Symmetry Handling in MIP solvers

Domenico Salvagnin
DEI, University of Padova

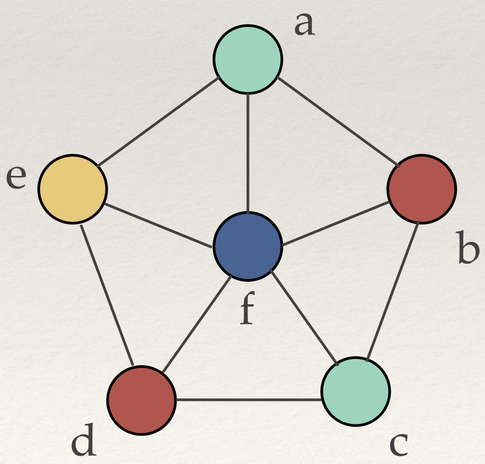
Agenda

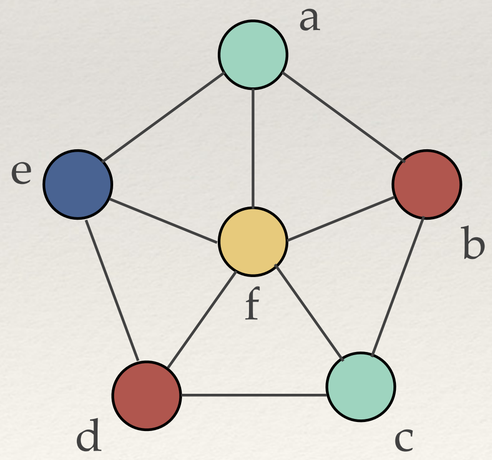
- What is a Symmetry?
- Symmetry Detection
- Symmetry Representation (Group Theory)
- Discrete Optimization
  - Orbital branching/fixing/probing
- Continuous Optimization
  - Fractional Symmetries
  - LP folding/folding eliminations

Introduction

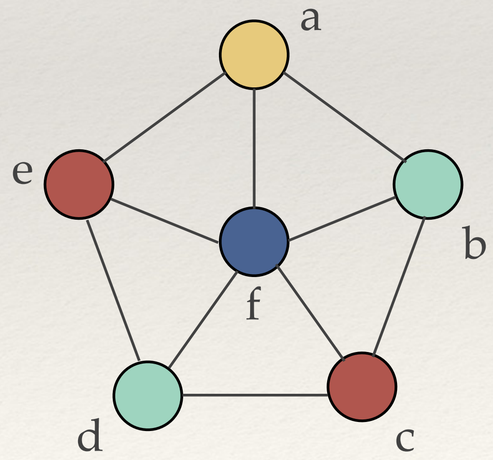
(Permutation) Symmetries in Optimization

- What is a symmetry in an optimization problem?
- Intuitively, a symmetry is a permutation π that we can apply to feasible solutions and that always returns (other) feasible solutions (of the same cost).





swap two colors



rotate graph

(Permutation) Symmetries in Optimization

- Is this good or bad news?
- In some sense, symmetries point to some redundant information in the problem:
  - If **not** exploited, this usually means useless work by the algorithm :-(
  - If exploited, they can lead to improved performance :-)
- How to exploit them depends on the class of problems and algorithms
- But first…how do we detect symmetries in a problem?

Symmetry Detection

Formulation Symmetries

- Given an optimization problem P with feasible set F(P) and objective function f, the definition of a permutation symmetry π given so far is along the following lines:

$$x \in F(P) \Leftrightarrow \pi(x) \in F(P) \qquad f(x) = f(\pi(x))$$

- Utterly useless: we don't know the set of feasible solutions of a problem :-(
- In practice we resort to *formulation symmetries*, i.e., symmetries that keep the formulation of the problem invariant.
  - Note that this is a **weaker** definition, easily fooled by innocuous changes to a formulation, like scaling or adding a redundant constraint!

