Orbital shrinking

Matteo Fischetti[1] and Leo Liberti[2]

[1] DEI, Università di Padova, Italy
matteo.fischetti@unipd.it
[2] LIX, École Polytechnique, 91128 Palaiseau, France
liberti@lix.polytechnique.fr

Abstract. Symmetry plays an important role in optimization. The usual approach to cope with symmetry in discrete optimization is to try to eliminate it by introducing artificial symmetry-breaking conditions into the problem, and/or by using an ad-hoc search strategy. In this paper we argue that symmetry is instead a beneficial feature that we should preserve and exploit as much as possible, breaking it only as a last resort. To this end, we outline a new approach, that we call orbital shrinking, where additional integer variables expressing variable sums within each symmetry orbit are introduces and used to "encapsulate" model symmetry. This leads to a discrete relaxation of the original problem, whose solution yields a bound on its optimal value. Encouraging preliminary computational experiments on the tightness and solution speed of this relaxation are presented.
Keywords: Mathematical programming, discrete optimization, algebra, symmetry, relaxation, MILP, convex MINLP.

1 Introduction

Nature loves symmetry. Artists love symmetry. People love symmetry. Mathematicians and computer scientists also love symmetry, with the only exception of people working in discrete optimization who always want to break it. Why? The answer is that symmetry is of great help in simplifying optimization in a convex setting, but in the discrete case it can trick search algorithms because symmetric solutions are visited again and again. The usual approach to cope with this redundancy source is to destroy symmetry by introducing artificial conditions into the problem, or by using a clever branching strategy such as isomorphism pruning [6, 7] or orbital branching [10]. We will outline a different approach, that we call *orbital shrinking*, where additional integer variables expressing variable sums within each orbit are introduces and used to "encapsulate" model symmetry. This leads to a discrete relaxation of the original problem, whose solution yields a bound on its optimal value. The underlying idea here is that we see symmetry as a positive feature of our model, so we want to preserve it as much as possible, breaking it only as a last resort. Preliminary computational experiments are presented.

2 Symmetry

We next review some main results about symmetry groups; we refer the reader to, e.g., [1] (pp. 189-190) and [6, 7, 4] for more details. For any positive integer n we use notation $[n] := \{1, \cdots, n\}$. For any optimization problem (Z) we let $v(Z)$ and $F(Z)$ denote the optimal objective function value and the feasible solution set of (Z), respectively. To ease presentation, all problems considered in this paper are assumed to be feasible and bounded.

Our order of business is to study the role of symmetry when addressing a generic optimization problem of the form

$$(P) \quad v(P) := \min\{f(x) : x \in F(P)\} \tag{1}$$

where $F(P) \subseteq \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$. To this end, let $G = \{Q^1, \cdots, Q^K\} \subset \mathbb{R}^{n\times n}$ be a finite group, i.e., a finite set of nonsingular $n \times n$ matrices closed under matrix product and inverse, and assume that, for all $x \in \mathbb{R}^n$ and for all $k \in [K]$:

(C1) the function f is *G-invariant* (or symmetric w.r.t. G), i.e., $f(Q^k x) = f(x)$;
(C2) $x \in F(P) \Rightarrow Q^k x \in F(P)$.

Let the *fixed subspace* of G be defined as $\mathcal{F} := \{x : \forall k \in [K]\ Q^k x = x\}$. Given a point x, we define

$$\overline{x} := \frac{1}{K} \sum_{k=1}^{K} Q^k x \tag{2}$$

as the new point obtained by "averaging x along G". By elementary group theory, left-multiplying G by any of its elements Q^k leaves G unchanged, hence one has $\overline{x} \in \mathcal{F}$ because

$$\forall k \in [K] \quad Q^k \overline{x} = \frac{1}{K} \sum_{h=1}^{K} (Q^k Q^h) x = \frac{1}{K} \sum_{h=1}^{K} Q^h x = \overline{x}. \tag{3}$$

