SYMMETRY IN MATHEMATICAL PROGRAMMING

LEO LIBERTI[*]

Abstract. Symmetry is mainly exploited in mathematical programming in order to reduce the computation times of enumerative algorithms. The most widespread approach rests on: (a) finding symmetries in the problem instance; (b) reformulating the problem so that it does not allow some of the symmetric optima; (c) solving the modified problem. Sometimes (b) and (c) are performed concurrently: the solution algorithm generates a sequence of subproblems, some of which are recognized to be symmetrically equivalent and either discarded or treated differently. We review symmetry-based analyses and methods for Linear Programming, Integer Linear Programming, Mixed-Integer Linear Programming and Semidefinite Programming. We then discuss a method (introduced in [35]) for automatically detecting symmetries of general (nonconvex) Nonlinear and Mixed-Integer Nonlinear Programming problems and a reformulation based on adjoining symmetry breaking constraints to the original formulation. We finally present a new theoretical and computational study of the formulation symmetries of the Kissing Number Problem.

Key words. MINLP, NLP, reformulation, group, graph isomorphism, permutation, expression tree.

AMS(MOS) subject classifications. 90C11, 90C26, 90C30, 05C25, 20B25.

1. Introduction.

Mathematical Programming (MP) is a language for formally describing classes of optimization problems. A MP consists of: parameters, encoding the problem input; decision variables, encoding the problem output; one objective function to be minimized; and a set of constraints describing the feasible set (some of these constraints may be bounds or integrality requirements on the decision variables). The objective function and constraints are described by mathematical expressions whose arguments are the parameters and the decision variables. Let $N = \{1, \ldots, n\}$ and $M = \{1, \ldots, m\}$ for some nonnegative integers m, n, and $Z \subseteq N$. In general, a MP formulation is as follows:

$$\left.\begin{array}{rrcl} \min & f(x) & & \\ \text{s.t.} & g(x) & \leq & 0 \\ \forall i \in Z & x_i & \in & \mathbb{Z}, \end{array}\right\} \qquad (1.1)$$

where $x \in \mathbb{R}^n$ is a vector of decision variables, and $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}^m$ are functions that can be written as mathematical expressions involving of a finite number of operators (e.g. $\{+, -, \times, \div, \uparrow, \log, \exp, \sin, \cos, \tan\}$) and x as arguments. If f, g are affine forms and $Z = \emptyset$, (1.1) is a Linear Program (LP). If f, g contain some nonlinear term and $Z = \emptyset$, (1.1) is a Nonlinear Program (NLP), and if f is a convex function and the feasible set $\{x \mid g(x) \leq 0\}$ is convex, (1.1) is a convex NLP (cNLP). If f, g are affine and

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

$Z \neq \emptyset$, (1.1) is a Mixed-Integer Linear Program (MILP) , and if $Z = N$ it is an Integer Linear Program (ILP). If f, g contain some nonlinear term and $Z \neq \emptyset$, (1.1) is a Mixed-Integer Nonlinear Program (MINLP), and if f and the feasible set are convex, it is a convex MINLP (cMINLP). A somewhat special case of MP, called Semidefinite Programming (SDP) is when f, g are affine forms, $Z = \emptyset$, x is a square matrix, and there is an additional constraint stating that x must be symmetric positive semidefinite ($x \succeq 0$). Although the semidefinite constraint cannot be written as a mathematical expression of the operators listed above, SDP is important because it can be solved in polynomial time by a special-purpose interior point method [1], and because many tight relaxations of polynomial programming problems can be cast as SDPs.

Symmetries have been used in MP for analysis purposes or in order to speed up solution methods. The general approach is as follows. First, symmetries are detected, either algorithmically or because of some known mathematical property of the given optimization problem. Once (some) problem symmetries are known, either the MP is reformulated so that some symmetric optima become infeasible and then solved via standard solution methods (static symmetry breaking [42]), or a known solution method is modified so that it recognizes and exploits symmetry dynamically as it goes along. Symmetries in MP can be broadly classified in two types: *solution* symmetries, i.e. those variable symmetries that fix the set of solutions setwise; and *formulation* symmetries, i.e. those variable symmetries that fix the formulation (encoded in some data structure). If the formulation group structure for a given MP varies considerably from instance to instance, then automatic symmetry detection methods may be required.

In general, when one reads the existing literature on the subject, one realizes that the main effort is that of *removing* symmetries from a problem in order to find a global optimum more quickly. After extensive computational experimentations with all the symmetric instances of most public instance libraries (MIPLib, GlobalLib and MINLPLib) solved by means of Couenne [6] and BARON [56] (both implementing a spatial Branch-and-Bound (sBB) algorithm) and also RECIPE [37] (based on the Variable Neighbourhood Search (VNS) metaheuristic [22]), my own very personal opinion is that removing symmetries is *good* when solving with sBB and *bad* when solving with VNS. The sBB algorithm is a tree search based on bisecting the variable bounds at each tree node along the spatial direction generating the largest error between the solutions of the upper bounding and lower bounding subproblems. The leaf nodes of the search tree contain globally optimal solutions or infeasible portions of space. If the problem has many symmetric optima, a correspondingly large number of leaves contain these optima. When some of the symmetric optima are eliminated from the feasible region, the bound relative to parent nodes is likely to increase, which accelerates sBB convergence. VNS, on the other hand, works by jumping from optimum to optimum via paths defined by local searches started from random points in increasingly large neighbourhoods of the incumbent, attempting to improve the objective function value. Since in general it is important to explore as much as possible of the feasible region, one usually tends to move to a new optimum even though the objective function value stays the same. This provides an intuitive explanation as regards the generally poor performance of the RECIPE algorithm on reformulated problems with fewer symmetric optima compared to the original symmetric instances.

In this paper, we provide a study of symmetry for NLPs and MINLPs in general form (no convexity assumption is made on f, g). Our literature review (Sect. 2) first presents a general overview of mathematical programming techniques drawing from group theory, particularly in regard to LP, ILP, MILP, SDP, and then focuses on those items that are relevant to automatic symmetry detection (one of the main topics discussed in this paper). An automatic symmetry detection method (originally introduced in [35]) is recounted in Sections 3-4: we construct a digraph that encodes the structure of the mathematical expression representation of f, g, and then apply known graph-based symmetry detection algorithms to derive the group of permutations of the decision variable indices that fix the symbolic structure of f, g. In Sect. 5 we introduce some linear inequalities that are valid for at least one optimum of (1.1) but which are likely to make at least some symmetric optima infeasible. We then present an original application of the proposed techniques to the Kissing Number Problem [29] in Sect. 6: we use our automatic symmetry detection method to formulate a conjecture on the KNP group structure, which we then prove to be true; we derive some symmetry breaking inequalities, and discuss computational results which show the positive impact of the proposed approach.

1.1. Notation. We follow the notation style common in classical algebra, see e.g. [10, 2], with some modifications drawn from computational group theory [59]. 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. For any function $f : S \to T$ (where S, T are sets) we denote S (the domain of f) by $\mathrm{dom}(f)$.