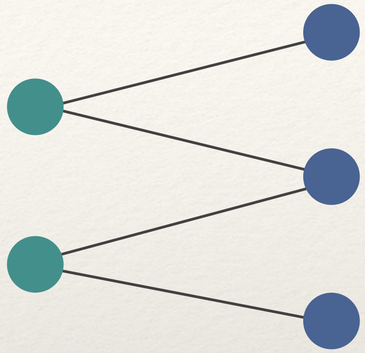
Formulation Symmetries

- A variable symmetry π is valid is there exists a corresponding constraint symmetry σ that keeps the *formulation* of the problem the same
- Concept is the same for more general constraints (arbitrary mathematical programs in algebraic form)
- Formulation is encoded as a ***vertex-coloured graph***
  - Again, very generic, not restricted to MILP
  - We can apply standard methods/packages to compute graph automorphisms
    - *Complexity unknown but usually fast in practice*

Symmetry Graph

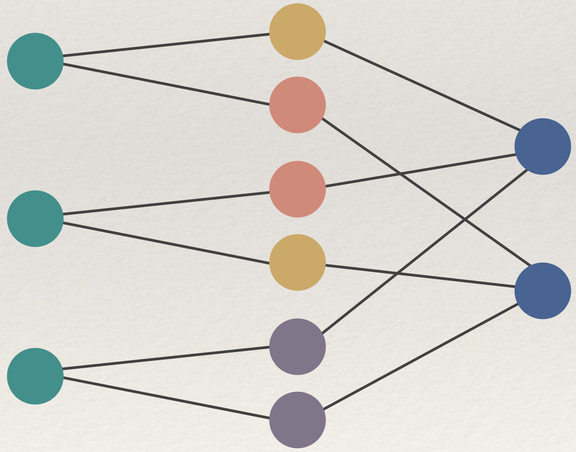
binary

$$\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$



general

$$\begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 3 & 3 \end{bmatrix}$$



Can be simplified: connect directly row-col pairs corresponding to most frequent coefficient!

Symmetry Representation

Symmetric Group

- How are the symmetries of a problem represented?
- A naïve way is to store the list of all symmetries…but this would work only on toy problems:
  - *Consider a graph coloring problem with 20 colors: that alone would give* $20! > 10^{18}$ *symmetries!*
- The key observation is to note that the symmetries of a problem form a (finite) **group**:
  - Any finite group can be represented by a (polynomially sized) list of **generators** (like in a vector space).

Group Concepts

- The group theory point of view is key in computing with (and exploiting) symmetries.
- Given a group G acting on ground set N *(in our case, the formulation symmetries acting on the variables of the problem)*, we can define the two fundamental concepts:

Orbit of an object x

$$G(x) = \{g(x),\ g \in G\}$$

Set of objects that x can be mapped to by permutations in G

Stabilizer of a set S

$$stab(G, S) = \{g \in G |\ g(S) = S\}$$

Set of permutations that map the objects in S to themselves (a subgroup of G)

Group Concepts

- Why are those two concepts important?
  - Orbits tell us which objects (variables) are interchangeable given a symmetry group
  - Stabilizers tell us which symmetries are still valid after we have taken some branching decisions
  - Note that we can (and do) compute orbits w.r.t. a stabilizer…
- Orbits partition the set of variables into equivalence classes, so we often talk of *orbit partition*.

Group Computations

- Orbits can be computed in linear time given generators
  - Standard union-find data structure
- For stabilizers it depends…
  - For point-wise stabilizers (each element of S is mapped to itself), it can be done in polynomial time (but the polynomial is not great…).
  - For set-wise stabilizers (the set S is mapped to itself as a whole), the complexity is the same as for graph automorphism :-(
  - In general the latter concept is more powerful but more expensive to compute (exactly).

