Reformulations in Mathematical Programming

Leo Liberti

LIX, École Polytechnique, France

Reformulations in Mathematical Programming: Definitions

Mathematical Programming

- Mathematical programs consist of sets of *parameters*, *variables*, *objective functions*, *constraints*
- The **objective functions** are mathematical expressions in terms of parameters and variables, together with an optimization direction
- The **constraints** are relations between mathematical expressions in terms of parameters and variables
- All such entities can also be expressed in terms of **indices**, which must be *quantified* over specified sets

The Language

- Consider an alphabet L including numbers, mathematical operators ($+, -, \times, \div, \uparrow, \sum, \prod, \log, \exp, \forall$), brackets, and symbols denoting parameters and variables
- **Given a sequence of elements of L, is it a well-formed statement of a mathematical program?**
- I.e. , we treat Mathematical Programming (MP) as a language, whose semantic purpose is to describe a set of points in a Euclidean space (the *optima*)
- One possible grammar of the MP language is specified in the appendix of the AMPL book

Main motivation

- Given an optimization problem, many different *MP formulations* can describe its solution set
- The performances of solution algorithms depend on the MP formulation
- **Given an optimization problem and a solution algorithm, what is the MP formulation yielding the best performance?**
- *How do we pass from one formulation to another that keeps some (all) of the mathematical properties of the old formulation?*

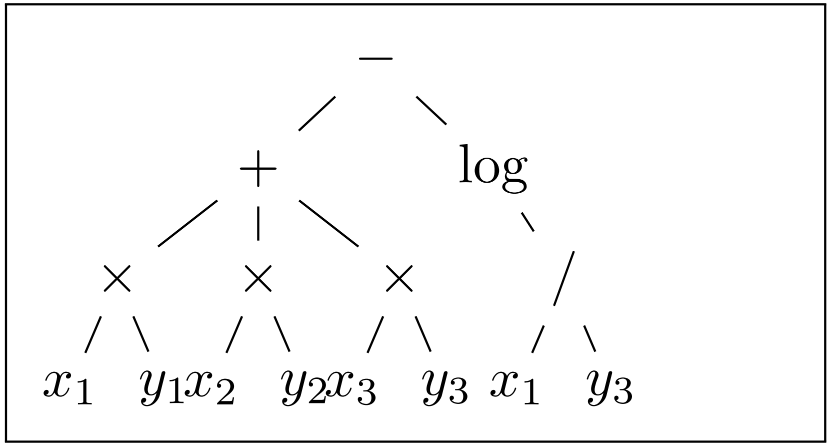
Reformulations: Existing definitions

- "Q is a reformulation of P": what does it mean?
- Definition in Mathematical Programming Glossary:
  Obtaining a new formulation Q of a problem P that is in some sense better, but equivalent to a given formulation. **Trouble:** vague.
- Definition by H. Sherali [private communication]:
  bijection between feasible sets, objective function of Q is a monotonic univariate function of that of P. **Trouble:** feasible sets bijection: condition is too restrictive
- Definition by P. Hansen [Audet et al., JOTA 1997]: P, Q *opt. problems; given an instance p of P and q of Q and an optimal solution y^* of q, Q is a reformulation of P if an optimal solution x^* of p can be computed from y^* within a polynomial amount of time*. **Trouble:** only maintains optimality, requires polynomial-time transformation

