Symmetry Breaking Inequalities from the Schreier-Sims table

Domenico Salvagnin

Department of Information Engineering (DEI), University of Padova
domenico.salvagnin@unipd.it

Abstract. We propose a way to derive symmetry breaking inequalities for a mixed-integer programming (MIP) model from the Schreier-Sims table of its formulation group. We then show how to consider only the action of the formulation group onto a subset of the variables. Computational results show that this can lead to considerable speedups on some classes of models.

1 Motivation

An optimization problem is symmetric if its variables can be permuted without changing the structure of the problem. Even for relatively small cases, symmetric optimization problems can be difficult to solve to proven optimality by traditional enumerative algorithms, as many subproblems in the enumeration tree are isomorphic, forcing a wasteful duplication of effort. Symmetry has long been recognized as a challenge for exact methods in constraint and integer programming, and many different methods have been proposed in the literature, see for example the surveys in [6,14].

A common source of symmetry in a model is the underlying presence of identical objects. Let's consider a MIP model in which a subset of variables, say $x_1, \dots, x_k$, corresponds to k identical objects, e.g., the k colors in a classical graph coloring model [8]. A natural way to get rid of the symmetry implied by those objects, which is well known and widely used by modelers, is to add to the formulation the following chain of static symmetry breaking inequalities:

$$x_1 \geq x_2 \geq \ldots \geq x_k \tag{1}$$

The validity of (4) can be easily proved as follows. If the k objects are identical, then G acts on $x_1, \dots, x_k$ as the full symmetric group S_k, and thus all those variables are in the same orbit. As such, we can always permute the variables such that the first is not less than the others, which means adding the inequalities $x_1 \geq x_i$ for all $i > 1$. This effectively singles out variable x_1. Still, what is left is the full symmetric group S_{k-1} on the $k-1$ variables $x_2, \dots, x_k$. We can apply the very same reasoning and add the inequalities $x_2 \geq x_i$ for all $i > 2$. Repeating the same argument till the bitter end we get that we added all inequalities of the form $x_i \geq x_j$ for all $i > j$, which is equivalent to the chain above after removing the redundant ones.

The effectiveness of such a simple symmetry handling technique can vary significantly. In particular, it depends on the distribution of the feasible solutions of the model: if many symmetric solutions lie on the hyperplanes $x_i = x_{i+1}$, then the method is usually rather ineffective, and not competitive with other more elaborate symmetry handling techniques, like isomorphism pruning [12,13] and orbital branching [15]. On the other hand, if there are no solutions with $x_i = x_{i+1}$, e.g. because the variables are linked by some all-different constraint, then (4) is equivalent to a chain of strict inequalities, and all symmetry in the model is effectively broken [17].

Example 1. Let's consider a simple 2D packing model[1], in which we have k squares of size 10×10 and a container of size $10(k-1)+1 \times 11$. As no two squares can fit vertically in the container, clearly only $k-1$ squares can be put into the container. A typical MIP model for this problem has a pair of continuous variables (x_i, y_i) for each square, encoding the coordinates of, say, the lower-left corner of the squares, plus $\Theta(k^2)$ binary variables to encode the non-overlapping constraints among squares. Now, because of the shape of the container, we can assume that w.l.o.g. $y_i = 0$ for all squares in any feasible solution, while clearly $x_i \neq x_j$ for any two squares in any feasible solution. So adding the symmetry breaking inequalities $x_1 \geq x_2 \geq \ldots \geq x_k$ effectively removes all the symmetries from the formulation—and indeed works very well in practice with most solvers—while adding the chain $y_1 \geq y_2 \geq \ldots \geq y_k$ is totally ineffective, and actually harms the solution process as it still destroys the symmetry in the formulation, preventing other symmetry handling methods to kick in. □

The example above confirms that inequalities (4) might or might not be an effective way to deal with symmetry in MIP. Still, there are cases in which they outperform all other symmetry handling techniques, so the question is: can we derive those static symmetry breaking inequalities from the model *automatically*? The answer is positive, as it turns out that inequalities (4) are a special case of a wider class of symmetry breaking inequalities that can be derived from the so-called *Schreier-Sims table* [19], a basic tool in computational group theory.