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 $Y = \{y\}$); xG is also known as the *orbit* of x in G (and 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 such that $XH = X$. For any permutation $\pi \in S_n$, let $\Gamma(\pi)$ be the set of its disjoint cycles, so that $\pi = \prod\limits_{\tau \in \Gamma(\pi)} \tau$. For a group G and $\pi \in G$ let $\langle \pi \rangle$ be the subgroup of G generated by π, and for a subset $S \subseteq G$ let $\langle S \rangle$ be the subgroup of G generated by all elements of S. 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 π fixes B setwise and $\rho = \pi[B]$ is the permutation of B induced by π. Because disjoint cycles commute, it follows from the definition that for all $k \in \mathbb{N}$, $\pi^k[B] = (\pi[B])^k$. A group $G \leq S_n$ with generators $\{g_1, \ldots, g_s\}$ acts on $B \subseteq \{1, \ldots, n\}$ as H if $\langle g_i[B] \mid i \leq s \rangle = H$; in this case we denote H by $G[B]$. If B is an orbit of the natural action of G on the integers (i.e. the natural action of G on $\bigcup\limits_{\pi \in G} \mathrm{dom}(\pi)$, which fixes every other integer), then it is easy to show that $G[B]$ is a transitive constituent of G [21]. In general, $G[B]$ may not be a subgroup of G: take $G = \langle (1,2)(3,4), (1,3), (4,2) \rangle$ and $B = \{1,2\}$, then $G[B] = \langle (1,2) \rangle \not\leq G$. 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 $\pi[B \cup D] = \pi[B]\pi[D]$.

2. Literature review.

This literature review does not only cover the material strictly inherent to later sections, but attempts to be as informative as possible as concerns the use of symmetry in MP, in such a way as to provide an overview which is complementary to [42]. More specifically, the first part (Sect. 2.1-2.3) of this review will cover a representative subset of the most important works about symmetry in optimization, notably in LP, MILP and (briefly) SDP. We survey those topics which are most relevant to later sections (i.e. symmetry detection methods) in Sect. 2.4.

2.1. Symmetry in Linear Programming.

The geometrical objects of LP are polyhedra, and there is a very rich literature on symmetric polyhedra [53]. Such results, however, are mostly about the classification of symmetric polyhedra and are rarely used in algorithmics.

The inherent symmetry of the simplex algorithm is studied in [63, 64, 65]. Given two $m \times n$ matrices A, B, let $\mathscr{S}_A = \{x \in \mathbb{R}^{m+n} \mid (I|A)x = 0\}$ (where $(I|A)$ is the $m \times (m+n)$ matrix formed by the $m \times m$ identity followed by the columns of A) and $\mathscr{S}_B = \{x \in \mathbb{R}^{m+n} \mid (I|B)x = 0\}$; A, B are combinatorially equivalent (written $A :: B$) if there exists π in the symmetric group S_{m+n} such that $\pi \mathscr{S}_A = \mathscr{S}_B$. The paper [63] gives different necessary and sufficient conditions for $A :: B$ (among which a formula for constructing all combinatorially equivalent matrices from submatrices of A). In [64] an application to solving matrix games via the simplex method is presented. In [54], Tucker's combinatorial equivalence is used to devise a simplex algorithm variant capable of solving a pair of primal/dual LPs directly without many of the necessary pre-processing steps.

2.2. Symmetry in Mixed-Integer Linear Programming.

The existing work on symmetry in MILP 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); (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. We consider MILPs of the form $\min\{cx \mid Ax \le b \wedge \forall i \in Z\ x_i \in \mathbb{Z}\}$.

Category (a) was established by R. Gomory [20]: given a basis B of the constraint matrix A, it exploits the (abelian) group $\mathscr{G} = \mathbb{Z}^n/\langle \mathrm{col}(B)\rangle$, where $\mathbb{Z}^n$ is the additive group of integer n-sequences and $\langle \mathrm{col}(B)\rangle$ is the additive group generated by the columns of the (nonsingular) matrix B. Consider the natural group homomorphism $\varphi : \mathbb{Z}^n \to \mathscr{G}$ with $\ker\varphi = \langle \mathrm{col}(B)\rangle$: letting (x_B, x_N) be a basic/nonbasic partition of the decision variables, apply φ to the standard form constraints $Bx_B + Nx_N = b$ to obtain $\varphi(Bx_B)+\varphi(Nx_N) = \varphi(b)$. Since $\varphi(Bx_B) = 0$ if and only if $x_B \in \mathbb{Z}^n$, setting $\varphi(Nx_N) = \varphi(b)$ is a necessary and sufficient condition for x_B to be integer feasible. Gomory's seminal paper gave rise to further research, among which [68, 5]. The book [24] is a good starting point.

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. [40]), appropriate global cuts (e.g. [60]) or special formulations [30, 9] 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 two main research streams. The first was established by Margot in the early 2000s [38, 39], and is applicable to Binary Linear Programs (BLPs) in the form:

$$\left.\begin{array}{rrcl} \min & cx & & \\ & Ax & \le & b \\ & x & \in & \{0,1\}^n. \end{array}\right\}$$

Margot [38, 42] defines the *relaxation group* $G^{\mathrm{LP}}(P)$ of a BLP P as:

$$G^{\mathrm{LP}}(P) = \{\pi \in S_n \mid c\pi = c \wedge \exists \sigma \in S_n\ (\sigma b = b \wedge \sigma A \pi = A)\}, \qquad (2.1)$$

or, in other words, all relabellings of problem variables for which the objective function and constraints are the same. The relaxation group (2.1) 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 [41]). Further results along the same lines (named *orbital branching*) are obtained for covering and packing problems in [49, 50]: if O is an orbit of some subgroup of the relaxation group, at each BB node the disjunction $\left(\bigvee_{i\in O} x_i = 1\right) \vee \sum_{i\in O} x_i = 0$ 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.

The second was established by Kaibel et al. in 2007 [25, 15], with the introduction of the packing and partitioning orbitopes, i.e. convex hulls of certain 0-1 matrices that represent possible solutions to sets of packing and partitioning constraints. These are used in problems defined in terms of matrices of binary decision variables x_{ij} (for $i \leq m, j \leq n$). Since a typical packing constraint is $\sum_{j\in J_i} x_{ij} \leq 1$ for some $i \leq m, J_i \subseteq \{1,\dots,n\}$ (partitioning constraints simply replace inequalities with equations), sets of such constraint may exhibit column symmetries in $\prod_{i\leq m} \mathrm{Sym}(J_i)$, and row symmetries in S_m. Orbitopes are convex hulls of binary $m \times n$ matrices that have lexicographically ordered columns: their vertices represent a subset of feasible solutions of the corresponding packing/partitioning problem from which several symmetries have been removed. Given a partition $C_1,\dots,C_q$ of the variable indices, a permutation $\pi \in G^{\mathrm{LP}}(P)$ is an *orbitopal symmetry* if there are $p, r \leq q$ such that π is a bijection $C_p \to C_r$ that keeps all other C_s elementwise fixed, for $s \notin \{p,r\}$ [7]. In [25], a complete description of packing/partitioning orbitopes in terms of linear inequalities is provided ([15] gives a much shorter, more enlightening and less technical presentation than that given in [25]). Inspired by the work on orbitopes, E. Friedman proposed a similar but more general approach leading to *fundamental domains* [17]: 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 $GF = X$.