In this paper we always consider the case where all Q^k's in G are permutation matrices, i.e., 0-1 matrices with exactly one entry equal to 1 in each row and in each column. In this case, $y = Q^k x$ if and only if there exists a permutation π^k of $[n]$ such that $y_j = x_i$ and $j = \pi^k(i)$ for all $i \in [n]$. In other words, the elements of group G are just permutations π^k that simply relabel the components of x. This naturally leads to a partition of the index set $[n]$ into $m \geq 1$ disjoint sets $\{\omega_1, \cdots, \omega_m\} = \Omega$, called the *orbits* of G, where i and j $(i < j)$ belong to a same orbit $\omega \in \Omega$ if and only if there exists $k \in [K]$ such that $j = \pi^k(i)$.

From an algorithmic point of view, we can visualize the action of a permutation group G as follows. Consider a digraph $D = (V, A)$ whose n nodes are associated with the indices of the x_j variables. Each π^k in the group then corresponds to the digraph $D^k = (V, A^k)$ whose arcset $A^k := \{(i, j) : i \in V, j = \pi^k(i)\}$ defines a family of arc-disjoint circuits covering all the nodes of D. By definition, $(i, j) \in A^k$ implies that i and j belong to a same orbit. So the orbits are just the connected component of D when taking $A = \bigcup_{k=1}^{K} A^k$. In practice, the group is

defined by a limited set of "basic" elements (called generators) whose product and inverse lead to the entire group, and the orbits are quickly computed by initializing $\omega_i = \{i\}$ for all $i \in [n]$, scanning the generators π^k, in turn, and merging the two orbits containing the endpoints of each arc $(i, \pi^k(i))$ for all $i \in V$ [11].

In practical applications, the permutation group is detected automatically by analyzing the problem formulation, i.e., the specific constraints used to define $F(P)$. In particular, a permutation π is part of the permutation groups only if there exist an associated permutation σ of the indices of the constraints defining $F(P)$, that makes it trivial to verify condition (C2); see e.g. [4].

A key property of permutation groups is that, because of (3), the average point $\overline{x}$ defined in (2) must have $\overline{x}_j$ constant within each orbit, so it can be computed efficiently by just taking x-averages inside the orbits, i.e.,

$$\forall \omega \in \Omega,\ \forall j \in \omega\ \overline{x}_j = \frac{1}{|\omega|} \sum_{i \in \omega} x_i. \tag{4}$$

3 Optimization under symmetry

We next address the role of symmetry in optimization.

3.1 Convex optimization

Assume first that (P) is a convex problem, i.e., f is a convex function and $F(P)$ is a convex set. Then $x \in F(P)$ implies $\overline{x} \in F(P)$ and $f(\overline{x}) = f(\sum_{k=1}^{K} Q^k x/K) \leq \sum_{k=1}^{K} f(Q^k x)/K = f(x)$, so one needs only consider average points $\overline{x}$ when looking for an optimal solution to (P). Because of (4), for permutation groups this means that the only unknowns for (P) are the average x-values inside each orbit or, equivalently, the sums $z_\omega = \sum_{j \in \omega} x_j$ for each $\omega \in \Omega$. It is known that (P) can therefore be reformulated as follows (see Cor. 3.2): (i) introduce a new variable z_ω for each orbit ω, (ii) replace variables x_j for all $j \in \omega$ with their optimal expression $z_\omega/|\omega|$. Optimizing the resulting projected problem on the space of the z variables yields the same optimal objective function value as for the original problem. As a result, symmetry is a very useful property in a convex optimization setting, in that it allows one to *simplify* the optimization problem—provided of course that the symmetry group (or a subgroup thereof) can be found effectively, as is often the case [4].