The outline of the paper is as follows. In Section 2 we review the basic concepts of group theory needed for our discussion, and define the Schreier-Sims representation of a group. In Section 3 we show how to use the Schreier-Sims table to derive symmetry breaking inequalities, and present some extensions/improvements over the basic method in Section 4. In Section 5 we outline an algorithm to actually compute the table. Computational results are given in Section 6, with conclusions and future directions of research drawn in Section 7.

2 The Schreier-Sims table

We follow the description of the Schreier-Sims [19,18] representation given in [12]. Let G be a permutation group on the ground set $N = \{1, \ldots, n\}$. A permutation

[1] This is a much simplified version of the `pigeon` models [2] in MIPLIB 2010 [9].

$g \in G$ is represented by an n-vector, with $g[i]$ being the image of i under g. Consider the following chain of (nested) subgroups of G:

$$\begin{aligned} G_0 &= G \\ G_1 &= \{g \in G_0 |\ g[1] = 1\} \\ G_2 &= \{g \in G_1 |\ g[2] = 2\} \\ &\dots \\ G_n &= \{g \in G_{n-1} |\ g[n] = n\} \end{aligned} \tag{2}$$

In other words, G_i is the stabilizer of i in G_{i-1}. Note that no such subgroup is empty, as the identity permutation is always contained in all G_i. For each $i = 1, \dots, n$, let $orb(i, G_{i-1}) = \{j_1, \dots, j_p\}$ be the orbit of i under G_{i-1}, i.e., the set of elements onto which i can be mapped according to G_{i-1}. Note that the orbit is never empty, as it always contains at least i. By definition, for each element j_k of the orbit there exists a permutation in G_{i-1} mapping i to j_k, and let h_{i,j_k} be any such permutation. Let $U_i = \{h_{i,j_1}, \dots, h_{i,j_p}\}$ be the set of these permutations, called *coset* representatives. Again, U_i is never empty. We can arrange the permutations in the sets U_i in an $n \times n$ table T, with:

$$T_{i,j} = \begin{cases} h_{i,j} & \text{if } j \in orb(i, G_{i-1}) \\ \emptyset & \text{otherwise} \end{cases} \tag{3}$$

The table T is called the Schreier-Sims representation of G. The most basic property of the table is that the set of permutations stored in the table form a set of strong generators for the group G, i.e., any permutation of $g \in G$ can be expressed as a product of at most n permutations in the set. It is also worth noting that the Schreier-Sims table not only provides a set of strong generators for G but also for all the nested subgroups G_i: indeed, a set of strong generators for G_i is obtained by taking all permutations in the table with row index $k \geq i$.

Example 2. Let's consider the simple permutation group G of the symmetries of the 2×2 square, with cells numbered top to bottom and left to right. G contains 8 permutations. The corresponding Schreier-Sims table is depicted below, where each permutation is written in cycle notation, and i is the identity. Note that G is not equal to S_4 because, e.g., no permutation exists maps cell 2 to cell 4 without affecting cell 1 as well.

	1	2	3	4
1	i	(1 2)(3 4)	(1 3)(2 4)	(1 4)(3 2)
2		i	(2 3)	
3			i	
4				i

□

Note that the Schreier-Sims table is always upper triangular, and it is fully dense iff $G = S_n$, so constructing the table is sufficient to detect whether a group

G is the full symmetric group S_n. Given an arbitrary set of generators for G, the Schreier-Sims table can be constructed in polynomial time (more details are given in Section 5).