2.3. Symmetry in Semidefinite Programming. There are several works describing the exploitation of symmetry in semidefinite programming (see e.g. [26, 19, 27]). Much of the material in this section is taken from the commendable tutorial [67]. Consider the following SDP:

$$\left.\begin{array}{rrcl} \min_X & C \bullet X & & \\ \forall k \leq m & A_k \bullet X & \leq & b_i \\ & X & \succeq & 0, \end{array}\right\} \qquad (2.2)$$

where X is an $n\times n$ symmetric matrix abd $M_1 \bullet M_1 = \mathrm{trace}({M_1}^\top M_2)$ is the trace product between matrices M_1, M_2. Let G^{SDP} be the largest subgroup of S_n such that if X^* is an optimum then πX^* is also an optimum for all $\pi \in G^{\mathrm{SDP}}$, where the action of π on an $n \times n$ matrix M is to permute the columns *and* the rows of M according to π. If X^* is an optimum, taking $\frac{1}{|G^{\mathrm{SDP}}|} \sum\limits_{\pi\in G} \pi X^*$ shows that there is always an optimal solution of (2.2) in $\mathcal{B}$, the space of G^{SDP}-invariant matrices. Let $R_1,\dots,R_k$ be the orbits of $\{(i,j) \mid i,j \leq n\}$ under G^{SDP}, and for all $r \leq k$ let $B^r = (b^r_{ij})$ the 0-1 incidence matrix of $(i,j) \in R_r$ (i.e. $b^r_{ij} = 1$ of $(i,j) \in R_r$ and 0 otherwise). Then $B^1,\dots,B^k$ is a basis of $\mathcal{B}$ and (2.2) can be re-cast as a search over the coefficients of a linear form in $B^1, \ldots, B^k$:

$$\left.\begin{array}{rrcl} \min_y & \sum\limits_{j\le k}(C\bullet B^j)y_j & & \\ \forall i\le m & \sum\limits_{j\le k}(A_i\bullet B^j)y_j & = & b_i \\ & \sum\limits_{j\le k} y_jB^j & \succeq & 0. \end{array}\right\} \quad (2.3)$$

By rewriting (2.2) and (2.3) over $\mathbb{C}$, $\mathcal{B}$ becomes a semisimple algebra over $\mathbb{C}$. This implies that it is possible to find an algebra isomorphism

$$\phi : \mathcal{B} \to \bigoplus_{t\le d} \mathbb{C}^{m_t\times m_t}$$

for some integers d and m_t $(t \le d)$. This allows a size reduction of the SDP being solved, as the search only needs to be conducted on the smaller-dimensional space spanned by $\phi(\mathcal{B})$.

A different line of research is pursued in [26]: motivated by an application (truss optimization), it is shown that the barrier subproblem of a typical interior point method for SDP "inherits" the same symmetries as the original SDP.

2.4. Automatic symmetry detection. Automatic symmetry detection does not appear prominently in the mathematical programming literature. A method for finding the MILP relaxation group (2.1), based on solving an auxiliary MILP encoding the condition $\sigma A\pi = A$, was proposed and tested in [32] (to the best of our knowledge, the only approach for symmetry detection that does not reduce the problem to a graph). A more practically efficient method consists in finding the automorphism group of vertex-colored bipartite graph encoding the incidence of variables in constraints. If the symmetry π is orbitopal and the system $Ax \le b$ contains at least a *leading constraint*, i.e. a π-invariant constraint that has exactly one nonzero column in each C_p (for $p \le q$) then a set of generators for $G^{\mathrm{LP}}(P)$ can be found in linear time in the number of nonzeroes of A [7].

The Constraint Programming (CP) literature contains many papers on symmetries. Whereas most of them discuss symmetry breaking techniques, a few of them deal with automatic symmetry detection and are relevant to the material presented in the rest of the paper; all of them rely on reducing the problem to a graph and solving the associated GRAPH ISOMORPHISM (GI) problem. In CP, symmetries are called *local* if they hold at a specific search tree node, and *global* otherwise. Solution symmetries are also called *semantic* symmetries, and formulation symmetries are also called *syntactic* or *constraint* symmetries. A Constraint Satisfaction Problem (CSP) can be represented by its *microstructure complement*, i.e. a graph whose vertices are assignments $x = a$ (where x ranges over all CSP variables and a over all values in the domain of x), and whose edges $(x_i = a, x_j = b)$ indicate that the two assignments $x_i = a$ and $x_j = b$ are incompatible either because of a constraint in the CSP or because $i = j$ and $a \neq b$. Constraint symmetries are defined in [11] as the automorphisms of the microstructure complement. A *k-ary nogood* is a k-partial solution (i.e. an assignment of values to k variables) which cannot be extended to a full solution of the given CSP instance. The *k-nogood hypergraph* of the CSP has assignments $x = a$ as vertices and all m-ary nogoods as hyperedges, for $m \leq k$. For a k-ary CSP (one whose constraints have maximum arity k), the group of solution symmetries is equal to the automorphism group of its k-nogood hypergraph [11]. In [13] (possibly the first work in which a reduction from formulation-type symmetries to GI was proposed), SAT symmetries are automatically detected by reducing the problem to a bipartite graph, and identified by solving the corresponding GI instance, similarly to the approach taken in [49]. In [52], constraints involving the arithmetic operations $+, -, \times$ are reduced to Directed Acyclic Graphs (DAG) whose leaf vertices represent variables and intermediate vertices represent operators; vertex colors identify same operator types and constraints having the same right hand sides. Thm. 3.1 in [52] shows that the automorphism group of this DAG is isomorphic to the constraint group of the corresponding CSP instance, and might be considered the CP equivalent of Thm. 4.1 and Thm. 4.2 appearing below (although the proof techniques are completely different). In [51], a systematic reduction of many types of constraints to an equivalent graph form is proposed; an improved representation and extensive computational results are given in [45]. The problem of determining the constraint group of a model (instead of an instance) is discussed in [46] — we pursue a similar line of reasoning when inferring the structure of the KNP group (independently of the instance) from a sequence of automatically computed KNP instance groups.

3. Groups of a mathematical program. Let P be a MP with formulation as in (1.1), and $\mathcal{F}(P)$ (resp. $\mathcal{G}(P)$) be the set of its feasible (resp. globally optimal) points. Two important groups are connected with P. The *solution group* is the group of all permutations of the variable indices which map $\mathcal{G}(P)$ into itself; it is defined formally as $G^*(P) = \mathrm{stab}(\mathcal{G}(P), S_n)$ and contains as a subgroup the "symmetry group" of P, defined limited to MILPs in [42] as the group of permutations mapping feasible solutions into feasible solutions having the same objective function value. Computing solution groups directly from their definition would imply knowing $\mathcal{G}(P)$ aprioristically, which is evidently irrealistic.