Storing MP formulations

- Mathematical expressions as n-ary *expression trees*

$$\sum_{i=1}^{3} x_i y_i - \log(x_1/y_3)$$



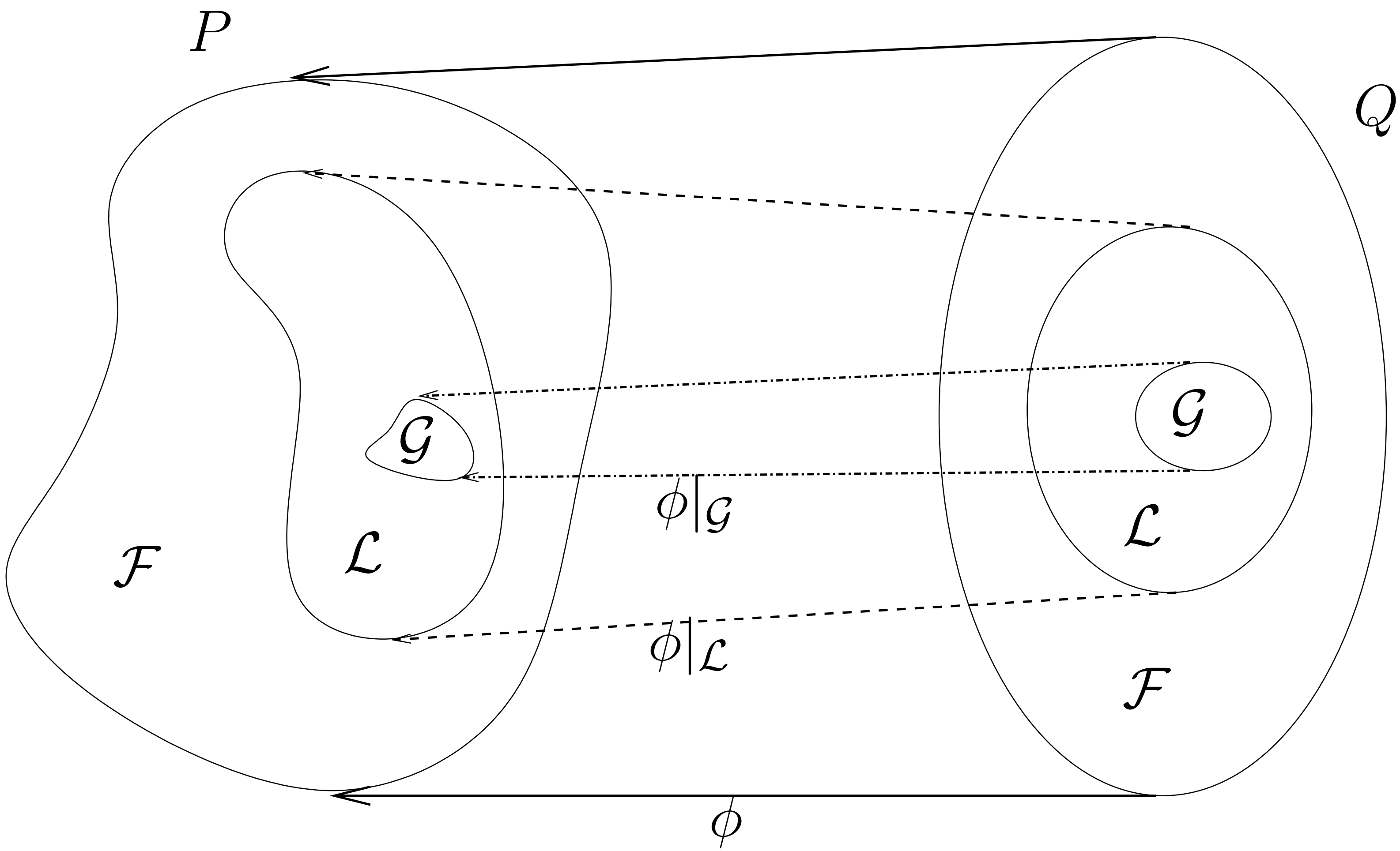
- A *formulation* P is a 7-tuple $(\mathcal{P}, \mathcal{V}, \mathcal{E}, \mathcal{O}, \mathcal{C}, \mathcal{B}, \mathcal{T})$ =(parameters, variables, expression trees, objective functions, constraints, bounds on variables, variable types)
- Objectives are encoded as pairs (d, f) where $d \in \{-1, 1\}$ is the optimization direction and f is the function being optimized
- Constraints are encoded as triplets $c \equiv (e, s, b)$ ($e \in \mathcal{E}$, $s \in \{\leq, \geq, =\}$, $b \in \mathbb{R}$)
- $\mathcal{F}(P) =$ feasible set, $\mathcal{L}(P) =$ local optima, $\mathcal{G}(P) =$ global optima

Auxiliary problems

If problems P, Q are related by a computable function f through the relation $f(P,Q) = 0$, Q is an *auxiliary problem* with respect to P.