3 Deriving symmetry breaking inequalities

Consider a MIP model P with n variables and let $G = \langle g_1, \ldots, g_r \rangle$ be its formulation symmetry group, i.e., the group of permutations of the variables that lead to an equivalent formulation, see [12,11] for a formal definition. Let T be the Schreier-Sims representation of G. Then it follows that:

Theorem 1. *The set of symmetry breaking inequalities $x_i \geq x_j$ for all $T_{ij} \neq \emptyset$ is valid for P.*

Proof. The proof is a simple generalization of the argument used to prove the validity of chain (4). Let us consider the orbit O_1 of variable x_1 according to G. By definition we can always permute the variables such that x_1 takes a value which is no less than the values taken by the other variables in the orbit, so the set of inequalities corresponding to the first row of T, namely $x_1 \geq x_j \quad \forall x_j \in O_1$ is valid for P. Now, let's add those inequalities to the model. The formulation group of the resulting model contains G_1, i.e., the stabilizer of x_1 in G, so we can proceed to the second row of the table, which gives exactly the orbit of x_2 in G_1. Thus we can reiterate the argument and conclude that the set of inequalities corresponding to the second row of T is valid for P. By induction we can continue until the very last row of T, which proves the theorem. □

It is worth noting that the addition of symmetry breaking inequalities can in principle result in new symmetries in the formulation, as shown by the following example:

Example 3. Consider the LP:

$$\min\{x_1 + x_2 + x_3 + x_4 : x_3 - x_4 \geq 0\} \tag{4}$$

The corresponding formulation group has only one symmetry, namely $(x_1\ x_2)$. However, adding $x_1 - x_2 \geq 0$ we get the additional symmetry $(x_1\ x_3)(x_2\ x_4)$, while the stabilizer of x_1 according to G would contain the identity permutation only. □

Note that Theorem 1 only proves that we can derive a valid set of symmetry breaking inequalities from the Schreier-Sims table T, but not that the inequalities above are in general sufficient to break all the symmetries in the model. Indeed, the latter statement would be false in general. What we can state however is that (i) adding those inequalities breaks all symmetries in the *original* formulation, and (ii) all solution symmetries of the original formulation are broken if the variables of the model are linked by an all-different constraint, a result already proved in [17]. The fact that in any case formulation symmetries are broken is

a double-edged sword: if solution symmetries are also broken then everything is fine, otherwise the addition of those inequalities is not only ineffective but also prevents other methods from being applied, as they would find no (or very little, see Example 3) symmetries to exploit, as shown in Example 1.

4 Improvements

Suppose we are interested in how the formulation group G acts on a subset T of variables of the model. For example, we might want to check whether G acts as $S_{|T|}$ on T, despite G possibly not being S_n. This can easily be achieved by a small extension of the Schreier-Sims construction, in which we do not consider the variables in order from x_1 to x_n when constructing the table, but in a different order, say β, such that the variables in T are considered first. Such order β is called the *base* of the table, and the construction can easily be extended to deal with an arbitrary base. Once the table is constructed, then we can conclude that G acts as $S_{|T|}$ on T iff the upper left $|T| \times |T|$ submatrix of T is (upper triangular) fully dense.

Constructing the complete Schreier-Sims table of order n when we are actually interested only in its upper left corner of size $|T| \times |T|$ can potentially be a big waste of computing resources. For example, in Example 1, the model has size $\Theta(k^2)$, while the continuous variables that encode the placing of each object are only $O(k)$. In general the full computation is needed if the set T has no structure. However, if we assume that T is an orbit according to the original group G, then we have a much better alternative: intuitively, we can project the generators of G and work with a new group G_T whose ground set is just T. Then we can construct the Schreier-Sims table of G_T which is exactly of size $|T| \times |T|$. Let us formalize this argument.

Any generator g of G (as any permutation for that matter), can be written in cycle notation. Because of our choice of T, by construction all cycles in g either move only variables in T or only variables in $N \setminus T$, as there is no permutation in G moving an element from T into $N \setminus T$, otherwise T would not be an orbit.