3.2 Discrete optimization

We now address a discrete optimization setting where the objective function f is still convex but the feasible set is defined as

$$F(P) = \{x \in X : \forall j \in J\ x_j \in \mathbb{Z}\},$$

where $X \subset \mathbb{R}^n$ is a convex set and $J \subseteq [n]$ identifies the variables with integrality requirement. As customary, we assume X be defined as

$$X = \{x \in \mathbb{R}^n : \forall i \in [r]\ f_i(x) \leq 0\},$$

where $f_i : \mathbb{R}^n \to \mathbb{R}$, $i \in [r]$, are convex functions. This framework is quite general in that it covers *Mixed-Integer Linear Programming* (MILP) as well many relevant cases of *Mixed-Integer Nonlinear Programming* (MINLP).

A natural way to exploit symmetry in the above setting is based on the observation that the well known Branch-and-Bound (BB) tree search can be seen as a way to subdivide (P) into a number of convex subproblems, which provide relaxations of (P) that are then used to compute bounds and henceforth prune the search tree. At each tree node one needs to solve a convex subproblem corresponding to X subject to the branching conditions—that we assume be expressed by additional convex constraints—hence symmetry can exploited to simplify this task. Note however that branching conditions alter the symmetry group of the convex subproblem computed at the root node of the search tree, so its recomputation (or heuristic update, as e.g. in [10]) at each node becomes necessary. In addition, the use of symmetry inside each node does not prevent the tree search to possibly enumerate symmetric solutions again and again, so ad-hoc branching strategies are still of fundamental importance.

3.3 Orbital shrinking relaxation

We next propose a different approach, intended to eliminate problem symmetry by "encapsulating" it in a new (relaxed) formulation. More specifically, instead of trying to break symmetries as is standard in the current literature [12, 6, 7, 10, 9, 4], our approach is to solve a discrete relaxation of the original problem, defined on a shrunken space with just one variable for each orbit.

Let G be the group for (P) (found e.g. as in [4]) and let $\Omega = \{\omega_1, \ldots, \omega_m\}$ be its set of orbits (by construction integer and continuous variables will be in different orbits). WLOG, assume the first $\tilde{m}$ orbits involve integer variables only, and let $\Theta = \{\omega_1, \ldots, \omega_{\tilde{m}}\}$, while the remaining orbits (if any) involve continuous variables only (if any). In addition, for each $x \in \mathbb{R}^n$, let $z(x) \in \mathbb{R}^m$ be defined as

$$\forall \omega \in \Omega \quad z_\omega(x) := \sum_{j \in \omega} x_j.$$

For any $g : \mathbb{R}^n \to \mathbb{R}$ let $\overline{g} : \mathbb{R}^m \to \mathbb{R}$ be obtained from $g(x)$ by replacing each x_j by $z_\omega/|\omega|$ ($j \in \omega$, $\omega \in \Omega$). Note that $\overline{g}(z(x)) = g(x)$ holds whenever x_j is constant within each orbit, hence because of (4) one has the identity

$$\overline{g}(z(\overline{x})) = g(\overline{x}). \tag{5}$$

Our *Orbital Shrinking Relaxation* (OSR) is constructed from (P) in two steps:

1. Let (P_{REL}) be the relaxed problem obtained from (P) by replacing the integrality conditions $x_j \in \mathbb{Z}$ $(j \in J)$ by their surrogate version

$$\forall \omega \in \Theta \;\; (\sum_{j \in \omega} x_j) \in \mathbb{Z}\,; \tag{6}$$

2. Reformulate (P_{REL}) as

$$(P_{\mathsf{OSR}}) \qquad v(P_{\mathsf{OSR}}) := \min\{\overline{f}(z) : \forall i \in [r]\ \overline{f}_i(z) \leq 0, \forall \omega \in \Theta\ z_\omega \in \mathbb{Z}\} \tag{7}$$

3.1 Theorem
$v(P_{\mathsf{OSR}}) = v(P_{\mathsf{REL}}) \leq v(P)$.

Proof. Inequality $v(P_{\mathsf{REL}}) \leq v(P)$ is obvious, so we only need to prove $v(P_{\mathsf{OSR}}) = v(P_{\mathsf{REL}})$.