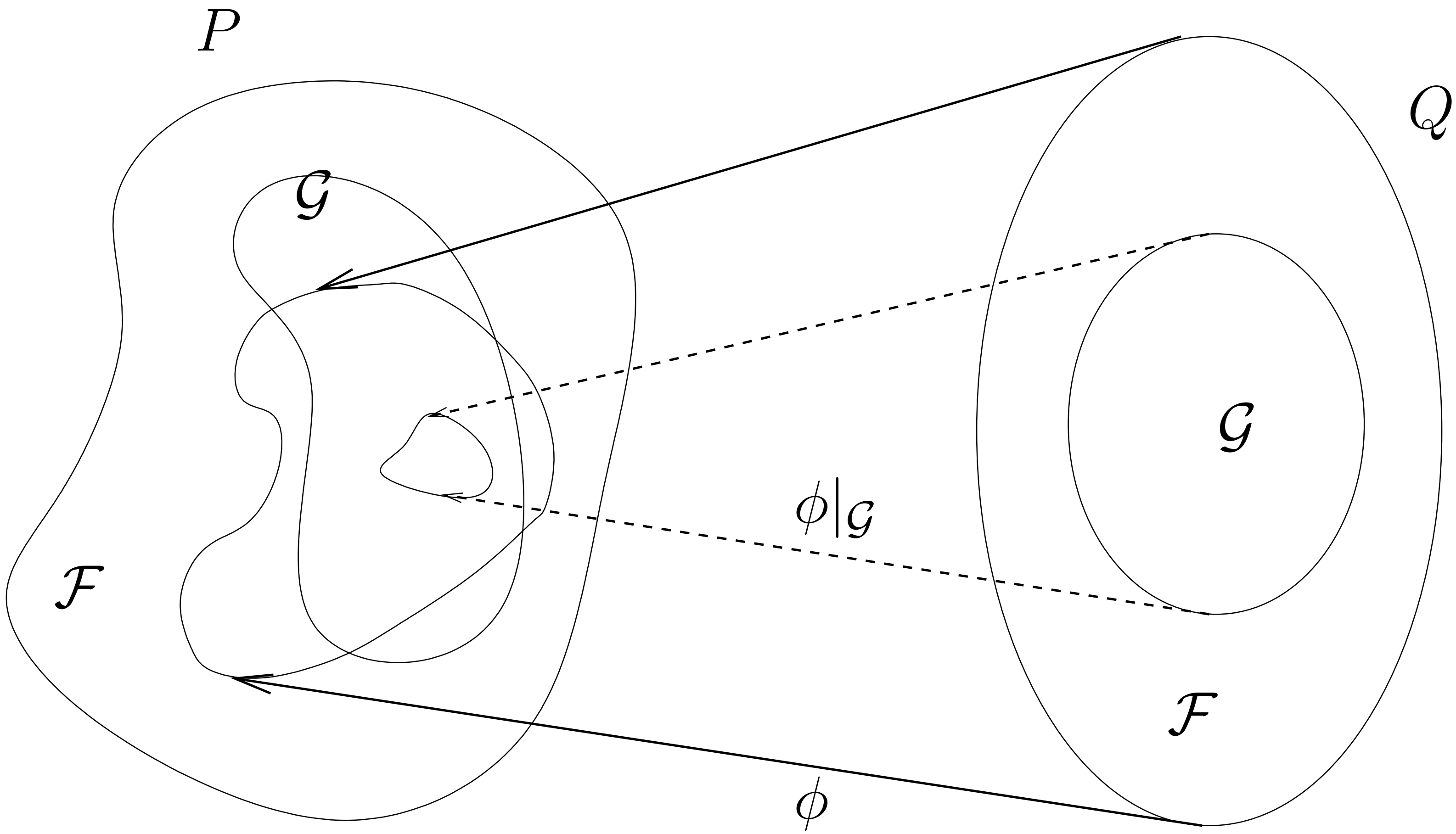
- **Opt-reformulations** (or *exact reformulations*): preserve all optimality properties
- **Narrowings**: preserve some optimality properties
- **Relaxations**: provide bounds to an objective function value towards its optimization direction
- **Approximations**: formulation Q depending on a parameter k such that "$\lim_{k\to\infty} Q(k)$" is an opt-reformulation, narrowing or relaxation

Opt-reformulations



Main idea: if we find an optimum of Q, we can map it back to the same type of optimum of P, and for all optima of P, there is a corresponding optimum in Q.

