Journal of Global Optimization manuscript No.
(will be inserted by the editor)

Stabilizer-based symmetry breaking constraints for mathematical programs

Leo Liberti · **James Ostrowski**

Received: August 20, 2013

Abstract Mathematical programs whose formulation is symmetric often take a long time to solve using Branch-and-Bound type algorithms, because of the several symmetric optima. A simple technique used in these cases is to adjoin symmetry breaking constraints to the formulation before solving the problem. These constraints: (a) aim to guarantee that at least one optimum is feasible, whilst making some of the symmetric optima infeasible; and (b) are usually associated to the different orbits of the action of the formulation group on the set of variable indices. In general, one cannot adjoin symmetry breaking constraints from more than one orbit. In [13], some (restrictive) sufficient conditions are presented which make it possible to adjoin such constraints from several orbits at the same time. In this paper we present a new, less restrictive method for the same task, and show it performs better computationally.

Keywords mathematical programming · static symmetry breaking · MILP · MINLP

1 Introduction

It is well known that Mathematical Programs (MP) with nontrivial symmetry on the decision variables may take Branch-and-Bound (BB) type solvers a very long time to prove the optimality of a solution, due to the many symmetric optima in the leaves of the BB tree [18, 14]. Two strategies are available to address this issue: *static* and *dynamic* symmetry breaking [18]. The former usually consists of adjoining new constraints to the formulation in order to make some symmetric optima infeasible, it is very easy to deploy, and is the object of the current contribution. Examples of the latter are isomorphism pruning [17] and orbital branching [21]; dynamic strategies usually require more involved implementations, but often also end up being more effective. The occurrence of symmetry in MP is far from

Leo Liberti
IBM "T.J. Watson" Research Center, Yorktown Heights, 10598 NY, USA; *and*
LIX, Ecole Polytechnique, 91128 Palaiseau, France.
E-mail: leoliberti@us.ibm.com;liberti@lix.polytechnique.fr

James Ostrowski
Dept. of Industrial and Information Engineering, University of Tennessee, Knoxville TN 37996-0700, USA.
E-mail: jostrows@utk.edu

rare: the computational experiments in [13] show that 18% of the instances in the MIPLib3 [2], MIPLib2003 [19], GlobalLib [3], MINLPLib public instance libraries have a nontrivial formulation group (defined below in Sect. 2).

Static symmetry breaking consists in adjoining some *symmetry breaking constraints* (SBC) to the problem formulation, yielding a reformulation which is guaranteed to keep at least one symmetric optimum feasible (such reformulations are called *narrowings* [12]). Several SBCs can be found in the literature for specific problems (such as, e.g., the Quadratic Assignment Problem [7], the Kissing Number Problem [14], the problem of Packing Equal Circles in a Square [6]), or specific symmetry groups (such as the full symmetric group [23]). Several SBC classes for MILPs are surveyed in [18]. Some general-purpose SBCs that work in the full Mixed-Integer Nonlinear Programming (MINLP) framework (which also includes the case of continuous Nonlinear Programs (NLP) and Mixed-Integer Quadratic Programs (MIQP)) are discussed in [13].

Although there appears to be no clearly defined relation between how large a formulation group is and how hard it is to solve the corresponding MP in practice, there are two common sense arguments motivating the study of symmetries in MP, having to do with two different algorithmic classes. When the MP is solved exactly (or approximately) using BB algorithms (such as [9] when it is a MILP or [1] when it is a MINLP) and only one global optimum is required, then multiple symmetric global optima generally yield larger BB trees, and therefore longer solution processes. When heuristic or meta-heuristic algorithms (e.g. [16]) are used in order to find good solutions of P, the picture is often reversed: if the algorithm stochastically explores the neighbourhood of the most recent (or best) found local optimum, having more optima prevents the algorithm from getting stuck early on in the search [14].

In this paper we address a limitation of the SBC generation method proposed in [13]. SBCs can be derived from each orbit of the action of the formulation group on the set of variable indices. In general, however, only SBCs referring to a single orbit can be adjoined to the formulation. If the formulation group has several orbits (as is the case in practice), however, the improvement on the BB solution time will be moderate if SBCs breaking symmetry for only one orbit are adjoined. This limitation was partly overcome in [13] by specifying sufficient conditions on subsets of orbits which makes it possible to simultaneously adjoin to the formulation SBCs from each orbit in the subset. These sufficient conditions, however, are too restrictive in practice. This paper discusses a different method for generating SBCs that can all simultaneously be adjoined to the formulation: we pick an orbit, generate the corresponding SBCs, then update the group with the orbit stabilizer; we repeat this procedure until the latter becomes trivial. We show empirically that the three proposed variants of this new method outperforms the old one on several MP instances (both MILP and MINLP) drawn from the literature, as well as on some intentionally generated MILP and MIQP edge coloring problems.

The rest of this paper is organized as follows. We give background material, formal definitions and notation in Sect. 2, present the new method and its variants in Sect. 3, and discuss the computational results in 4.

2 Background and notation

First, we give a formal definition of the type of MPs we consider. We then recall some group definitions, including the formulation group of a MP and of a graph, and finally formally introduce SBCs.