The other important group related to P (denoted by $\bar{G}_P$) fixes the formulation of P. For two functions $f_1, f_2 : \mathbb{R}^n \to \mathbb{R}$ we write $f_1 = f_2$ to mean $\mathrm{dom}(f_1) = \mathrm{dom}(f_2) \wedge \forall x \in \mathrm{dom}(f_1)\ (f_1(x) = f_2(x))$. Then

$$\bar{G}_P = \{\pi \in S_n \mid Z\pi = Z \wedge f(x\pi) = f(x) \wedge \exists\, \sigma \in S_m\ (\sigma g(x\pi) = g(x))\}.$$

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

The definition of $\bar{G}_P$ implies the existence of a method for testing whether $f(x\pi) = f(x)$ and whether there is a permutation $\sigma \in S_m$ such that $\sigma g(x\pi) = g(x)$. Since Nonlinear Equations (determining if a set of general nonlinear equations has a solution) is an undecidable problem in general [69], such tests are algorithmically intractable. Instead, we assume the existence of a YES/NO oracle $\equiv$ that answers YES if it can establish that $f_1 = f_2$ (i.e. f_1, f_2 have the same domain and are pointwise equal on their domain). Such an oracle defines an equivalence relation $\equiv$ on the set of all functions appearing in (1.1): if a pair of functions (f_1, f_2) belongs to the relation then the functions are equal, but not all pairs of equal functions might belong to $\equiv$ (i.e. $\equiv$ might answer NO even though $f_1 = f_2$). This weakening of the equality relationship will allow us to give an algorithmically feasible definition of the symmetry group of the formulation.

We define the $\equiv$ oracle by only considering functions that can be written syntactically using infix notation in terms of a finite set of operators (e.g. arithmetic, logarithm, exponential and so on), a finite set of constants in $\mathbb{Q}$ and the set of problem variables $x_1, \ldots, x_n$. Such functions can be naturally represented by means of expression trees (Fig. 1 left) which, by contracting leaf vertices with equal labels, can be transformed into DAGs as shown in Fig. 1 (right). The $\equiv$ oracle is then implemented as a recur-

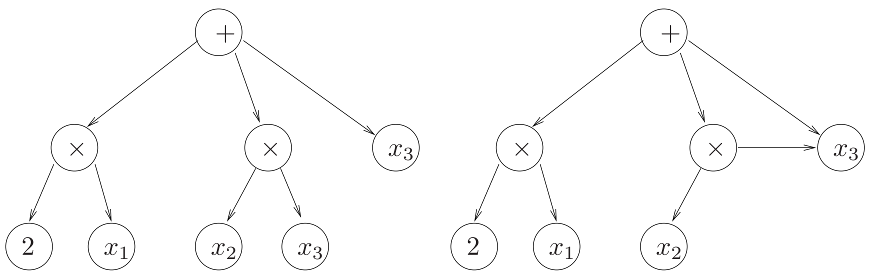

Fig. 1. *Expression tree for $2x_1 + x_2x_3 + x_3$ (left). Equal variable vertices can be contracted to obtain a DAG (right).*

sive graph exploration. The function DAG representation is well known and should perhaps be attributed to folklore (it is mentioned e.g. in [28], Sect. 2.3). DAG representable functions are routinely used in Global Optimization (GO) to automatically build relaxations of MINLPs [61, 31, 6] or tighten the variable bounds [57]. In the context of symmetry in MP, precise definitions for DAG representable functions and the $\equiv$ oracle implementation are given in [35, 12]. The *formulation group* of P can now be defined as:

$$G_P = \{\pi \in S_n | Z\pi = Z \wedge f(x\pi) \equiv f(x) \wedge \exists \sigma \in S_m (\sigma g(x\pi) \equiv g(x))\}. \quad (3.1)$$

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 (1.1), if P is a BLP, then $\bar{G}_P = G^{\mathrm{LP}}(P)$ [35]. We remark that if f_1, f_2 are linear forms, then $f_1 = f_2$ implies $f_1 \equiv f_2$. In other words, for linear forms, $\equiv$ and $=$ are the same relation [35]. As a corollary, if P is a BLP, then $G_P = G^{\mathrm{LP}}(P)$.

If a set of mathematical functions share the same arguments, as for the objective function f and constraints g of (1.1), the corresponding DAGs for $f, g_1, \ldots, g_m$ can share the same variable leaf vertices. This yields a DAG $D_P = (V_P, A_P)$ (formed by the union of all the DAGs of functions in P followed by the contraction of leaf vertices with same variable index label) which represents the mathematical structure P [48, 57].

4. Automatic computation of the formulation group. The method proposed in this section also appears (with more details) in [35]. As mentioned in the literature review, similar techniques are available in CP [52].

We first define an equivalence relation on V_P which determines the interchangeability of two vertices of D_P. Let $\mathcal{S}_F$ be the singleton set containing the root vertex of the objective function, $\mathcal{S}_C$ of all constraint root vertices, $\mathcal{S}_O$ of all vertices representing operators, $\mathcal{S}_K$ of all constant vertices and $\mathcal{S}_V$ of all variable vertices. For $v \in \mathcal{S}_F$, we denote optimization direction of the corresponding objective function by $d(v)$; for $v \in \mathcal{S}_C$, we denote the constraint sense by $s(v)$. For $v \in \mathcal{S}_O$, we let $\ell(v)$ be the level of v in D_P, i.e. the length of the path from the root to v (ℓ is well defined as the only vertices with more than one incoming arc are the leaf vertices), $\lambda(v)$ be its operator label and $o(v)$ be the order of v as an argument of its parent vertex if the latter represents a noncommutative operator, or 1 otherwise. For $v \in \mathcal{S}_K$, we let $\mu(v)$ be the value of v. For $v \in \mathcal{S}_V$ 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 now define the relation $\sim$ on V_P as follows.

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

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

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 $(u, v) \in A$ [55]. Let $G^{\mathrm{DAG}}(P)$ be the largest subgroup of $\mathrm{Aut}(D_P)$ fixing V_k setwise for all $k \leq K$. We assume without loss of generality that the vertices of D_P are uniquely numbered so that for all $j \leq n$, the j-th vertex corresponds to the leaf vertex for variable x_j (the rest of the numbering is not important), i.e. $\mathcal{S}_V = \{1, \ldots, n\}$.

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

Theorem 4.1 ([35], Thm. 4). *ψ is a group homomorphism if and only if G stabilizes ω setwise.*

Next, we note that $G^{\text{DAG}}(P)$ fixes $\mathcal{S}_V$ setwise [35]. As a corollary to Thm. 4.1, the map $\varphi : G^{\text{DAG}}(P) \to \text{Sym}(\mathcal{S}_V)$ given by $\varphi(\gamma) = \gamma[\mathcal{S}_V]$ is a group homomorphism.