Let x be any optimal solution to (P_{REL}). We claim that $\overline{x}$ is an equivalent optimal solution to (P_{REL}). Indeed, because of (C2) one has $Q^k x \in X$ for all $k \in [K]$, hence $\overline{x} = \left(\frac{1}{K}\sum_{k\in[K]} Q^k x\right) \in X$ because of the convexity of X. In addition, because of (4), $\sum_{j\in\omega} \overline{x}_j = \sum_{j\in\omega} x_j$ for all $\omega \in \Theta$, hence $\overline{x}$ also satisfies (6) and is therefore feasible for (P_{REL}). The optimality of $\overline{x}$ for (P_{REL}) then follows immediately from the convexity of f, which implies $f(\overline{x}) = f\left(\frac{1}{K}\sum_{k\in[K]} Q^k x\right) \leq \frac{1}{K}\sum_{k\in[K]} f(Q^k x) = f(x)$, i.e., $f(\overline{x}) = v(P_{\mathsf{REL}})$.

Now take the point $z(\overline{x})$. Because of (5) and since $\overline{x} \in X$, for all $i \in [r]$ we have $\overline{f}_i(z(\overline{x})) = f_i(\overline{x}) \leq 0$. In addition, $z_\omega(\overline{x}) \in \mathbb{Z}$ for all $\omega \in \Theta$ because $\overline{x} \in F(P_{\mathsf{REL}})$ satisfies (6), hence $z(\overline{x}) \in F(P_{\mathsf{OSR}})$ and then $v(P_{\mathsf{OSR}}) \leq \overline{f}(z(\overline{x})) = f(\overline{x}) = v(P_{\mathsf{REL}})$, thus proving $v(P_{\mathsf{OSR}}) \leq v(P_{\mathsf{REL}})$.

To show $v(P_{\mathsf{OSR}}) \geq v(P_{\mathsf{REL}})$, let z be any optimal solution to (P_{OSR}) and consider the point $x \in \mathbb{R}^n$ with $x_j := z_\omega/|\omega|$ for all $j \in \omega$ and $\omega \in \Omega$. By construction, $\overline{x} = x$ and $z(\overline{x}) = z$. Because of (5), for all $i \in [r]$ one then has $f_i(x) = f_i(\overline{x}) = \overline{f}_i(z(\overline{x})) = \overline{f}_i(z) \leq 0$, i.e., $x \in X$. In addition, for all $\omega \in \Theta$, $\sum_{j\in\omega} x_j = z_\omega \in \mathbb{Z}$, i.e., x also satisfies (6) and therefore $x \in F(P_{\mathsf{REL}})$. Finally, again because of (5), $v(P_{\mathsf{REL}}) \leq f(x) = f(\overline{x}) = \overline{f}(z(\overline{x})) = \overline{f}(z) = v(P_{\mathsf{OSR}})$, i.e., $v(P_{\mathsf{REL}}) \leq v(P_{\mathsf{OSR}})$ as required.

3.2 Corollary
For convex optimization (case $J = \emptyset$), (P_{OSR}) is a reformulation of (P), in the sense that $v(P_{\mathsf{OSR}}) = v(P)$.

Proof. Just observe that (P_{REL}) coincides with (P) when $J = \emptyset$.

4 Experiments on finding the best subgroup

Theorem 3.1 remains obviously valid if a subgroup G' of G is used (instead of G) in the construction of (P_{OSR}). Different choices of G' lead to different relaxations and hence to different bounds $v(P_{\mathsf{OSR}})$. If G' is the trivial group induced by

the identity permutation, no shrinking at all is performed and the relaxation coincides with the original problem. As G' grows in size, it generates longer orbits and the relaxation becomes more compact and easier to solve (also because more symmetry is encapsulated into the relaxation), but the lower bound quality decreases.