2.1 A formal definition for MP

Mathematical Programming (MP) is a formal language for describing optimization problems of the form:

$$\min\{f(x) \mid g(x) \leq 0 \wedge x \in X\}, \tag{1}$$

where $x \in \mathbb{R}^n$ is a vector of decision variables, $X \subseteq \mathbb{R}^n$ might include bounds and integrality constraints on subsequences of x, and $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}^m$ are functions that can be written as strings of a formal language $\mathscr{E}$ on the alphabet $\mathscr{A} = \mathscr{O} \cup \mathbb{Q} \cup \mathscr{V}$, where $\mathscr{V} = \{\mathsf{x}_i \mid i \in \mathbb{N}\}$ and $\mathscr{O} = \{+, -, \times, \div, (\cdot)^{(\cdot)}, \log, \exp, (,)\}$. The strings of $\mathscr{E}$ are only and all those that can be obtained by the recursive application of the following rules [6]: (a) $\forall \mathsf{s} \in \mathbb{Q} \cup \mathscr{V}$ $(\mathsf{s} \in \mathscr{E})$; (b) $\forall \otimes \in \mathscr{O}$ representing a k-ary operator and $\mathsf{e}_1, \ldots, \mathsf{e}_k \in \mathscr{E}$, $\otimes(\mathsf{e}_j \mid j \leq k)$ is in $\mathscr{E}$.

The recognition process (or parsing) of a valid string $h \in \mathscr{E}$ naturally yields a directed graph $D(h)$ whose leaf nodes are elements of $\mathbb{Q} \cup \mathscr{V}$ and whose non-leaf nodes are elements of $\mathscr{O}$ [6]. Every $P \in$ MP has a Directed Acyclic Graph (DAG) representation $D(P)$ obtained as a minor of $D(f) \cup \bigcup_{i \leq m} D(g_i)$ by contracting all equal leaf nodes [6].

For $P \in$ MP, let $\mathscr{G}(P)$ be the set of globally optimal solutions of P.

2.2 Solution and formulation groups

Let $[n] = \{1, \ldots, n\}$. For any vector $v \in \mathbb{R}^n$ and a subset $B \subseteq [n]$, we let $v[B]$ be the vector consisting of the components of v indexed by elements of B.

A group G is generated by elements of a set S (denoted $G = \langle S \rangle$) when G is the closure of S with respect to the group product. For a group G acting on X, we let $Gx = \{gx \mid g \in G\}$ be the orbit of x in G for all $x \in X$; for any $Y \subseteq X$ we let $\mathsf{stab}(Y, G) = \{g \in G \mid \forall y \in Y \ (gy \in Y)\}$ be the setwise stabilizer and $G^Y = \{g \in G \mid \forall y \in Y \ (gy = y)\}$ be the pointwise stabilizer of Y w.r.t. G. For a permutation $\pi \in S_n$ let $\Gamma(\pi)$ be the set of all the cycles in its (unique) disjoint cycle representation. For $N \subseteq [n]$ we define $\pi[N] = \prod_{\sigma \in \Gamma(\pi) \cap \mathsf{stab}(N, S_n)} \sigma$ to be the restriction of π to N. If $\pi_1, \ldots, \pi_k \in G \leq S_n$ are generators for G (i.e. $G = \langle \pi_j \mid j \leq k \rangle$) we define $G[N] = \langle \pi_j[N] \mid j \leq k \rangle$ to be the restriction of G to N. If N is an orbit of the action of G on $[n]$ then $G[N]$ is a *transitive constituent* of G w.r.t. N. By [5], p. 35, for each orbit ω of G, the (right) map $\cdot[\omega] : G \to G[\omega]$ given by $\pi \to \pi[\omega]$ is a group homomorphism whose kernel is the pointwise stabilizer G^ω, which is therefore a normal subgroup of G.

We remark that stabilizers are usually denoted by G_Y rather than G^Y, and transitive constituents by G^N [5, 22]. Our nonstandard notation allows us to disambiguate expressions such as G_P as concerns the two possible interpretations "the formulation group of P" and "the stabilizer of the set P in G".

We consider the action of $G \leq S_n$ on X given by $\pi x = (x_{\pi^{-1}(i)} \mid i \leq n)$. This action induces a right action $P \mapsto P\pi$ (where $\pi \in G$) on MP by replacing x with πx everywhere in (1). S_m also induces a left action $P \mapsto \sigma P$ (where $\sigma \in S_m$) given by replacing $g = (g_1, \ldots, g_m)$ by $\sigma g = (g_{\sigma^{-1}(j)} | j \leq m)$. Because optimization problems (1) are independent of the order of the constraint sequence g, the two problems σP and P have the same feasible region and optima (both local and global) for all $\sigma \in S_m$, which implies that $(\sigma P)\pi$ and $P\pi$ also have the same feasible region and optima for $\sigma \in S_m, \pi \in S_n$, whence $\forall \sigma \in S_m, \pi \in S_n \ (\sigma P)\pi = \sigma(P\pi)$.

We define the *solution group* $G^*(P) = \mathsf{stab}(\mathscr{G}(P), S_n)$ and the *formulation group* $G_P = \langle \pi \in S_n \mid \exists \sigma \in S_m \ (\sigma P \pi = P) \rangle$ of a MP formulation P given by (1); it is easy to show that $G_P \leq G^*(P)$. Computing the solution group in general requires aprioristic knowledge of $\mathscr{G}(P)$, which is usually the ultimate aim when considering and solving MPs, and is therefore

impractical. Since deciding whether two function encodings $h_1, h_2 \in \mathscr{E}$ are equal has linear complexity in $|D(P)|$, computing generators for G_P is a decidable problem [13] that can be solved once the formulation of P is known. By choosing an appropriate colouring $\gamma : D(P) \to \mathbb{N}$ of the vertices of $D(P)$ (in order to avoid permutations of nodes of different types, e.g., operator nodes with variable nodes), we show that $G_P = \mathsf{Aut}(D(P), \gamma)[N]$, where $\mathsf{Aut}(\mathscr{G}, \delta)$ is the group of automorphisms of the graph $\mathscr{G}$ which stabilizes each equivalence class given by the vertex colouring δ setwise [13].