Discrete Optimization

Orbital Branching

- Consider an orbit O = {x_{i1},…,x_{ik}} of binary variables at the root of the branch-and-bound tree.
- Standard branching would pick one (say the first), and apply the disjunction:

$$x_{i_1} = 1 \vee x_{i_1} = 0$$

- However, the right branch still contains all the symmetric solutions x_{i2}=1, x_{i3}=1,…
- This is even more explicit by considering (still valid) k+1 branch:

$$x_{i_1} = 1 \vee \ldots \vee x_{i_k} = 1 \vee x_{i_1} = \ldots = x_{i_k} = 0$$

Ostrowski et al. *"Orbital Branching"* Mathematical Programming 2011

Orbital Branching

- The gist of orbital branching is to drop all the equivalent branches (except one) and thus consider the disjunction:

$$x_{i_1} = 1 \vee x_{i_1} = \ldots = x_{i_k} = 0$$

- The same can be done when branching on a general integer variable: the standard disjunction:

$$x_{i_1} \geq v + 1 \vee x_{i_1} \leq v$$

becomes

$$x_{i_1} \geq v + 1 \vee x_{i_1} \leq v, \ldots, x_{i_k} \leq v$$

Ostrowski et al. *"Orbital Branching"* Mathematical Programming 2011

Orbital Branching

- What about the rest of the tree?
- The logic is exactly the same, but we need to use an appropriate stabilizer of the global symmetry group for computing the orbits!
- The most general approach consists in keeping track of which bound changes are *symmetry breaking* (branching only so far) and which are not, and compute the stabilizer w.r.t. those only
  - The stabilizer will thus only map variables with tightened bounds (by branching) only to other variables with the same bounds.
    - *Simple case: variables fixed to one (resp. zero) by branching can only map to other variables fixed to one (resp. zero) by branching.*

Ostrowski et al. *"Orbital Branching"* Mathematical Programming 2011

Orbital Fixing

- This tries to apply the orbital logic to tightenings that come from other arguments (primal reductions, reduced cost fixing, etc…)
- Consider the orbits at the current node (again, those are computed w.r.t. a stabilizer that only considers symmetry breaking bound changes).
- Within each orbit, compute the maximum lower bound L and the minimum upper bound U, and set them for all variables in the orbit.
- *Example: if reduced cost fixing deduces that a variable in an orbit can be fixed zero, we would fix to zero all the variables in the same orbit.*
- This is why we need to distinguish symmetry bound changes: otherwise orbital fixing would never tighten anything…

Ostrowski et al. *"Orbital Branching"* Mathematical Programming 2011

Orbital Probing

- For each orbit of integer variables:
  - Fix all the variables in the orbit to their (identical) lower bound v
  - Perform constraint propagation
  - If infeasible we can set one variable in the orbit to v+1
  - **Update orbits**

also a symmetry breaking bound change!

Achterberg *"Probing, Probing, Probing and Probing"* MIP Workshop 2013

Results

		NO SYMMETRY				SYMMETRY				
class	#	#S	time	nodes	PDI	#S	tQ	nQ	pdiQ	W/L
all	4190	3970	166.88	2327	42.46	4010***	0.93***	0.86***	0.96***	43/13
[0,3600}	4028	3970	146.68	1896	39.24	4010***	0.92***	0.86***	0.96***	43/13
[1,3600}	4016	3958	147.95	1925	39.47	3998***	0.92***	0.86***	0.96***	43/13
[10,3600}	3874	3816	161.72	2199	41.85	3856***	0.92***	0.85***	0.96***	43/13
[100,3600}	2293	2235	361.45	5773	71.21	2275***	0.87***	0.78***	0.94***	36/9
[1000,3600}	452	394	1703.72	99885	172.40	434***	0.64***	0.50***	0.78***	13/3
all-opt	3952	3952	138.30	1654	38.14	3952	0.95***	0.89***	0.98***	33/11
[0,0)	0	0	0.00	0	0.00	0	1.00	1.00	1.00	0/0
[0,1)	12	12	0.41	1	0.30	12	1.01	1.00	1.02	0/0
[1,10)	142	142	6.16	42	3.72	142	1.00	0.99	1.01	0/0
[10,100)	1581	1581	46.08	537	17.04	1581	1.00	0.97***	1.01	5/1
[100,1000)	1841	1841	245.19	2863	56.58	1841	0.94***	0.87***	0.99***	12/4
[1000,3600)	376	376	1550.33	53566	177.49	376	0.82***	0.69***	0.89***	5/1