Ideally, we wish the relaxation to be (a) as tight as possible and (b) as efficient as possible with respect to the CPU time taken to solve it. In this section we discuss and computationally evaluate a few ideas for generating subgroups G' which should intuitively yield "good" relaxations in a MILP context. All experiments were conducted on a 1.4GHz Intel Core 2 Duo 64bit with 3GB of RAM. The MILP solver of choice is IBM ILOG Cplex 12.2.

4.1 Automatic generation of the whole symmetry group

The formulation group is detected automatically using the techniques discussed in [4]: the MILP is transformed into a Directed Acyclic Graph (DAG) encoding the incidence of variables in objective and constraints, and a graph automorphism software (`nauty` [8]) is then called on the DAG. The orbital-shrinking relaxation is constructed automatically using a mixture of `bash` scripting, GAP [3], AMPL [2], and ROSE [5].

4.2 The instance set

We considered the following 39 symmetric MILP instances:

```
ca36243 ca57245 ca77247 clique9 cod105 cod105r cod83 cod83r cod93 cod93r
cov1053 cov1054 cov1075 cov1076 cov1174 cov954 flosn52 flosn60 flosn84
jgt18 jgt30 mered 04_35 oa25332 oa26332 oa36243 oa57245 oa77247 of5_14_7
of7_18_9 ofsub9 pa36243 pa57245 pa77247 sts135 sts27 sts45 sts63 sts81
```

all taken from F. Margot's website.

4.3 Generator ranking

Using orbital shrinking with the whole symmetry group G has the merit of yielding the most compact relaxation. On our test set, however, this approach yields a relaxation bound which is not better than the LP bound 31 times out of 39, and for the remaining 8 times it is not better than the root-node Cplex's bound (i.e., LP plus root node cuts)—although this will not necessarily be the case for other symmetric instances (e.g., for instances with small symmetry groups).

We observe that the final OSR only depends on the orbits of G' rather than on G' itself, and the smaller G' the more (and/or shorter) orbits it yields. We therefore consider the idea of testing subgroups with orbits of varying size, from small to large. Since testing all subgroups of G is out of the question, we look at its generator list $\Pi = (\pi_0, \ldots, \pi_k)$ (including the identity permutation). For

any permutation π we let $\mathrm{fix}(\pi)$ be the subset of $[n]$ fixed by π, i.e., containing those i such that $\pi(i) = i$. We then reorder our list Π so that

$$|\,\mathrm{fix}(\pi_0)| \geq \cdots \geq |\,\mathrm{fix}(\pi_k)|$$

and for all $\ell \leq k$ we define G_ℓ as the subgroup of G induced by the sublist $(\pi_0, \cdots, \pi_\ell)$. This leads to a subgroup chain

$$G_0, G_1, \cdots, G_k = G$$

with increasing number of generators and hence larger and larger orbits (G_0 being the trivial group induced by the identity permutation). In our view, the first generators in the list are the most attractive ones in terms of bound quality—having a large $\mathrm{fix}(\pi)$ implies that the generated subgroup is likely to remain valid for (P) even when several variables are fixed by branching.

For each instance in our test set, we generated the relaxations corresponding to each G_ℓ and recorded bound values and CPU times, plotting the results against ℓ. We set a maximum user CPU time of 1800s, as we deemed a relaxation useless if it takes too long to solve. The typical behavior of the relaxation in terms of bound value and CPU time was observed to be mostly monotonically decreasing in function of the number ℓ of involved generators. Figure 1 shows an example of these results on the `sts81` instance.

4.4 Choosing a good set of generators

Our generator ranking provides a "dial" to trade bound quality versus CPU time. We now consider the question of how to set this dial automatically, i.e., how to choose a value of $\ell \in [k]$ leading to a good subgroup G_ℓ.