2.3 Symmetry breaking constraints

Formally, a *symmetry breaking constraint* (SBC) for problem P and a group $G \leq G_P$ is a set of constraints $g(x) \leq 0$ (where $g : \mathbb{R}^n \to \mathbb{R}^p$ for some $p \in \mathbb{N}$) such that there exists an optimum $y \in \mathscr{G}(P)$ and $\pi \in G$ with $g(\pi y) \leq 0$ [13]. Strictly speaking, this definition does not force SBCs to actually break any symmetry, but only to keep at least one optimum feasible; this ensures the definition also works with optima y which are invariant to all permutations. In practice, however, SBCs are constructed in such a way that $g(\sigma y) > 0$ for as many $\sigma \in G$ as possible, so as to make as many symmetric optima in $\mathscr{G}(P)$ as possible infeasible in the narrowing [11]. If $g(x) \leq 0$ are SBCs involving only variables x_j with j in a given set B, we emphasize this by writing $g[B](x) \leq 0$, and say $g(x) \leq 0$ are *SBCs with respect to* B. G is taken to be the whole of G_P unless specified otherwise.

In [13], new methods were presented in order to break symmetries with two types of general-purpose SBC derived from the set Ω of orbits of the action of G_P on the variable index set $[n]$. Specifically, for a nontrivial orbit $\omega \in \Omega$, if the transitive constituent $G_P[\omega]$ can be ascertained to be isomorphic to the full symmetric group $\mathsf{Sym}(\omega)$ on ω, then a unique order can be imposed on the variables indexed by ω by adjoining the following linear inequalities to P:

$$\forall j \in \omega \smallsetminus \{\max \omega\} \quad x_j \leq x_{j^+}, \tag{2}$$

where j^+ is the successor of j in ω. Otherwise, for any structure $G_P[\omega]$ might have, one can always choose a variable (for example $x_{\min \omega}$) that should have minimum values among all those indexed by ω:

$$\forall j \in \omega \smallsetminus \{\min \omega\} \quad x_{\min \omega} \leq x_j. \tag{3}$$

We remark that the choices of $\max \omega$ and j^+, which define the orbit order, and of $\min \omega$, which defines the element having minimum value in the orbit, are arbitrary.

In general, if $\omega, \theta \in \Omega$ with $\omega \neq \theta$ and $g^\omega(x) \leq 0$ and $g^\theta(x) \leq 0$ are SBCs with respect to, respectively, ω and θ, adjoining both $g^\omega(x) \leq 0$ and $g^\theta(x) \leq 0$ to P may not yield a valid narrowing of P, as Example 1 shows.

Example 1 Let P be the following MILP:

$$\left.\begin{array}{c} \min\limits_{x \in \{0,1\}^4} \quad x_1 + x_2 + 2x_3 + 2x_4 \\ \begin{pmatrix} -1 & -1 & 0 & 0 \\ 0 & 0 & -1 & -1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \geq \begin{pmatrix} -1 \\ -1 \\ 1 \\ 1 \end{pmatrix}. \end{array}\right\}$$

Its formulation group is $G_P = \langle (1,2)(3,4) \rangle \cong C_2$, with two orbits $\omega_1 = \{1,2\}$ and $\omega_2 = \{3,4\}$ and optima $\mathscr{G}(P) = \{(0,1,1,0), (1,0,0,1)\}$. Valid SBCs for ω_1 (resp. ω_2) are $x_1 \leq x_2$ (resp. $x_3 \leq x_4$). Whereas adjoining either of the two SBCs leads to a valid narrowing, adjoining both at the same time results in an infeasible problem.

2.4 Coprime orbits narrowing

We showed in [13] some sufficient conditions by which SBCs originating from different orbits could be combined into a valid narrowing of P. By Cor. 14 in [13], this holds if:

- $G_P[\omega \cup \theta]$ contains a subgroup H such that $H[\omega] \cong C_{|\omega|}$ and $H[\theta] \cong C_{|\theta|}$ (where C_p is the cyclic group of order p for all $p \in \mathbb{N}$);
- $\gcd(|\omega|, |\theta|) = 1$.

Adjoining such SBCs to the original formulation results into a reformulation which we call the *coprime orbits narrowing*.

We remark that these conditions are restrictive but not necessary, as Example 2 shows.

Example 2 The problem $P \equiv \min\{\sum_{j\leq 6} x_j \mid x_1 + x_2 + 2\sum_{3\leq j\leq 6} x_j \geq 3 \wedge x \in \{0,1\}^6\}$ has formulation group $G_P = \langle (1,2),(3,4),(4,5),(5,6)\rangle$. The action of G on $\{1,\ldots,6\}$ has the two orbits $\omega_1 = \{1,2\}$ and $\omega_2 = \{3,4,5,6\}$, yielding SBCs $x_1 \leq x_2$ and, respectively, $x_3 \leq x_4$, $x_3 \leq x_5$, $x_3 \leq x_6$. These two sets of SBCs can both be adjoined to P at the same time, even though $\gcd(|\omega_1|, |\omega_2|) = 2 \neq 1$.

3 A new method for SBC generation

Consider the SBC set C generated by Alg. 1. This algorithm iteratively builds a sequence

Algorithm 1 Orbit stabilizer based SBC generator

1: Let $G = G_P$ and $C = \varnothing$;
2: **repeat**
3: Let Ω be the set of orbits of the action of G on $[n]$;
4: Choose an orbit $\omega \in \Omega$;
5: Let $g[\omega](x) \leq 0$ be some SBCs for P and G w.r.t. ω;
6: Let $C = C \cup \{g[\omega](x) \leq 0\}$;
7: Replace G with the (pointwise) stabilizer G^{ω} of ω
8: **until** $G = \{e\}$
9: Return C.