Theorem 4.2 ([35], Thm. 7). $\text{Im}\varphi = G_P$ *groupwise.*

By Thm. 4.2, we can automatically generate G_P by looking for the largest subgroup of $\text{Aut}(D_P)$ 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 GI problem [3]. 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 [66]. Rooted DAGs are GI-complete [8] but there is an algorithm for solving the GI problem on trees which is linear in the number of vertices in the tree ([55], Ch. 8.5.2). This should give an insight as to the type of difficulty inherent to computing $\text{Aut}(D_P)$.

Corollary 4.1. *If C' is a set of group generators of $G^{DAG}(P)$, then $C = \{\pi[\mathcal{S}_V] \mid \pi \in C'\}$ is a set of generators for G_P.*

Cor. 4.1 allows the practical computation of a formulation group: one first forms the graph D_P, then computes generators C' of $G^{\text{DAG}}(P)$, and finally considers their action on $\mathcal{S}_V$ to explicitly construct C. Based on the results of this section, we implemented a software system (called `symmgroup`) that automatically detects the formulation group of a problem (1.1). Our system first calls AMPL [16] to parse the instance; the ROSE Reformulation/Optimization Software Engine [36] 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 [44, 43] `dreadnaut` shell to efficiently compute the generators of $\text{Aut}(D_P)$. A system of shell scripts and Unix tools parses the *nauty* output to form a valid GAP [18] input, used to print the actual group description via the command `StructureDescription`.

5. Symmetry Breaking Constraints.

Once the formulation group is detected, we can adjoin constraints to (1.1) in order to make some of the symmetric optima infeasible. According to the classification in [34], this is a reformulation of the narrowing type.

Definition 5.1. *Given a problem P, a narrowing Q of P is a formulation (1.1) such that (a) there is a function $\eta : \mathcal{F}(Q) \to \mathcal{F}(P)$ for which*

$\eta(\mathcal{G}(Q))$ (the image of $\mathcal{G}(Q)$ under η) is a subset of $\mathcal{G}(P)$, and (b) Q is infeasible only if P is.

Our narrowing rests on adjoining some static symmetry breaking inequalities (SSBIs) [42] 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 can then be solved by standard software packages such as CPLEX [23] (for MILPs) and Couenne [6] or BARON [56] for MINLPs.

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

Definition 5.2. *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, $\{g_1(x) \le 0, \dots, g_q(x) \le 0\}$) 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 is 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, $X = \mathcal{G}(P)$. 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 $B \subseteq \{1, \dots, n\}$. If $g(x) \equiv g(x[B])$ (i.e. 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 1. *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. $\{x_1 \le x_2, x_2 \le x_3\}$ are SBCs with respect to S_3 and X because $(-1, 1, 1) = y(1, 2, 3) \in XS_n$; however, they are not SBCs with respect 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 5.1 ([35], Thm. 11). *If $g(x) \le 0$ are SBCs for any subgroup G of G_P and $\mathcal{G}(P)$, then the problem Q obtained by adjoining $g(x) \le 0$ to the constraints of P is a narrowing of P.*

6. An application to the Kissing Number Problem. Given positive integers D, N, the decision version of the Kissing Number Problem (KNP) [29] asks whether N unit spheres can be positioned adjacent to a unit sphere centered in the origin in $\mathbb{R}^D$. The optimization version asks for the maximum possible N. The pedigree of this problem is illustrious, having originated in a discussion between I. Newton and D. Gregory. The name of the problem arose is linked to billiard game jargon: when two balls touch each other, they are said to "kiss". As both Newton and Gregory were of British stock, one may almost picture the two chaps going down the pub arm in arm for a game of pool and a pint of ale; and then, in the fumes of alcohol, getting into a brawl about whether twelve or thirteen spheres might kiss a central one if the billiard table was tridimensional. This theory disregards the alleged scholarly note (mentioned in [62]) about the problem arising from an astronomical question. When $D = 2$, the maximum feasible N is of course 6 (hexagonal lattice). When $D = 3$, the maximum feasible N was conjectured by Newton to be 12 and by Gregory to be 13 (Newton was proven right 180 years later [58]). The problem for $D = 4$ was settled recently with $N = 24$ [47]. The problem for $D = 5$ is still open: a lower bound taken from lattice theory is 40, and an upper bound derived with Bachoc and Vallentin's extension [4] of Delsarte's Linear Programming (LP) bound [14] is 45.

We formulate the decision version of the KNP as a nonconvex NLP:

$$\left.\begin{array}{rrcl} \max_{x,\alpha} & \alpha & & \\ \forall i \le N & \|x^i\|^2 & = & 4 \\ \forall i < j \le N & \|x^i - x^j\|^2 & \ge & 4\alpha \\ \forall i \le N & x^i & \in & [-2,2]^D \\ & \alpha & \in & [0,1]. \end{array}\right\} \tag{6.1}$$

For any given $N, D > 1$, if a global optimum (x^*, α^*) of (6.1) has $\alpha^* = 1$ then a kissing configuration of N balls in $\mathbb{R}^D$ exists; otherwise, it does not. In practice, (6.1) is usually solved by heuristics such as Variable Neighbourhood Search (VNS) [29], because solving it by sBB takes too long even on very small instances. One of the reasons for the slow convergence of sBB is that (6.1) has many symmetries. In fact, $\mathrm{Aut}(\mathcal{G}(\mathrm{KNP}))$ has infinite (uncountable) cardinality: each optimum x^* can be rotated by any angle in $\mathbb{R}^D$, and hence for all orthogonal transformations $\mu \in SO(D, \mathbb{R})$ (the special orthogonal group of $\mathbb{R}^D$), $\mu(x^*) \in \mathcal{G}(\mathrm{KNP})$. Such symmetries can be easily disposed of by deciding the placement of D spheres so that they are mutually adjacent as well as adjacent to the central sphere in $\mathbb{R}^D$, but computational experience suggests that this does little, by itself, to decrease the size of the sBB tree.

We used the `symmgroup` system in order to detect the structure of $G_{(6.1)}$ automatically for a few KNP instances, obtaining an indication that $G_{(6.1)} \cong S_D$. However, since D is small with respect to N, this is not likely to help the solution process significantly. Let $x^i = (x_{i1}, \ldots, x_{iD})$ for all $i \le N$. As in [29] we remark that, for all $i < j \le N$:

$$\|x^i - x^j\|^2 = \sum_{k \le D} (x_{ik} - x_{jk})^2 = 8 - 2 \sum_{k \le D} x_{ik} x_{jk}, \tag{6.2}$$

because $\sum_{k\le D} x_{ik}^2 = \|x^i\|^2 = 4$ for all $i \le N$. Let Q be (6.1) reformulated according to (6.2): automatic detection of G_Q yields an indication that $G_Q \cong S_D \times S_N$, which is a considerably larger group. The difference lies in the fact that the binary minus is in general not commutative; however, it *is* commutative whenever it appears in terms like $\|x^i - x^j\|$ (by definition of Euclidean norm). Since automatic symmetry detection is based on expression trees, commutativity of an operator is decided at the vertex representing the operator, rather than at the parent vertex. Thus, on (6.1), our automatic system fails to detect the larger group. Reformulation (6.2) prevents this from happening, thereby allowing the automatic detection of the larger group.