FICO XPRESS 9.7 - 838 instances 5 seeds 16 threads

Continuous Optimization

Continuous Optimization

- All the symmetry handling techniques presented are discrete in nature and are applied during branch-and-bound
- Does it mean that there is nothing to do if the problem is continuous???
- To the contrary!!!
- If the problem is continuous and **convex**, we can do so much more :-)

Symmetry and Convexity

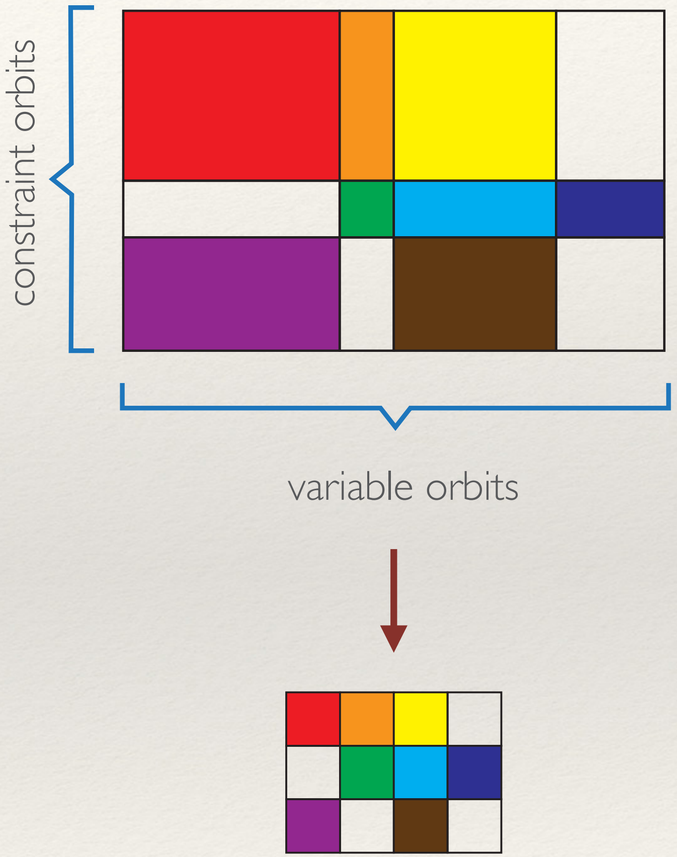
- The key intuition is that if the problem is convex, we can always "average" symmetrical solutions and thus assume that all variables within an orbit have the same value
- Consider a permutation swapping, among other things, the first and the second variable:

- Repeating the argument for all variables in all orbits gets the claim.

Orbital Shrinking

Not actually implemented, but simpler to understand:

- Aggregate variables in the same column orbit
  - *Convex continuous: can always average symmetric optimal solutions*
- Keep only one constraint for each row orbit
  - *All constraints in the same orbit would be identical copies after the variable aggregations above*



It is a no brainer if we already have the symmetry group available, but…

- Computing the symmetry group is potentially expensive
- We can actually do strictly better!!!

K. Gatermann and P. Parrilo *"Symmetry groups, semidefinite programs and sums of squares"* Journal of Pure and Applied Algebra 2004

Fractional Symmetries

$$PAQ = A$$

~~*permutation matrices*~~

doubly-stochastic matrices

- Still valid in the convex continuous case
- Can be computed in polynomial time (and efficiently in practice)
- Give coarser (equitable) partitions than orbital partitions!