$\omega_1, \ldots, \omega_k$ of subsets of $[n]$ with associated SBCs $g_\ell[\omega_\ell](x) \leq 0$, and a chain of normal subgroups

$$G_P = G_1 \rhd G_1^{\omega_1} = G_2 \rhd \ldots \rhd G_{k-1}^{\omega_{k-1}} = G_k \rhd G_k^{\omega_k} = \{e\}. \tag{4}$$

It is easy to see that $G_\ell = G_P^{\bigcup_{i<\ell}\omega_i}$ for each $\ell \leq k$, i.e. permutations in G_ℓ stabilize all elements of the orbits $\omega_1, \ldots, \omega_{\ell-1}$ pointwise.

Theorem 3 *The constraint set $C_k = \{g_\ell[\omega_\ell](x) \leq 0 \mid \ell \leq k\}$ is an SBC system for P.*

Proof If P is infeasible then adjoining the constraints in C to P does not change its infeasibility, so assume P is feasible. We prove the result by induction on k. When $k = 1$ the result trivially holds because $g_1[\omega_1](x) \leq 0$ are SBCs for P by construction. Let P' be like P with all the constraints in C_{k-1} adjoined to it. By the induction hypothesis C_{k-1} is an SBC set for P, so $\mathscr{G}(P') \subseteq \mathscr{G}(P)$ is non-empty; and since $k-1$ is not the last iteration of the algorithm, G_k is a nontrivial group. Since $g_k[\omega_k](x) \leq 0$ are SBCs for P' and G_k with respect to ω_k, there exist $y \in \mathscr{G}(P')$ and $\pi \in G_k$ such that $g_k[\omega_k](\pi y) \leq 0$. Since $\pi \in G_k$, it stabilizes the orbits

$\omega_1, \ldots, \omega_{k-1}$ pointwise, which implies that $(\pi y)[\omega_\ell] = y[\omega_\ell]$ for all $\ell < k$. Since $y \in \mathscr{G}(P')$, y is feasible in P', so it satisfies all constraints in C_{k-1}: this means $\forall \ell < k\ g[\omega_\ell](\pi y) \leq 0$, which concludes the proof.

Adjoining C to the original formulation yields a reformulation which we call the *orbit stabilizer narrowing*.

We remark that there is a choice of ω in Step 4 which guarantees that the stabilizer narrowing is always "stronger" than the coprime orbits one: i.e., by choosing coprime orbits first. The notion of strength we consider here is based on the $\subseteq$ relation on the sets of SBCs generated by the two narrowings: a set S is stronger than S' if $S' \subseteq S$. It was observed empirically, however, that best results are obtained when ω is chosen as the orbit of smallest cardinality in Ω (intuitively, this is likely to increase the length of the chain in Eq. (4) and therefore the size of the SBC set), which is the policy we follow below.

3.1 Variants

We propose two variants of Alg. 1 that exploit the structure of the SBCs (2)-(3).

First observe that whenever the structure of the transitive constituent $G[\omega]$ is unknown, we employ SBCs (3). Their effect is weak, in the sense that we only impose that $x_{\min \omega}$ takes minimum value among all x_j (for $j \in \omega$). Any permutation π in the stabilizer $G[\omega]^{\min \omega}$ of the variable index $\min \omega$ will still yield several symmetric optima. This suggests a variant to Alg. 1 where G^ω appearing in Step 7 is replaced by $G^{\min \omega}$, i.e. at each iteration G is assigned the stabilizer of the index $\min \omega$ instead of the whole orbit stabilizer. This leads to longer chains (4) and hopefully larger SBC sets C. Adjoining C so the original formulation yields a reformulation which we call the *point stabilizer narrowing*.

Now suppose $G[\omega] \cong \mathrm{Sym}(\omega)$, where $\omega = \{i_1, \ldots, i_h\}$ is assumed to be sorted. Applying the point stabilizer narrowing will result in the following SBCs: $x_{i_1} \leq x_{i_2}, \ldots, x_{i_1} \leq x_{i_h}$ from G_P, $x_{i_2} \leq x_{i_3}, \ldots, x_{i_2} \leq x_{i_h}$ from $G_P^{i_1}$, $x_{i_3} \leq x_{i_4}, \ldots, x_{i_3} \leq x_{i_h}$ from $G_P^{i_2}$, and so on, up to $x_{i_{h-1}} \leq x_{i_h}$. In other words, we obtain SBCs (2) plus a set of several other dominated constraints. It is more efficient to detect whether $G[\omega] \cong \mathrm{Sym}(\omega)$ and, if so, generate SBCs (2) in Step 5. Step 7 is adjusted as follows: we replace G by G^ω whenever we generate SBCs (2), and by $G^{\min \omega}$ if we generate SBCs (3) in Step 5. The resulting reformulation is called *hybrid stabilizer narrowing*.

The proofs that both these variants yield valid narrowings are very similar to that of Thm. 3.

4 Computational results

We compare the effect that the coprime orbits narrowing (Sect. 2.4) and the stabilizer based narrowings (Sect. 3) have on BB. We present two sets of tests: one on MILPs, NLPs and MINLPs from public instance libraries, and another one on MILP and MINLP edge coloring problems defined on flower snark graphs [10]. All results were obtained on a quad-CPU Intel Xeon at 2.66 GHz with 24GB RAM. Automatic group detection has been carried out using the ROSE reformulator [15] and `nauty` [20]. Other group computations have been carried out using GAP v. 4.4.10 [8].