Define the operator $\varphi : S_n \leftrightarrow S_{|T|}$ as the operator that drops from a permutation written in cycle notation all the cycles not in T. For example, if $g = (13)(25)(789)$ and $T = \{1, 2, 3, 4, 5\}$, then $\varphi(g) = (13)(25)$.

Let $t_1, \ldots, t_r$ be the permutations obtained by applying φ to the generators of G and let $G_T = \langle t_1, \ldots, t_r \rangle$. It is not difficult to prove that φ is a homomorphism from G to G_T: let $a = \gamma_1 \cdots \gamma_k \delta_1 \ldots \delta_p$ and $b = \sigma_1 \cdots \sigma_l \omega_1 \ldots \omega_q$, where we used γ and σ to indicate the cycles moving variables in T and δ and ω to indicate the cycles moving variables in $N \setminus T$ (and we can always write a and b into this form as the cycles can be written down in any order). Then:

$$\varphi(ab) = \varphi(\gamma_1 \cdots \gamma_k \delta_1 \ldots \delta_p \sigma_1 \cdots \sigma_l \omega_1 \ldots \omega_q) \quad (5)$$
$$= \varphi(\gamma_1 \cdots \gamma_k \sigma_1 \cdots \sigma_l \delta_1 \ldots \delta_p \omega_1 \ldots \omega_q) \quad (6)$$
$$= \gamma_1 \cdots \gamma_k \sigma_1 \cdots \sigma_l \quad (7)$$
$$= \varphi(a)\varphi(b) \quad (8)$$

where the first rearrangement of the cycles is allowed because they are disjoint.

In addition, as a homomorphism from G to G_T, φ is surjective. Indeed, let π be a permutation in G_T. By definition it can be obtained by the generators of G_T and their inverses. But for each generator t_i of G_T we know a permutation h of G such that $\varphi(h) = t_i$, namely $h = g_i$ and the same is true for the inverses, because $\varphi(g_i^-) = t_i^-$. Thus we can always construct a permutation $g \in G$ such that $\varphi(g) = \pi$. For example, let $\pi = t_1 t_2^- t_5$. Then $g = g_1 g_2^- g_5$.

Thus, by working directly with the group G_T we are not introducing (nor removing) any symmetry *among the variables in* T that was not already in G, hence we can use G_T to study how G acts on T. The results still holds if T is not just a single orbit but a union of orbits of G.

5 Constructing the Schreier-Sims table

A recursive algorithm to compute the Schreier-Sims table is described in [10], and used in [12,13]. However, in our computational experience, we found a different iterative algorithm to perform better in practice. The iterative algorithm constructs the Schreier-Sims table one row at the time, and works as follows. At any given iteration i, the algorithm assumes that a set of generators for G_{i-1} is readily available (this condition is trivially satisfied for the first row, where we can just use the generators of G). Then, it computes the orbit O_i and the set of coset representatives U_i for element i. This is a basic algorithm in computational group theory, called Schreier vector construction [3]. Note that this is enough to fill row i of the table. Then we need to compute the generators for G_i, in order to be ready for the next iteration. This is achieved in two steps:

1. Compute a set of generators for G_i applying the Schreier's lemma. In details, given $G_{i-1} = \langle g_1, \ldots, g_r \rangle$ and the coset representatives $U_i = \{r_1, \ldots, r_k\}$, we can obtain a set of generators for G_i as $\langle r_s^{-1} g r \rangle$, with $g \in G_i$, $r \in U_i$, and r_s chosen such that $(r_s^{-1} g r)[i] = i$.
2. Reduce the set of generators for G_i applying the Sims' filter. This leaves at most $O(n^2)$ generators for G_i. This is needed in order to obtain a polynomial algorithm for the Schreier-Sims table construction. Note that other filters are known, such as for example Jerrum's filter [4]. However, those are more complicated to implement.

The overall complexity of the construction is $O(n^6)$. As noted already in [12], an algorithm with a worst-case complexity of $O(n^6)$ might seem impractical even

for reasonable values of n. However, we confirm that those bounds are very pessimistic and that the actual runtime of the algorithm was always negligible w.r.t. to the overall solution process. Still, care must be taken in the implementation, allowing the construction to be interrupted in case it becomes too time consuming.