Example 2. *Consider the KNP instance defined by $N = 6, D = 2$, whose variable mapping*

$$\begin{pmatrix} x_{11} & x_{12} & x_{21} & x_{22} & x_{31} & x_{32} & x_{41} & x_{42} & x_{51} & x_{52} & x_{61} & x_{62} & \alpha \\ y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_7 & y_8 & y_9 & y_{10} & y_{11} & y_{12} & y_{13} \end{pmatrix}$$

yields the following flat [31] instance:

$$\begin{array}{rcc}
\min(-y_{13}) & 2y_{13}+y_1y_3+y_2y_4 \le 4 & 2y_{13}+y_3y_{11}+y_4y_{12} \le 4 \\
y_1^2+y_2^2=4 & 2y_{13}+y_1y_5+y_2y_6 \le 4 & 2y_{13}+y_5y_7+y_6y_8 \le 4 \\
y_3^2+y_4^2=4 & 2y_{13}+y_1y_7+y_2y_8 \le 4 & 2y_{13}+y_5y_9+y_6y_{10} \le 4 \\
y_5^2+y_6^2=4 & 2y_{13}+y_1y_9+y_2y_{10} \le 4 & 2y_{13}+y_5y_{11}+y_6y_{12} \le 4 \\
y_7^2+y_8^2=4 & 2y_{13}+y_1y_{11}+y_2y_{12} \le 4 & 2y_{13}+y_7y_9+y_8y_{10} \le 4 \\
y_9^2+y_{10}^2=4 & 2y_{13}+y_3y_5+y_4y_6 \le 4 & 2y_{13}+y_7y_{11}+y_8y_{12} \le 4 \\
y_{11}^2+y_{12}^2=4 & 2y_{13}+y_3y_7+y_4y_8 \le 4 & 2y_{13}+y_9y_{11}+y_{10}y_{12} \le 4. \\
 & 2y_{13}+y_3y_9+y_4y_{10} \le 4 &
\end{array}$$

On the above instance, the `symmgroup` *system reports $G_P \cong C_2 \times S_6$, generated as:*

$$\langle (1,2)(3,4)(5,6)(7,8)(9,10)(11,12),\\ (1,3)(2,4),(3,5)(4,6),\ (5,7)(6,8),(7,9)(8,10),\ (9,11)(10,12)\rangle,$$

which, in original variable space, maps to:

$$\langle (x_{11},x_{12})(x_{21},x_{22})(x_{31},x_{32})(x_{41},x_{42})(x_{51},x_{52})(x_{61}x_{62}),\\ (x_{11},x_{21})(x_{12},x_{22}),(x_{21},x_{31})(x_{22},x_{32}),(x_{31},x_{41})(x_{32},x_{42}),\\ (x_{41},x_{51})(x_{42},x_{52}),(x_{51},x_{61})(x_{52},x_{62})\rangle,$$

or, in other words, letting $x^i = (x_{i1}, x_{i2})$ for all $i \le 6$,

$$\langle \tau, (x^1,x^2),(x^2,x^3),(x^3,x^4),(x^4,x^5),(x^5,x^6)\rangle$$

where $\tau = \prod_{i=1}^6 (x_{i1}, x_{i2})$. Carried over to the spheres in $\mathbb{R}^2$, this is a symmetric group action acting independently on the six spheres and on the two spatial dimensions.

For $N = 12, D = 3$ the formulation group is $S_3 \times S_{12}$ and for $N = 24, D = 4$ it is $S_4 \times S_{24}$. This suggests a formulation group $S_D \times S_N$ in general, where the solutions can be permuted by symmetric actions on the coordinate indices and, independently, the sphere indices. We now prove this statement formally. For all $i \le N$ call the constraints $\|x_i\|^2 = 4$ the *center* constraints and for all $i < j \le N$ call the constraints $\sum_{k \le D} x_{ik}x_{jk} \le 4 - 2\alpha$ the *distance* constraints.

THEOREM 6.1. *$G_Q \cong S_D \times S_N$.*

Proof. Let $(x, \alpha) \in \mathcal{G}(Q)$; the following claims are easy to establish.

1. For any $k \le D - 1$, the permutation $\tau_k = \prod_{i \le N}(x_{ik}, x_{i,k+1})$ is in G_Q, as both center and distance constraints are invariant w.r.t. it; notice that $\langle \tau_k \mid k \le D - 1 \rangle \cong S_D$.
2. For any $i \le N - 1$, the permutation $\sigma_i = \prod_{k \le D}(x_{ik}, x_{i+1,k})$ is in G_Q, as both center and distance constraints are invariant w.r.t. it; notice that $\langle \sigma_i \mid i \le N - 1 \rangle \cong S_N$.
3. Any permutation moving α to one of the x variables is not in G_Q. This follows because the objective function only consists of the variable α, so it is only invariant w.r.t. identity permutation.
4. For any $k \le D - 1$, if $\pi \in G_Q$ such that $\pi(x_{ik}) = x_{i,k+1}$ for some $i \le N$ then $\pi(x_{ik}) = x_{i,k+1}$ for all $i \le N$, as otherwise the term $\sum_{k \le D} x_{ik}x_{jk}$ (appearing in the distance constraints) would not be invariant.
5. For any $i \le N - 1$, if $\pi \in G_Q$ such that $\pi(x_{ik}) = x_{i+1,k}$ for some $k \le D$, then $\pi(x_{ik}) = x_{i+1,k}$ for all $k \le D$, as otherwise the term $\sum_{k \le D} x_{ik}x_{i+1,k}$ (appearing in some of the distance constraints) would not be invariant.

Let $H_D = \langle \tau_k \mid k \le D - 1 \rangle$ and $H_N = \langle \sigma_i \mid i \le N - 1 \rangle$. Claims 1-2 imply that $H_D, H_N \le G_Q$. It is easy (but tedious) to check that $H_D H_N = H_N H_D$; it follows that $H_D H_N \le G_Q$ [10] and hence H_D, H_N are normal subgroups of $H_D H_N$. Since $H_D \cap H_N = \{e\}$, we have $H_D H_N \cong H_D \times H_N \cong S_D \times S_N \le G_Q$ [2]. Now suppose $\pi \in G_Q$ with $\pi \ne e$. By Claim 3, π cannot move α so it must map x_{ih} to x_{jk} for some $i < j \le N, h < k \le D$; the action $h \to k$ (resp. $i \to j$) on the components (resp. spheres) indices can be decomposed into a product of transpositions $h \to h + 1, \ldots, k - 1 \to k$ (resp. $i \to i + 1, \ldots, j - 1 \to j$). Thus, by Claim 4 (resp. 5), π involves a certain product γ of τ_k's and σ_i's; furthermore, since by definition γ maps x_{ih} to x_{jk}, any permutation in G_Q (including π) can be obtained as a product of these elements γ; hence π is an element of $H_D H_N$, which shows $G_Q \le H_D H_N$. Therefore, $G_Q \cong S_D \times S_N$ as claimed. □

In problems involving Euclidean distances, it is often assumed that symmetries are rotations and translations of $\mathbb{R}^n$; we remark that G_Q is not necessarily isomorphic to a (finite) subgroup of $SO(D, \mathbb{R})$. Permuting two sphere indices out of N is an action in G_Q but in general there is no rotation that can act in the same way in $\mathbb{R}^D$. Hence enforcing SBCs for G_Q is not implied by simply fixing D adjacent spheres in order to break symmetries in the special orthogonal group.