It turns out that the new methods for generating SBCs outperform the old one. The difference is particularly striking when BB is used to detect infeasibility of symmetric MILPs.

4.1 Publically available instances

Our test set consists of symmetric MILPs taken from François Margot's symmetric MILP repository `http://wpweb2.tepper.cmu.edu/fmargot/prob_inst.html`, symmetric NLPs taken from the GlobalLib [3] and symmetric MINLPs taken from the MINLPLib [4]. We solve MILPs using CPLEX 12.2 [9], and NLP/MINLPs using COUENNE 0.4. Both solvers are configured using their default settings. The time limit was set to 1800s of user CPU time. The time data are those reported by the operating system. For CPLEX, these take into account the time taken by all CPUs (since CPLEX 12.2 runs in parallel by default), which is usually close to 7000s.

We benchmark four SBC-based narrowings against the original formulation: coprime orbits, orbit, point, and hybrid stabilizer. In each instance of Table 1 we report: the best optimum value found, the CPU time, the gap still open at termination, and the solution status (`opt`=optimum found, `inf`=infeasible instance, `lim`=time limit reached). Best values are emphasized in boldface. We remark that all instances for which all formulations found global optima in CPU times differing by less than 1s were discarded from Table 1.

It appears clear that the hybrid stabilizer narrowing is best on MILP and performs well on NLP (the MINLP test set is too small to draw any conclusion). It is interesting to remark that the original formulation spectacularly beats the others, CPU time-wise, in two MILP formulations: `cov1053` and `pa57245`. We think the reason for these occurrences is due to all our reformulations containing some arbitrary choices (e.g. we arbitrarily decide to employ $\min \omega$ in (3), when in fact any index in ω would do). This unduly influences CPLEX's branching decisions, resulting in poor performance. This seems to be an inherent limitation of all SBCs in the literature.

In Table 2 we present aggregated statistics: the global CPU time Δ_{CPU} saved (or wasted, if negative) by a given reformulation with respect to the original one, and the number of instances on which the reformulation is best, by best optimum value found, CPU time, closed gap or preferable solution status (e.g. `opt` is better than `lim`). The aggregated statistics confirm the superiority of the hybdrid stabilizer based narrowing with respect to the others.

4.2 Edge coloring problems on flower snarks

The stabilizer-based SBC generation method described in Sect. 3 (and its variants, Sect. 3.1) exploits stabilizer chains such as the one given in Eq. (4) in order to generate SBCs, a set thereof per stabilized element (either orbit or point). It should be intuitively clear, then, that the algorithm stands a greater chance of success on MPs whose formulation group has long stabilizer chains. Flower snarks, first defined in [10], form an infinite family of biconnected cubic graphs whose automorphism groups usually contain long stabilizer chains. Flower snark graphs are denoted by J_k, for $k \geq 3$, and are constructed in the following way (see `en.wikipedia.org/wiki/Flower_snark`):

1. draw k copies of a star on four vertices
2. label the vertices with star size 3 by $O_1, \ldots, O_k$
3. label the other vertices as $A_1, \ldots, A_k, B_1, \ldots, B_k, C_1, \ldots, C_k$
4. draw an n-cycle $A_1, \ldots, A_k$
5. draw a $2n$-cycle $B_1, \ldots, B_k, C_1, \ldots, C_k$.

It was shown in [10] that it takes four colors to color the edges of flower snarks of odd order k in such a way that no two adjacent edges are assigned the same color c. An edge coloring