Grohe, Kersting, Mladenov, Selman *"Dimension Reduction via Colour Refinement"* CoRR 2013

LP Folding

- Use equitable partition to shrink model
- Independent of symmetry detection for MIP
- Note that it does not give a basic solution for the original problem!
  - Nothing to do if interested in a primal-dual pair;
  - Need crossover otherwise (can be expensive)
- Enabled by default on when solving LPs
- Disabled by default when solving MIPs!

Results

		NO LP FOLDING.			LP FOLDING				
class	#	#S	time	nodes	PDI	#S	tQ	nQ	pdiQ	W/L
all	840	834	67.16	1	0.00	836	0.88***	1.00	1.00	30/4
[0,3600}	836	834	65.76	1	0.00	836	0.88***	1.00	1.00	30/4
[1,3600}	833	831	66.29	1	0.00	833	0.88***	1.00	1.00	30/4
[10,3600}	667	665	105.86	1	0.00	667	0.87***	1.00	1.00	24/4
[100,3600}	292	290	331.96	1	-0.00	292	0.78***	1.00	1.00	17/2
[1000,3600}	40	38	2015.70	1	0.00	40	0.33***	1.00	1.00	2/0
all-opt	834	834	65.06	1	0.00	834	0.89***	1.00	1.00	28/4
[0,0)	0	0	0.00	0	0.00	0	1.00	1.00	1.00	0/0
[0,1)	3	3	0.84	1	0.00	3	1.00	1.00	1.00	0/0
[1,10)	166	166	4.23	1	0.00	166	0.88***	1.00	1.00	6/0
[10,100)	375	375	39.88	1	0.00	375	0.95***	1.00	1.00	6/2
[100,1000)	252	252	247.84	1	0.00	252	0.89***	1.00	1.00	10/2
[1000,3600)	27	27	1523.75	1	0.00	27	0.49	1.00	1.00	1/0

FICO XPRESS 9.7 - 240 instances 3 seeds 16 threads

Folding Eliminations

- What if we have symmetries affecting continuous variables ***only*** in a MIP?
  - What does this even mean?
  - The symmetries must be valid for the *point-wise* stabiliser of the integer variables
  - Then we can still aggregate along those orbits (of continuous variables only)
- This is weaker than necessary:
  - We can again use fractional symmetries to compute a coarser partition (but still having each integer variable in a singleton block).

Results

		NO FOLDING ELIM				FOLDING ELIM				
class	#	#S	time	nodes	PDI	#S	tQ	nQ	pdiQ	W/L
all	4190	4002	156.16	2023	41.08	4010	0.99***	0.99	1.00**	10/3
[0,3600}	4013	4002	135.06	1609	37.58	4010	0.99***	0.99	1.00**	10/3
[1,3600}	3999	3988	136.41	1635	37.84	3996	0.99***	0.99	1.00**	10/3
[10,3600}	3837	3826	150.62	1858	40.46	3834	0.99***	0.99	1.00**	10/3
[100,3600}	2173	2162	347.51	4419	71.72	2170	0.98***	0.98	0.99*	9/2
[1000,3600}	363	352	1665.61	54241	178.73	360	0.92**	0.85**	1.00	3/0
all-opt	3999	3999	133.64	1585	37.56	3999	0.99*	1.00	1.00*	7/3
[0,0)	0	0	0.00	0	0.00	0	1.00	1.00	1.00	0/0
[0,1)	14	14	0.47	6	0.27	14	1.00	1.00	1.01	0/0
[1,10)	162	162	6.32	71	3.51	162	1.00	1.00	1.00	0/0
[10,100)	1664	1664	46.49	595	16.88	1664	1.00	1.00	1.00	1/1
[100,1000)	1810	1810	252.27	2670	59.09	1810	0.99*	1.01	0.99	4/0
[1000,3600)	349	349	1640.31	52495	188.79	349	0.98	0.97	0.99	1/0

FICO XPRESS 9.7 - 838 instances 5 seeds 16 threads

Thanks for your attention!