By Thm. 6.1, $G_Q = \langle \tau_k, \sigma_i \mid k \leq D-1, i \leq N-1 \rangle$. It is easy to show that there is just one orbit in the natural action of G_Q on the set $A = \{1, \ldots, N\} \times \{1, \ldots, D\}$, and that the action of G_Q on A is not symmetric (otherwise G_Q would be isomorphic to S_{ND}, contradicting Thm. 6.1).

Proposition 6.1. *For any fixed $h \leq D$,*

$$\forall i \leq N \smallsetminus \{1\} \quad x_{i-1,h} \leq x_{ih} \tag{6.3}$$

are SBCs with respect to G_Q, $\mathcal{G}(Q)$.

Proof. Let $\bar{x} \in \mathcal{G}(Q)$; since the σ_i generate the symmetric group acting on the N spheres, there exists a permutation $\pi \in G_Q$ such that $(\bar{x}_{\pi(i),h} \mid i \leq N)$ are ordered as in (6.3). □

6.1. Computational results on the KNP. Comparative solutions yielded by running BARON [56] on KNP instances with and without SBC reformulation 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. These results are shown in Table 1, which contains the following statistics at termination (occurring after 10h of user CPU time):

1. the objective function value of the incumbent
2. the seconds of user CPU time taken (meaningful if < 10h)
3. the gap still open
4. the number of BB nodes closed and those still on the tree.

The first column contains the instance name in the form `knp-`N`_`D. The first subsequent set of three columns refer to the solution of the original formulations (*CPU* time, best optimal objective function value f^*, open *gap* at termination, number of *nodes* created and number of open nodes in the *tree* at termination); the second subsequent set of three columns (labelled *NarrowingKNP*) refer to the solution of the formulation obtained by adjoining (6.3) to the original formulation. The last column (*R.t.*) contains the time (in user CPU seconds) needed to automatically compute the formulation group using the methods in Sect. 4. In both formulations we fixed the first sphere at $(-2, 0, \ldots, 0)$ to break some of the orthogonal symmetries. We remark that the objective function values are negative because we are using a minimization direction (instead of maximization).

Judging from the 2-dimensional KNP instances, where BARON converges to optimality, it is evident that the *NarrowingKNP* reformulation is crucial to decrease the CPU time significantly: the total CPU time needed to solve the five 2D instances in the original formulation is 74047s, whereas the *NarrowingKNP* reformulations only take 173s, that is a gain of over 400 times. It also appears clear from the results relating to the larger instances that adjoining SBCs to the formulation makes a definite (positive) difference in the exploration rate of the search tree. The beneficial effects of the narrowing decrease with the instance size (to the extent of disappearing completely for `knp-25_4`) because we are keeping the CPU time fixed at

		Original problem			NarrowingKNP			
			*f**	*nodes*		*f**	*nodes*	
Instance	*Slv*	*CPU*	*gap*	*tree*	*CPU*	*gap*	*tree*	*R.t.*
			-1	1118		-1	**186**	
`knp-6_2`	B	8.66	**0%**	0	**1.91**	**0%**	**0**	1.43
			-0.753	13729		-0.753	**260**	
`knp-7_2`	B	147.21	0%	0	**3.86**	0%	**0**	1.47
			-0.586	179994		-0.586	**650**	
`knp-8_2`	B	1892	0%	0	**12.17**	0%	**0**	2.94
			-0.47	1502116		-0.47	**1554**	
`knp-9_2`	B	36000	33.75%	176357	**37.36**	**0%**	**0**	1.96
			-0.382	936911		-0.382	**3446**	
`knp-10_2`	B	36000	170%	167182	**117.79**	**0%**	**0**	1.97
			-1.105	299241		-1.105	**273923**	
`knp-12_3`	B	36000	8.55%	12840	36000	8.55%	**5356**	3.39
			-0.914	102150		-0.914	**68248**	
`knp-13_3`	B	36000	118%	64499	36000	118%	**33013**	3.38
			-0.966	10156		-0.92	4059	
`knp-24_4`	B	36000	**107%**	7487	36000	117%	2985	5.62
			-0.93	**7768**		-0.89	4251	
`knp-24_5`	B	36000	**116%**	**5655**	36000	124%	3122	6.1

Table 1
Computational results for the Kissing Number Problem.

10h. We remark that the effectiveness of the *NarrowingKNP* reformulation in low-dimensional spaces can be partly explained by the fact that it is designed to break sphere-related symmetries rather than dimension-related ones (naturally, the instance size also counts: the largest 2D instance, `knp-10_2`, has 21 variables, whereas the smallest 3D one, `knp-12_3`, has 37 variables).

7. Conclusion.

This paper introduces the study of symmetries in nonlinear and mixed-integer nonlinear programming. We use a generalization of the definition of formulation group given by Margot, based on transforming a mathematical programming formulation into a DAG. This allows automatic symmetry detection using graph isomorphism tools. Symmetries are then broken by means of static symmetry-breaking inequalities. We present an application of our findings to the Kissing Number Problem.

Acknowledgements. I wish to thank François Margot for many useful discussions and suggestions, and for carefully checking [33] (from which some of the present material is taken) as well as one particularly careful referee. This work was financially supported by grants: ANR 07-JCJC-0151 "ARS", Digiteo Chair 2009-14D "RMNCCO", Digiteo Emergence 2009-55D "ARM".

REFERENCES

[1] F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, 5(1):13–51, 1995.

[2] R. Allenby. *Rings, Fields and Groups: an Introduction to Abstract Algebra.* Edward Arnold, London, 1991.

[3] 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.

[4] C. Bachoc and F. Vallentin. New upper bounds for kissing numbers from semidefinite programming. *Journal of the American Mathematical Society*, 21:909–924, 2008.

[5] D. Bell. Constructive group relaxations for integer programs. *SIAM Journal on Applied Mathematics*, 30(4):708–719, 1976.

[6] 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.

[7] 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.

[8] K. Booth and C. Colbourn. Problems polynomially equivalent to graph isomorphism. Technical Report CS-77-04, University of Waterloo, 1979.

[9] M. Boulle. Compact mathematical formulation for graph partitioning. *Optimization and Engineering*, 5:315–333, 2004.

[10] A. Clark. *Elements of Abstract Algebra.* Dover, New York, 1984.