	Original				Coprime orbits				Orbit stabilizer				Point stabilizer				Hybrid stabilizer			
Instance	*Best*	*Time*	*Gap*	*St.*	*Best*	*Time*	*Gap*	*St.*	*Best*	*Time*	*Gap*	*St.*	*Best*	*Time*	*Gap*	*St.*	*Best*	*Time*	*Gap*	*St.*
MILP																				
04_35	∞	0.11	∞	inf	∞	227.65	∞	inf	∞	228.18	∞	inf	∞	1.68	∞	inf	∞	**0.08**	∞	inf
ca57245	82	357.45	0%	opt	82	104.27	0%	opt	82	104.22	0%	opt	82	**70.56**	0%	opt	82	105.17	0%	opt
ca77247	113	432.76	0%	opt	113	309.14	0%	opt	113	309.26	0%	opt	113	51.18	0%	opt	113	**36.45**	0%	opt
cod105	-12	**104.12**	0%	opt	-12	104.15	0%	opt	-12	104.19	0%	opt	-12	1494.25	0%	opt	-12	1413.86	0%	opt
cod105r	-11	**17.52**	0%	opt	-11	17.61	0%	opt	-11	27.82	0%	opt	-11	414.51	0%	opt	-11	27.83	0%	opt
cod83	-20	**81.02**	0%	opt	-20	81.08	0%	opt	-20	81.05	0%	opt	-20	293.43	0%	opt	-20	266.23	0%	opt
cod83r	-19	39.18	0%	opt	-19	28.66	0%	opt	-19	**27.03**	0%	opt	-19	938.93	0%	opt	-19	27.06	0%	opt
cod93	-40	6706.01	10.17%	lim	-40	6705.44	10.17%	lim	-40	6699.33	10.17%	lim	-40	6690.06	**10.12%**	lim	-40	6690.32	**10.12%**	lim
cod93r	-39	6944.72	4.82%	lim	-39	6916.93	6.92%	lim	-39	4697.75	0%	opt	-39	6633.27	10.94%	lim	-39	**4705.74**	**0%**	opt
cov1053	17	**1703.23**	0%	opt	17	3757.57	0%	opt	17	3749.71	0%	opt	17	6952.49	5.88%	lim	17	6941.96	5.88%	lim
cov1054	51	6922.25	1.96%	lim	51	6917.58	1.96%	lim	51	6906.69	1.96%	lim	51	6972.76	1.96%	lim	51	6935.24	1.96%	lim
cov1075	20	34.70	0%	opt	20	19.46	0%	opt	20	**19.38**	0%	opt	20	276.40	0%	opt	20	299.52	0%	opt
cov1076	45	6704.63	3.03%	lim	45	6847.38	2.64%	lim	45	6813.22	2.64%	lim	45	1650.59	0%	opt	45	**1029.96**	**0%**	opt
cov1174	17	6609.14	7.56%	lim	17	6563.15	7.56%	lim	17	6574.31	7.56%	lim	17	6590.64	7.56%	lim	17	6549.60	7.56%	lim
cov954	30	3.71	0%	opt	30	3.11	0%	opt	30	**3.08**	0%	opt	30	4.43	0%	opt	30	4.85	0%	opt
flosn52	∞	6.32	∞	inf	∞	11.58	∞	inf	∞	**0.00**	∞	inf	∞	0.01	∞	inf	∞	**0.00**	∞	inf
flosn60	∞	16.48	∞	inf	∞	40.44	∞	inf	∞	**0.00**	∞	inf	∞	0.01	∞	inf	∞	0.01	∞	inf
jgt30	∞	**57.78**	∞	inf	∞	209.29	∞	inf	∞	131.77	∞	inf	∞	120.42	∞	inf	∞	131.67	∞	inf
mered	∞	1.63	∞	inf	∞	8.80	∞	inf	∞	**0.98**	∞	inf	∞	5.89	∞	inf	∞	1.78	∞	inf
oa36243	∞	**0.22**	∞	inf	∞	1.25	∞	inf	∞	1.19	∞	inf	∞	0.92	∞	inf	∞	0.14	∞	inf
oa57245	∞	**0.47**	∞	inf	∞	2.98	∞	inf	∞	2.97	∞	inf	∞	3.74	∞	inf	∞	3.29	∞	inf
oa77247	∞	143.88	∞	inf	∞	**4.40**	∞	inf	∞	**4.40**	∞	inf	∞	6.92	∞	inf	∞	10.24	∞	inf
ofsub9	∞	382.36	∞	inf	∞	144.10	∞	inf	∞	288.73	∞	inf	∞	80.74	∞	inf	∞	**51.28**	∞	inf
pa57245	-76	**476.19**	0%	opt	-76	4382.17	0%	opt	-76	4384.69	0%	opt	-76	1421.99	0%	opt	-76	2301.60	0%	opt
pa77247	-108	6637.25	2.37%	lim	-108	6612.15	**2.24%**	lim	-108	6613.15	**2.24%**	lim	-108	6642.54	2.34%	lim	-108	6612.48	2.36%	lim
sts45	30	10.25	0%	opt	30	10.28	0%	opt	30	10.23	0%	opt	30	7.51	0%	opt	30	**7.49**	0%	opt
sts63	45	872.66	0%	opt	45	875.14	0%	opt	45	874.44	0%	opt	45	**116.67**	0%	opt	45	117.12	0%	opt
sts81	61	5651.01	13.93%	lim	61	5919.95	13.66%	lim	61	5911.77	13.66%	lim	61	503.30	0%	opt	61	**502.53**	0%	opt
sts135	105	6441.95	26.19%	lim	104	6128.68	25.80%	lim	104	6179.64	25.79%	lim	104	4261.48	24.52%	lim	104	4305.37	**24.36%**	lim
NLP																				
ex5_2_5	-3.5e3	1.8e3	89%	lim	-3.5e3	1.8e3	**82%**	lim	-3.5e3	1.8e3	82%	lim	-3.5e3	1.8e3	86%	lim	-3.5e3	1.8e3	**82%**	lim
ex6_1_1	-0.02	85	0%	opt	-0.02	**22**	0%	opt	-0.02	**22**	0%	opt	-0.02	**22**	0%	opt	-0.02	**22**	0%	opt
ex6_1_3	-0.35	3.5e2	0%	opt	-0.35	**1.9e2**	0%	opt	-0.35	**1.9e2**	0%	opt	-0.35	**1.9e2**	0%	opt	-0.35	**1.9e2**	0%	opt
ex6_2_9	-0.035	2.2e2	0%	opt	-0.035	**63**	0%	opt	-0.035	**63**	0%	opt	-0.035	**63**	0%	opt	-0.035	**63**	0%	opt
ex6_2_10	-3.1	1.8e3	2.9e2%	lim	-3.1	1.8e3	**1.2e2%**	lim	-3.1	1.8e3	**1.2e2%**	lim	-3.1	1.8e3	**1.2e2%**	lim	-3.1	1.8e3	**1.2e2%**	lim
ex6_2_12	0.29	47	0%	opt	0.29	**16**	0%	opt	0.29	**16**	0%	opt	0.29	**16**	0%	opt	0.29	**16**	0%	opt
ex6_2_13	-0.22	1.8e3	7.2e2%	lim	-0.22	1.8e3	**2.8e2%**	lim	-0.22	1.8e3	**2.8e2%**	lim	-0.22	1.8e3	**2.8e2%**	lim	-0.22	1.8e3	**2.8e2%**	lim
ex6_2_14	-0.7	7.9	0%	opt	-0.7	**3.1**	0%	opt	-0.7	3.2	0%	opt	-0.7	3.2	0%	opt	-0.7	3.2	0%	opt
ex8_3_12	-8.8e-06	1.8e3	1.3e8%	lim	-0.00054	1.8e3	1.9e6%	lim	-0.79	1.8e3	1.2e3%	lim	**-0.93**	1.5e3	**9.7e2%**	lim	-0.79	1.8e3	1.2e3%	lim
ex9_1_8	-3.2	3.6	0%	opt	-3.2	0.055	0%	opt	-3.2	**0.053**	0%	opt	-3.2	0.055	0%	opt	-3.2	0.058	0%	opt
ex9_1_10	-3.2	3.6	0%	opt	-3.2	**0.056**	0%	opt	-3.2	**0.056**	0%	opt	-3.2	**0.056**	0%	opt	-3.2	0.057	0%	opt
st_rv9	-1.2e2	5.3	0%	opt	-1.2e2	4.4	0%	opt	-1.2e2	**3.2**	0%	opt	-1.2e2	**3.2**	0%	opt	-1.2e2	**3.2**	0%	opt
MINLP																				
elf	0.19	5.7	0%	opt	0.19	**1.7**	0%	opt	0.19	**1.7**	0%	opt	0.19	**1.7**	0%	opt	0.19	**1.7**	0%	opt
synheat	∞	1.8e3	∞	lim	**1.5e5**	1.8e3	6.6%	lim	**1.5e5**	1.8e3	6.6%	lim	**1.5e5**	1.8e3	**6.4%**	lim	**1.5e5**	1.8e3	6.6%	lim