Out of the 39 instances in our test set, 16 yields the same bound independently of ℓ, and were hence discarded from this test. The remaining 23 instances:

```
ca36243 clique9 cod105 cod105r cod83 cod83r cod93 cod93r cov1075 cov1076
cov954 mered O4_35 oa36243 oa77247 of5_14_7 of7_18_9 pa36243 sts135 sts27
sts45 sts63 sts81
```

yields a nonzero decrease in bound value as ℓ increases, so they are of interest for our test.

Having generated and solved relaxations for all $\ell \leq k$, we hand-picked good values of ℓ for each instance, based on these prioritized criteria:

1. bound provided by G_ℓ strictly tighter than LP bound;
2. minimize user CPU time, with strong penalty for choices of ℓ leading to excess of 10 seconds;
3. on lack of other priorities, choose ℓ leading to bounds around midrange in $[\mathrm{bnd}(G_k), \mathrm{bnd}(G_1)]$, where $\mathrm{bnd}(G')$ denotes the bound value obtained by solving the orbital-shrinking relaxation based on the subgroup G'.

sts81		
ℓ/k	obj	CPU
1/14	45	3.60
2/14	45	1.51
3/14	45	1.18
4/14	45	1.13
5/14	33	0.01
6/14	33	0.01
7/14	33	0.02
8/14	33	0.00
9/14	29	0.02
10/14	29	0.00
11/14	29	0.01
12/14	28	0.01
13/14	28	0.00
14/14	27	0.00

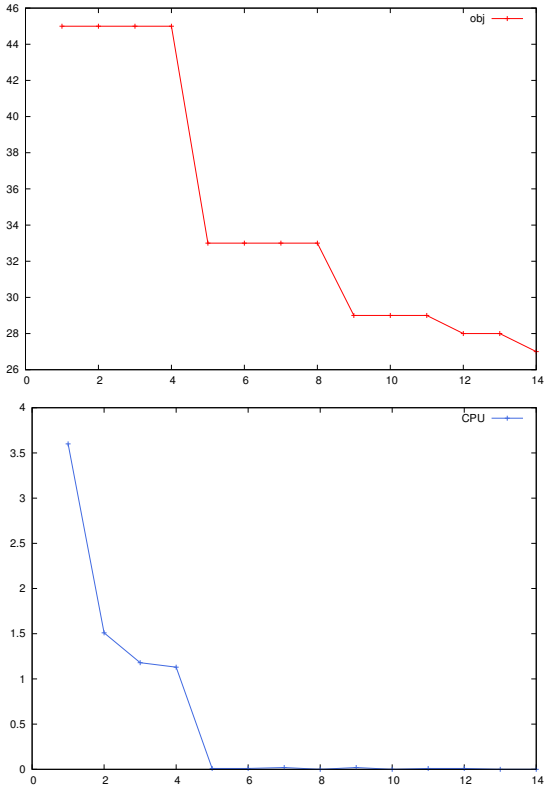

Fig. 1. Bound values and CPU times against the number ℓ of generators for instance `sts81`.

Instance	G_ℓ	LP	CPU	$\frac{\mathrm{inc}(G_\ell)}{\mathrm{inc}(G)}$
ca36243	49	48	0.07	0.50
clique9	∞	36	0.06	0.87
cod105	-16	-18	4.91	0.99
cod105r	-13	-15	0.25	0.99
cod83	-26	-28	0.12	0.98
cod83r	-22	-25	4.44	0.88
cod93	-48	-51	3.07	0.98
cod93r	-46	-47	2.74	0.97
cov1075	19	18	3.03	0.86
cov1076	44	43	185.83	0.73
cov954	28	26	0.45	0.79
mered	∞	140	0.12	0.92
04_35	∞	70	0.07	0.75
oa36243	∞	48	0.75	0.50
oa77247	∞	112	0.00	0.98
of5_14_7	∞	35	0.13	0.62
of7_18_9	∞	63	0.04	0.91
pa36243	-44	-48	1.26	0.50
sts135	60	45	0.05	0.88
sts27	12	9	0.01	0.88
sts45	24	15	0.39	0.66
sts63	27	21	0.00	1.00
sts81	33	27	0.00	0.88

Table 1. Hand-picked choice of the subgroup G_ℓ.