[11] D. Cohen, P. Jeavons, C. Jefferson, K. Petrie, and B. Smith. Symmetry definitions for constraint satisfaction problems. In P. van Beek, editor, *Constraint Programming*, volume 3709 of *LNCS*. Springer, 2005.

[12] A. Costa, P. Hansen, and L. Liberti. Formulation symmetries in circle packing. In R. Mahjoub, editor, *Proceedings of the International Symposium on Combinatorial Optimization*, Electronic Notes in Discrete Mathematics, Amsterdam, accepted. Elsevier.

[13] J. Crawford, M. Ginsberg, E. Luks, and A. Roy. Symmetry-breaking predicates for search problems. In *Principles of Knowledge Representation and Reasoning*, pages 148–159, Cambridge, MA, 1996. Morgan Kaufmann.

[14] Ph. Delsarte. Bounds for unrestricted codes by linear programming. *Philips Research Reports*, 27:272–289, 1972.

[15] Y. Faenza and V. Kaibel. Extended formulations for packing and partitioning orbitopes. *Mathematics of Operations Research*, 34(3):686–697, 2009.

[16] R. Fourer and D. Gay. *The AMPL Book.* Duxbury Press, Pacific Grove, 2002.

[17] 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.

[18] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.4.10*, 2007.

[19] K. Gatermann and P. Parrilo. Symmetry groups, semidefinite programs and sums of squares. *Journal of Pure and Applied Algebra*, 192:95–128, 2004.

[20] R. Gomory. Some polyhedra related to combinatorial problems. *Linear Algebra and Its Applications*, 2(4):451–558, 1969.

[21] M. Hall. *Theory of Groups.* Chelsea Publishing Company, New York, 2nd edition, 1976.

[22] P. Hansen and N. Mladenović. Variable neighbourhood search: Principles and applications. *European Journal of Operations Research*, 130:449–467, 2001.

[23] ILOG. *ILOG CPLEX 11.0 User's Manual.* ILOG S.A., Gentilly, France, 2008.

[24] E. Johnson. *Integer Programming: Facets, Subadditivity and Duality for Group and Semi-group Problems.* SIAM, Philadelphia, 1980.

[25] V. Kaibel and M. Pfetsch. Packing and partitioning orbitopes. *Mathematical Programming*, 114(1):1–36, 2008.

[26] Y. Kanno, M. Ohsaki, K. Murota, and N. Katoh. Group symmetry in interior-point methods for semidefinite program. *Optimization and Engineering*, 2:293–320, 2001.

[27] E. De Klerk and R. Sotirov. Exploiting group symmetry in semidefinite pro-

gramming relaxations of the quadratic assignment problem. *Mathematical Programming*, to appear.

[28] D.E. Knuth. *The Art of Computer Programming, Part I: Fundamental Algorithms.* Addison-Wesley, Reading, MA, 1968.

[29] S. Kucherenko, P. Belotti, L. Liberti, and N. Maculan. New formulations for the kissing number problem. *Discrete Applied Mathematics*, 155(14):1837–1841, 2007.

[30] J. Lee and F. Margot. On a binary-encoded ILP coloring formulation. *INFORMS Journal on Computing*, 19(3):406–415, 2007.

[31] 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.

[32] 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.

[33] L. Liberti. Reformulations in mathematical programming: Symmetry. Technical Report 2165, Optimization Online, 2008.

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

[35] L. Liberti. Reformulations in mathematical programming: Automatic symmetry detection and exploitation. *Mathematical Programming*, to appear.

[36] 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.

[37] 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.

[38] F. Margot. Pruning by isomorphism in branch-and-cut. *Mathematical Programming*, 94:71–90, 2002.

[39] F. Margot. Exploiting orbits in symmetric ILP. *Mathematical Programming B*, 98:3–21, 2003.

[40] F. Margot. Small covering designs by branch-and-cut. *Mathematical Programming B*, 94:207–220, 2003.

[41] F. Margot. Symmetric ILP: coloring and small integers. *Discrete Optimization*, 4:40–62, 2007.

[42] 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.

[43] B. McKay. Practical graph isomorphism. *Congressus Numerantium*, 30:45–87, 1981.

[44] B. McKay. *nauty User's Guide (Version 2.4).* Computer Science Dept. , Australian National University, 2007.

[45] C. Mears, M. Garcia de la Banda, and M. Wallace. On implementing symmetry detection. *Constraints*, 14(2009):443–477, 2009.

[46] C. Mears, M. Garcia de la Banda, M. Wallace, and B. Demoen. A novel approach for detecting symmetries in CSP models. In L. Perron and M. Trick, editors, *Constraint Programming, Artificial Intelligence and Operations Research*, volume 5015 of *LNCS*, pages 158–172, New York, 2008. Springer.

[47] O. Musin. The kissing number in four dimensions. *arXiv:math.MG/0309430v2*, April 2005.

[48] A. Neumaier. Complete search in continuous global optimization and constraint satisfaction. *Acta Numerica*, 13:271–369, 2004.

[49] 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.

[50] 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.

[51] J.-F. Puget. Automatic detection of variable and value symmetries. In P. van Beek, editor, *Constraint Programming*, volume 3709 of *LNCS*, pages 475–489, New York, 2005. Springer.

[52] 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.

[53] S. Robertson. *Polytopes and Symmetry*. Cambridge University Press, Cambridge, 1984.

[54] R.T. Rockafellar. A combinatorial algorithm for linear programs in the general mixed form. *Journal of the Society for Industrial and Applied Mathematics*, 12(1):215–225, 1964.

[55] K.H. Rosen, editor. *Handbook of Discrete and Combinatorial Mathematics*. CRC Press, New York, 2000.

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

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

[58] K. Schütte and B.L. van der Waerden. Das problem der dreizehn kugeln. *Mathematische Annalen*, 125:325–334, 1953.

[59] A. Seress. *Permutation Group Algorithms*. Cambridge University Press, Cambridge, 2003.

[60] H. Sherali and C. Smith. Improving discrete model representations via symmetry considerations. *Management Science*, 47(10):1396–1407, 2001.

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

[62] G. Szpiro. Newton and the kissing problem. *Plus magazine (online)*, 23, January 2003.

[63] A.W. Tucker. A combinatorial equivalence of matrices. In R. Bellman and M. Hall, editors, *Proceedings of the 10th Symposium of Applied Mathematics*, pages 129–140, Providence, Rhode Island, 1960. AMS.

[64] A.W. Tucker. Solving a matrix game by linear programming. *IBM Journal of Research and Development*, 4:507–517, 1960.

[65] A.W. Tucker. Combinatorial theory underlying linear programs. In L. Graves and P. Wolfe, editors, *Recent Advances in Mathematical Programming*. McGraw-Hill, New York, 1963.

[66] 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.

[67] F. Vallentin. Symmetry in semidefinite programs. *Linear Algebra and its Applications*, 430:360–369, 2009.

[68] L. Wolsey. Group representation theory in integer programming. Technical Report Op. Res. Center 41, MIT, 1969.

[69] W. Zhu. Unsolvability of some optimization problems. *Applied Mathematics and Computation*, 174:921–926, 2006.