Table 1 MILP results obtained using CPLEX, NLP/MINLP using COUENNE.

	Original	Coprime orbits		Orbit stabilizer		Point stabilizer		Hybrid stabilizer	
Dataset	# Best	Δ_{CPU}	# Best	Δ_{CPU}	# Best	Δ_{CPU}	# Best	Δ_{CPU}	# Best
MILP	8	-5601	2	-3390	8	5151	3	**8280**	**10**
NLP	2	423	**9**	425	**9**	**724**	**9**	425	8
MINLP	0	**4**	**2**	0	**2**	0	**2**	0	**2**
Total	10	-5173	13	-2965	19	5876	15	**8705**	**20**

Table 2 Aggregated solution statistics (the higher, the better). Negative Δ_{CPU}: reformulation performance is slower by given amount.

problem on odd flower snarks with maximum edge chromatic number 3 therefore yields infeasible instances.

In this section we present computational results of two types over the flower snarks $J_k = (V,E)$.

– A feasibility edge-coloring problem on a graph $G = (V,E)$ colored with 3 colours (the index c ranges over the colours $\{1,2,3\}$):

$$\left.\begin{array}{rl} \forall u \in V, c \leq 3 & \sum\limits_{\{u,v\}\in E} x_{uvc} \leq 1 \\ \forall \{u,v\} \in E & \sum\limits_{c\leq 3} x_{uvc} = 1. \end{array}\right\} \tag{5}$$

– A nonlinear cost edge-coloring problem where we pay quadratically for the number of edges colored with the fourth color (the index c may range over $\{1,\ldots,4\}$):

$$\left.\begin{array}{rl} \min\limits_{x\in\{0,1\}^{4|E|}} & \sum\limits_{\{u,v\}\in E}\sum\limits_{c\leq 3} x_{uvc} + \left(\sum\limits_{\{u,v\}\in E} x_{uv4}\right)^2 \\ \forall u \in V, c \leq 4 & \sum\limits_{\{u,v\}\in E} x_{uvc} \leq 1 \\ \forall \{u,v\} \in E & \sum\limits_{c\leq 4} x_{uvc} = 1. \end{array}\right\} \tag{6}$$

Problem (5) (which the instances `flosn52`, `flosn60` that appear in Table 1 belong to) is infeasible on all flower snarks of odd order. Since detecting infeasibility is a typical task that BB solvers are used for, (5) provides an interesting MILP test for our method. Problem (6) consists of a set of feasible (convex) MIQP instances. Both problems were solved using CPLEX 12.2. The results are reported in Tables 3.

The reason for the stunning success of the stabilizer-based reformulation on the infeasibility detection test is that, in each case, the set of generated SBCs was rich enough for CPLEX to detect infeasibility at the presolver stage, i.e. no branching was involved.

Acknowledgments

The first author was partially supported by Digiteo Chair grant 2009-14D "RMNCCO" and Digiteo Emergence grant 2009-55D "ARM".