This choice led to the first three columns of Table 1 (the fourth will be explained later). Next we looked for a feature of the solution data over all $\ell \leq k$ and over all instances, whose average value corresponds to values of ℓ that are close to the hand-picked ones in Table 1. Again, intuition led our choice for this feature. Our reasoning is as follows. We observe that, given any orbit ω, our OSR replaces $\sum_{j\in\omega} x_j$ with a single variable z_ω. Suppose now that a constraint $\sum_{j\in\omega} x_j \leq b_i$ happens to exist in the MILP formulation (P): this is simply reformulated to a bound constraint $z_\omega \leq b_i$, thus replacing a $|\omega|$-ary original relation on the decision variables x with a unary relation on the decision variables z. Intuitively, this will over-simplify the problem and will likely yield a poor relaxation. Instead, we would like to deal with orbits that are somehow "orthogonal" to the problem constraints.

To this aim, consider the i-th (out of, say, r) MILP constraint, namely $\sum_{j\in[n]} a_{ij}x_j \leq b_i$, and define the *incidence* of an orbit ω with respect to the support of this constraint as $|\omega \cap \{j \in [n] : a_{ij} \neq 0\}|$. Intuitively, the lower the incidence of an orbit, the farther we are from the situation where problem constraints become over-simplified range constraints in the relaxation. Lower incidence orbits should yield tighter relaxations, albeit perhaps harder to solve. Given a subgroup G' with orbits $\Omega' = \{\omega'_1, \cdots, \omega'_{m'}\}$, we then extend the inci-

dence notion to G'

$$\mathrm{inc}(G', i) := \left| \bigcup_{\omega' \in \Omega'} \omega' \cap \{j \in [n] : a_{ij} \neq 0\} \right|, \tag{8}$$

and finally to the whole MILP formulation

$$\mathrm{inc}(G') = \sum_{i \in [r]} \mathrm{inc}(G', i). \tag{9}$$

The rightmost column of Table 1 reports the relative incidence of G_ℓ, computed as $\mathrm{inc}(G_\ell)/\mathrm{inc}(G)$, for those ℓ that were hand-chosen to be "best" according to the prioritized criteria listed above. Its average is 0.82 with standard deviation 0.17. This value allows us to generate a relaxation which is hopefully "good", by automatically selecting the value of ℓ such that $\mathrm{inc}(G_\ell)/\mathrm{inc}(G)$ is as close to 0.82 as possible.

5 Computational experiments

The quality of the OSR we obtain with the method of Section 4.4 is reported in Table 2 whose columns include: the instance name, the automatically determined value of ℓ and the total number k of generators, the best-known optimal objective function value for the instance (starred values correspond to guaranteed optima), the bound given by G_1 which provides the tightest non-trivial OSR bound, the bound given by G_ℓ and the associated CPU time, the CPU time "cpx_t" spent by CPLEX 12.2 (default settings) on the original formulation to get the same bound as OSR (only reported when the OSR bound is strictly better than the LP bound), and the LP bound. Entry *limit* marks an exceeded time limit of 1800 sec.s, while boldface highlights the best results for each instance.

The results are quite encouraging: our bound is very often stronger than the LP bound, whilst often taking only a fraction of a second to solve. The effect of orbital shrinking can be noticed by looking at the "cpx_t" column, where it is evident that normal branching takes significantly longer to reach the bound given by our relaxation.

6 Conclusions

We discussed a new methodology for deriving a tight relaxation of a given discrete optimization problem, based on orbit shrinking. Results on a testbed of MILP instances are quite encouraging. Our work opens up several directions: how to find a good generator set, whether it is possible to find extensions to general MINLPs, how will the relaxation perform when used within a Branch-and-Bound algorithm and how to dynamically change it along the search tree, and which insights for heuristics can be derived from the integer solution of the relaxed problem.

