
nuclearva	351	168	0	317	1664
nuclearvb	351	168	0	317	1628
nuclearvc	351	168	0	317	1628
nuclearvd	351	168	0	317	2504
nuclearve	351	168	0	317	2504
nuclearvf	351	168	0	317	2504
nvs01	3	0	2	3	13
nvs02	8	0	5	3	23
nvs03	2	0	1	2	4
nvs04	2	0	2	0	4
nvs05	8	0	2	9	48
nvs06	2	0	2	0	13
nvs07	3	0	2	2	4
nvs08	3	0	2	3	11

MINLPLib 2/2

Instance	*n*	*Bin.*	*Int.*	*m*	*NLT*
nvs09	10	0	10	0	50
nvs10	2	0	2	2	16
nvs11	3	0	3	3	44
nvs12	4	0	4	4	92
nvs13	5	0	5	5	158
nvs14	8	0	5	3	22
nvs15	3	0	3	1	9
nvs16	2	0	2	0	8
nvs17	7	0	7	7	392
nvs18	6	0	6	6	258
nvs19	8	0	8	8	556
nvs20	16	0	5	8	122
nvs21	3	0	2	2	11
nvs22	8	0	4	9	48
nvs23	9	0	9	9	762
nvs24	10	0	10	10	1020
o7_2	114	42	0	211	14
o7	114	42	0	211	14
oaer	9	3	0	7	3
oil2	937	2	0	927	1981
oil	1535	19	0	1546	2871
ortez	87	18	0	74	57
parallel	205	25	0	115	280
prob02	6	0	6	8	5
prob03	2	0	2	1	1
prob10	3	0	1	3	9
procsel	10	3	0	7	3
product2	2842	128	0	3125	2112
product	1553	107	0	1925	528
pump	24	0	6	34	75
qapw	450	225	0	255	225
ravem	112	53	1	186	43
risk2b	463	12	2	580	6
saa_2	4407	400	0	6205	143699
sep1	29	2	0	31	6
space25a	383	240	0	201	101
space25	893	750	0	235	101
space960	5537	0	960	6497	3748
spectra2	70	30	0	73	504
spring	17	9	1	8	19
st_e13	2	1	0	2	1
st_e14	11	1	0	13	17
st_e15	5	3	0	5	2
st_e27	4	2	0	6	2
st_e29	11	5	0	7	5
st_e31	112	24	0	135	8
st_e32	35	0	19	18	92
st_e35	32	7	0	39	59
st_e36	2	0	1	2	36
st_e38	4	0	0	3	17
st_e40	4	0	3	8	24
st_miqp1	5	5	0	1	10
st_miqp2	4	2	2	3	4
st_miqp3	2	0	2	1	2
st_miqp4	6	3	0	4	6
st_miqp5	7	2	0	13	4
stockcycle	480	432	0	97	48
st_test1	5	5	0	1	8
st_test2	6	5	1	2	10
st_test3	13	10	3	10	10
st_test4	6	2	4	5	4
st_test5	10	10	0	11	14
st_test6	10	10	0	5	20
st_test8	24	0	24	20	48
st_testgr1	10	0	10	5	20
st_testgr3	20	0	20	20	40
st_testph4	3	0	3	10	6
super1	1308	44	0	1659	2545
super2	1308	44	0	1659	2545
super3	1308	44	0	1659	2545
super3t	1056	44	0	1343	2169
synheat	56	12	0	64	82
synthes1	6	2	0	6	11
synthes2	11	3	0	14	9
synthes3	17	5	0	23	16
tln12	168	12	156	72	144
tln2	8	2	6	12	4
tln4	24	4	20	24	16
tln5	35	5	30	30	25
tln6	48	6	42	36	36
tln7	63	7	56	42	49
tloss	48	6	42	53	36
tls12	812	656	12	384	288
tls2	37	31	2	24	8
tls4	105	85	4	64	32
tls5	161	131	5	90	50
tls6	215	173	6	120	72
tls7	345	289	7	154	98
tltr	48	12	36	54	27
uselinear	6792	58	0	7030	9549
util	146	28	0	168	5
var_con10	573	10	2	464	4368
var_con5	573	10	2	464	4368
waste	2484	400	0	1991	1368
water4	195	126	0	137	32
waterx	70	14	0	54	102
waterz	195	126	0	137	32
windfac	14	0	3	13	36

Table 14 MINLP instance statistics. *NLT* is the number of nonlinear terms in the problem. AMPL errors on `blendgap, meanvarxsc, water3, waterful2, watersbp, waters, watersym1, watersym2` (MINLPLib).