References

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

	Original				Coprime orbits				Orbit stabilizer				Point stabilizer				Hybrid stabilizer			
Instance	*Best*	*Time*	*Gap*	*St.*	*Best*	*Time*	*Gap*	*St.*	*Best*	*Time*	*Gap*	*St.*	*Best*	*Time*	*Gap*	*St.*	*Best*	*Time*	*Gap*	*St.*
MILP																				
`flowersnark-9`	∞	0.29	∞	inf	∞	1.77	∞	inf	∞	1.16	∞	inf	∞	**0.19**	∞	inf	∞	0.19	∞	inf
`flowersnark-11`	∞	1.73	∞	inf	∞	5.07	∞	inf	∞	**0.01**	∞	inf	∞	**0.01**	∞	inf	∞	**0.01**	∞	inf
`flowersnark-13`	∞	4.73	∞	inf	∞	20.09	∞	inf	∞	**0.01**	∞	inf	∞	**0.01**	∞	inf	∞	**0.01**	∞	inf
`flowersnark-15`	∞	7.36	∞	inf	∞	77.80	∞	inf	∞	**0.00**	∞	inf	∞	0.02	∞	inf	∞	0.02	∞	inf
`flowersnark-17`	∞	72.96	∞	inf	∞	244.31	∞	inf	∞	**0.01**	∞	inf	∞	0.04	∞	inf	∞	0.02	∞	inf
`flowersnark-19`	∞	159.20	∞	inf	∞	1549.11	∞	inf	∞	**0.01**	∞	inf	∞	0.05	∞	inf	∞	0.03	∞	inf
`flowersnark-21`	∞	787.85	∞	inf	∞	5114.75	∞	inf	∞	**0.01**	∞	inf	∞	0.04	∞	inf	∞	0.04	∞	inf
`flowersnark-23`	∞	1502.54	∞	inf	∞	7086.59	∞	lim	∞	**0.01**	∞	inf	∞	0.09	∞	inf	∞	0.06	∞	inf
MIQP																				
`flowersnark-5`	32	28.48	0%	opt	32	37.42	0%	opt	32	16.30	0%	opt	32	**5.32**	0%	opt	32	6.73	0%	opt
`flowersnark-7`	44	343.17	0%	opt	44	186.54	0%	opt	44	110.59	0%	opt	44	**62.45**	0%	opt	44	68.10	0%	opt
`flowersnark-9`	56	2499.80	0%	opt	56	1073.01	0%	opt	56	847.15	0%	opt	56	364.74	0%	opt	56	**288.89**	0%	opt

Table 3 Infeasible MILP and feasible MIQP on flower snark graphs.

2. Bixby, R., Ceria, S., McZeal, C., Savelsbergh, M.: An updated mixed integer programming library: Miplib 3. Tech. Rep. TR98-03, Rice University (1998)
3. Bussieck, M.: Globallib — a collection of nonlinear programming problems (2004). (`http://www.gamsworld.org/global/globallib.htm`)
4. Bussieck, M., Drud, A., Meeraus, A.: MINLPLib — A collection of test models for mixed-integer nonlinear programming. INFORMS Journal on Computing **15**(1) (2003)
5. Cameron, P.: Polynomial aspects of codes, matroids and permutation groups (online). Lecture notes
6. Costa, A., Hansen, P., Liberti, L.: Formulation symmetries in circle packing. In: R. Mahjoub (ed.) Proceedings of the International Symposium on Combinatorial Optimization, *Electronic Notes in Discrete Mathematics*, vol. 36, pp. 1303–1310. Elsevier, Amsterdam (2010)
7. Fischetti, M., Monaci, M., Salvagnin, D.: Three ideas for the quadratic assignment problem. Operations Research **60**, 954–964 (2012)
8. The GAP Group: GAP – Groups, Algorithms, and Programming, Version 4.4.10 (2007). URL `http://www.gap-system.org`
9. IBM: ILOG CPLEX 12.2 User's Manual. IBM (2010)
10. Isaacs, R.: Infinite families of nontrivial trivalent graphs which are not Tait colorable. American Mathematical Monthly **82**(3), 221–239 (1975)
11. Liberti, L.: Automatic generation of symmetry-breaking constraints. In: B. Yang, D.Z. Du, C. Wang (eds.) Combinatorial Optimization, Constraints and Applications (COCOA08), *LNCS*, vol. 5165, pp. 328–338. Springer, Berlin (2008)
12. Liberti, L.: Reformulations in mathematical programming: Definitions and systematics. RAIRO-RO **43**(1), 55–86 (2009)
13. Liberti, L.: Reformulations in mathematical programming: Automatic symmetry detection and exploitation. Mathematical Programming A **131**, 273–304 (2012). DOI 10.1007/s10107-010-0351-0
14. Liberti, L.: Symmetry in mathematical programming. In: J. Lee, S. Leyffer (eds.) Mixed Integer Nonlinear Programming, *IMA*, vol. 154, pp. 263–286. Springer, New York (2012)
15. Liberti, L., Cafieri, S., Savourey, D.: Reformulation optimization software engine. In: K. Fukuda, J. van der Hoeven, M. Joswig, N. Takayama (eds.) Mathematical Software, *LNCS*, vol. 6327, pp. 303–314. Springer, New York (2010)
16. Liberti, L., Mladenović, N., Nannicini, G.: A good recipe for solving MINLPs. In: V. Maniezzo, T. Stützle, S. Voß (eds.) Hybridizing metaheuristics and mathematical programming, *Annals of Information Systems*, vol. 10, pp. 231–244. Springer, New York (2009)
17. Margot, F.: Pruning by isomorphism in branch-and-cut. Mathematical Programming **94**, 71–90 (2002)
18. Margot, F.: Symmetry in integer linear programming. In: M. Jünger, T. Liebling, D. Naddef, G. Nemhauser, W. Pulleyblank, G. Reinelt, G. Rinaldi, L. Wolsey (eds.) 50 Years of Integer Programming, pp. 647–681. Springer, Berlin (2010)
19. Martin, A., Achterberg, T., Koch, T.: Miplib 2003. Tech. Rep. 05-28, ZIB (2005)
20. McKay, B.: *nauty* User's Guide (Version 2.4). Computer Science Dept. , Australian National University (2007)
21. Ostrowski, J., Linderoth, J., Rossi, F., Smriglio, S.: Orbital branching. Mathematical Programming **126**, 147–178 (2011)
22. Seress, A.: Permutation Group Algorithms. Cambridge University Press, Cambridge (2003)
23. Sherali, H., Smith, C.: Improving discrete model representations via symmetry considerations. Management Science **47**(10), 1396–1407 (2001)