Instance	ℓ/k	best	G_1	G_ℓ	CPU	cpx_t	LP
ca36243	3/6	49*	49	48	0.02		48
clique9	5/15	∞*	∞	∞	**0.06**	0.17	36
cod105	3/11	-12*	*limit*	**-14.09**†	*limit*		-18
cod105r	3/10	-11*	-11	**-11**	**24.12**	28.36	-15
cod83	3/9	-20*	-21	**-24**	16.78	**9.54**	-28
cod83r	3/7	-19*	-21	**-22**	**4.44**	7.85	-25
cod93	3/10	-40		**-46.11**†	*limit*		-51
cod93r	3/8	-38	-39	**-44**	**271.74**	446.48	-47
cov1075	3/9	20*	20	**19**	**3.03**	79.79	18
cov1076	3/9	45	44	43	2.78		43
cov954	3/8	30*	28	26	0.11		26
mered	21/31	∞*	∞	∞	**0.15**	3.37	140
O4_35	3/9	∞*	∞	70	0.00		70
oa36243	3/6	∞*	∞	48	0.01		48
oa77247	3/7	∞*	∞	∞	**0.10**	265.92	112
of5_14_7	7/9	∞*	∞	35	0.00		35
of7_18_9	7/16	∞*	∞	∞	**0.09**	0.15	63
pa36243	3/6	-44*	-44	-48	0.01		-48
sts135	3/8	106	75	**60**	**0.11**	109.81	45
sts27	4/8	18*	14	**12**	0.01	1.05	9
sts45	2/5	30*	24	15	0.00		15
sts63	4/9	45*	36	**27**	**0.02**	1.99	21
sts81	5/14	61	45	**33**	**0.01**	3.92	27

Table 2. OSR performance (in the G_ℓ/CPU columns). Entries marked * denote guaranteed optimal values; those marked † denote the best lower bound at the time limit. In sts27, CPLEX closes the gap at the root node. Values for cpx_t are only present when the OSR bound is integer and better than the LP bound.

Acknowledgements

The first author was supported by the *Progetto di Ateneo* on "Computational Integer Programming" of the University of Padova. The second author was partially supported by grants Digiteo Chair 2009-14D "RMNCCO" and Digiteo 2009-55D "ARM". We thank Mauro Diligenti who asked the question that originated the present work—Why do you discrete optimization guys hate symmetry and want to destroy it?

References

1. S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, 2004.
2. R. Fourer and D. Gay. *The AMPL Book*. Duxbury Press, Pacific Grove, 2002.
3. The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.4.10*, 2007.
4. L. Liberti. Reformulations in mathematical programming: Automatic symmetry detection and exploitation. *Mathematical Programming A*, DOI 10.1007/s10107-010-0351-0.

5. L. Liberti, S. Cafieri, and D. Savourey. Reformulation optimization software engine. In K. Fukuda, J. van der Hoeven, M. Joswig, and N. Takayama, editors, *Mathematical Software*, volume 6327 of *LNCS*, pages 303–314, New York, 2010. Springer.
6. F. Margot. Pruning by isomorphism in branch-and-cut. *Mathematical Programming*, 94:71–90, 2002.
7. F. Margot. Exploiting orbits in symmetric ILP. *Mathematical Programming B*, 98:3–21, 2003.
8. B. McKay. *nauty User's Guide (Version 2.4)*. Computer Science Dept. , Australian National University, 2007.
9. 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.
10. J. Ostrowski, J. Linderoth, F. Rossi, and S. Smriglio. Orbital branching. *Mathematical Programming*, 126:147–178, 2011.
11. A. Seress. *Permutation Group Algorithms*. Cambridge University Press, Cambridge, 2003.
12. H. Sherali and C. Smith. Improving discrete model representations via symmetry considerations. *Management Science*, 47(10):1396–1407, 2001.