6 Computational results

We implemented the separation of the static symmetry breaking inequalities described in Theorem 1 during the development cycle between IBM ILOG CPLEX 12.7.0 and 12.7.1 [7]. In particular, at the end of presolve, we use the generators of the formulation group, freshly computed with `AUTOM` [16], to construct the Schreier-Sims table on the orbit of continuous variables with largest domain. While the approach can in principle be applied to binary and general integer variables as well, we decided to apply the method very conservatively. The choice of continuous variables with large domains is intuitively justified by the fact that it is "less likely" to have solutions lying on the $x_i = x_{i+1}$ in this case. If the table is sufficiently dense, we add the symmetry breaking inequalities and erase the generators (they are no longer valid), otherwise we forget about the table and continue.

We tested the method on the CPLEX internal testbed, which consists of approximately 3270 models, coming from a mix of publicly available and commercial sources. Tests were executed on a cluster of identical machines, each equipped with two Intel Xeon E5-2667v4 CPUs (for a total of 16 cores) running at 3.2GHz, and 64GB of RAM. Each run was given a time limit of 10.000 seconds. To limit the effect of performance variability [5,9], we compared the two methods, namely CPLEX defaults with (`symbreak`) and without (`cpx`) the addition of the symmetry breaking inequalities derived from the Schreier-Sims table, with 5 different random seeds. Aggregated results over the 5 seeds are given in Table 1.

The structure of the table is as follows. Instances are divided in different subsets, based on the *difficulty of the models*. To avoid any bias in the analysis, the level of difficulty is defined by taking into account both methods under comparison. The subclasses "$[n,\ 10\text{k}\}$" ($n = 1, 10, 100, 1\text{k}$), contain the subset of models for which at least one of the methods took at least n seconds to solve and that were solved to optimality within the time limit by at least one of the methods. Finally, the subclasses "$[n,\ 10\text{k})$" ($n = 1, 10, 100, 1\text{k}$) contain all models in "$[n,\ 10\text{k}\}$" but considering only the models that were solved to optimality by both methods. The first column of the table identifies the class of models. Then the first group of 5 columns, under the heading "all models", reports results on all instances in the class, while the second group of columns, under the heading "affected", repeats the same information for the subset of models in each class where the two methods took a different solution path. Within each group, column "# models" reports the number of models in the class, columns "#tl" the number of time limits for each method, and columns "time" and

"nodes" report the shifted geometric means [1] of the ratios of solution times and number of branch-and-bound nodes, respectively. Ratios $t < 1$ indicate a speedup factor of $1/t$.

	all models					affected		
	`cpx`		`symbreak`			`cpx`	`symbreak`	
class	# models	# tl	# tl	time	nodes	# models	time	nodes
$[0, 10K\}$	16185	127	111	0.99	0.98	317	0.52	0.32
$[1, 10K\}$	9475	82	66	0.98	0.96	303	0.50	0.30
$[100, 10K\}$	2645	79	63	0.93	0.87	159	0.27	0.10
$[0, 1)$	6665	0	0	1.00	1.00	14	1.36	1.94
$[1, 10)$	3905	0	0	1.00	1.00	48	0.97	1.00
$[10, 100)$	2920	0	0	1.00	1.00	96	1.05	1.09
$[100, 1K)$	1765	0	0	0.99	0.97	86	0.73	0.51
$[1K, 10K)$	680	0	0	0.94	0.89	40	0.35	0.14

Table 1: Aggregated results.

According to Table 1, the symmetry breaking inequalities affect only around 2% of the models, which is not unexpected given the conservative criteria that trigger their generation. Still, they are so effective that they produce a non negligible speedup also on the whole testbed, with speedups ranging from 1% to 7% (for the subset of hard models in the "[100, 10k)" bracket). Also the number of time limits is significantly reduced. Aggregated results seed by seed (not reported) also confirm that the improvement is consistent across seeds.

7 Conclusions

In this paper we investigated computationally the effectiveness of generating static symmetry breaking inequalities from the Schreier-Sims table of the formulation symmetry group. Computational results show that the approach can be extremely effective on some models. The technique is implemented and activated by default in the release 12.7.1 of the commercial solver IBM ILOG CPLEX. Future direction of research include extending the classes of models on which the method is tried, e.g., on pure binary models.

8 Acknowledgements

The author would like to thank Jean-François Puget for an inspiring discussion about the Schreier-Sims table, and three anonymous reviewers for their careful reading and constructive comments.

References

1. Tobias Achterberg. *Constraint Integer Programming.* PhD thesis, Technische Universität Berlin, 2007.
2. Sam D. Allen, Edmund K. Burke, and Jakub Marecek. A space-indexed formulation of packing boxes into a larger box. *Operations Research Letters*, 40:20–24, 2012.
3. Gregory Butler and John J. Cannon. Computing in permutation and matrix groups I: Normal closure, commutator subgroups, series. *Mathematics of Computation*, 39:663–670, 1982.
4. Peter J. Cameron. *Permutation Groups.* London Mathematical Society St. Cambridge University Press, 1999.
5. Emilie Danna. Performance variability in mixed integer programming, 2008. MIP 2008 workshop in New Work, http://coral.ie.lehigh.edu/~jeff/mip-2008/talks/danna.pdf.
6. Ian P. Gent, Karen E. Petrie, and Jean-François Puget. Symmetry in constraint programming. In Francesca Rossi, Peter van Beek, and Toby Walsh, editors, *Handbook of Constraint Programming*, pages 329–376. Elsevier, 2006.
7. IBM. ILOG CPLEX 12.7.1 User's Manual, 2017.
8. Volker Kaibel and Marc Pfetsch. Packing and partitioning orbitopes. *Mathematical Programming*, 114:1–36, 2008.
9. Thorsten Koch, Tobias Achterberg, Erling Andersen, Oliver Bastert, Timo Berthold, Robert E. Bixby, Emilie Danna, Gerald Gamrath, Ambros M. Gleixner, Stefan Heinz, Andrea Lodi, Hans Mittelmann, Ted Ralphs, Domenico Salvagnin, Daniel E. Steffy, and Kati Wolter. MIPLIB 2010 - Mixed Integer Programming Library version 5. *Mathematical Programming Computation*, 3:103–163, 2011.
10. Donald L. Kreher and Douglas R. Stinson. *Combinatorial Algorithms: Generation, Enumeration, and Search.* CRC Press, 1999.
11. Leo Liberti. Reformulations in mathematical programming: Automatic symmetry detection and exploitation. *Mathematical Programming*, 131:273–304, 2012.
12. François Margot. Pruning by isomorphism in branch-and-cut. *Mathematical Programming*, 94(1):71–90, 2002.
13. François Margot. Exploiting orbits in symmetric ILP. *Mathematical Programming*, 98(1):3–21, 2003.
14. François Margot. Symmetry in integer linear programming. In *50 Years of Integer Programming.* Springer, 2009.
15. Jim Ostrowski, Jeff Linderoth, Fabrizio Rossi, and Stefano Smriglio. Orbital branching. *Mathematical Programming*, 126(1):147–178, 2011.
16. Jean-François Puget. Automatic detection of variable and value symmetries. In Peter van Beek, editor, *Constraint Programming*, volume 3709 of *LNCS*, pages 475–489, New York, 2005. Springer.
17. Jean-François Puget. Breaking symmetries in all different problems. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI*, pages 272–277, 2005.
18. Ákos Seress. *Permutation Group Algorithms.* Cambridge University Press, 2003.
19. Charles C. Sims. Computational methods in the study of permutation groups. In *Computational problems in abstract algebra*, pages 169–183, Oxford, 1970. (Oxford, 1967), Pergamon Press.