Narrowings



Main idea: if we find a global optimum of Q, we can map it back to a global optimum of P. There may be optima of P without a corresponding optimum in Q.

Relaxations

- A problem Q is a *relaxation* of P if: (a) $\mathcal{F}(P) \subseteq \mathcal{F}(Q)$ and (b) for all $(f,d) \in \mathcal{O}(P)$, $(\bar{f},\bar{d}) \in \mathcal{O}(Q)$ and $x \in \mathcal{F}(P)$ we have $\bar{d}\bar{f}(x) \geq df(x)$
- **Relaxations guarantee the bound of all objectives over all the feasible region**
- - A problem Q is a *weak relaxation* of P if there are: $(d,f) \in \mathcal{O}(P)$, $(\bar{d},\bar{f}) \in \mathcal{O}(P)$, $x^* \in \mathcal{G}(P)$, $y^* \in \mathcal{G}(Q)$ such that $\bar{d}\bar{f}(y^*) \geq df(x^*)$
  - **Weak relaxations identify order relations between optima of single objective pairs**

Approximations

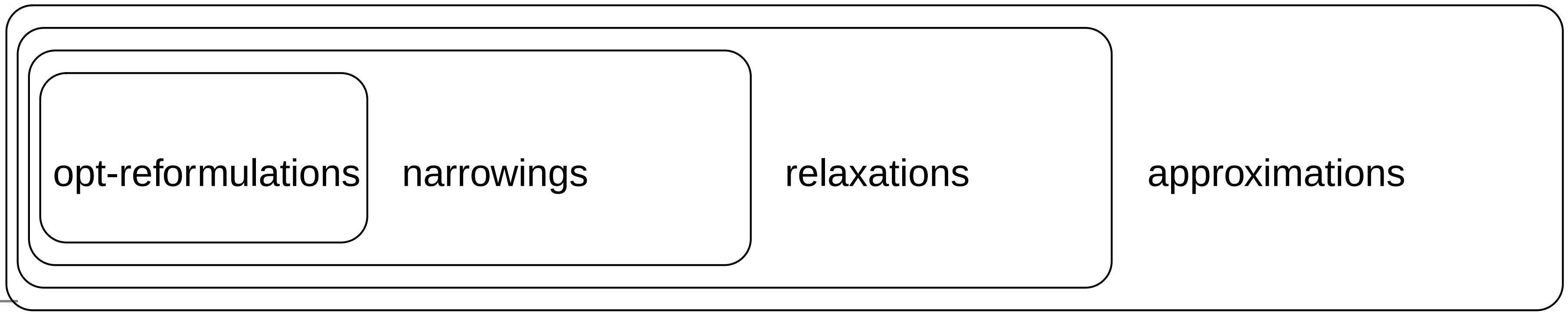
Q is an *approximation* of P if there exist: (a) an auxiliary problem Q^* of P; (b) a sequence $\{Q_k\}$ of problems; (c) an integer $k' > 0$; such that:

1. $Q = Q_{k'}$
2. $\forall f^* \in \mathcal{O}(Q^*)$ there is a sequence of functions $f_k \in \mathcal{O}(Q_k)$ converging uniformly to f^*;
3. $\forall c^* = (e^*, s^*, b^*) \in \mathcal{C}(Q^*)$ there is a sequence of constraints $c_k = (e_k, s_k, b_k) \in \mathcal{C}(Q_k)$ such that e_k converges uniformly to e^*, $s_k = s^*$ for all k, and b_k converges to b^*.

There can be approximations to opt-reformulations, narrowings, relaxations

Composition laws

- Opt-reformulation, narrowing, relaxation, approximation are all transitive relations, **so they can be chained**
- An approximation of *any reformulation chain* is an approximation
- A reformulation chain involving *opt-reformulations, narrowings, relaxations* is a relaxation
- A reformulation chain involving *opt-reformulations and narrowings* is a narrowing
- A reformulation chain involving *opt-reformulations only* is an opt-reformulation



Research programme

- Identify a library of reformulations that can be carried out *automatically* (by a computer) — **under way**
- Implement data structures for holding MP formulations as well as algorithms for changing their structures — **under way**
- Create a language for combining elementary reformulations into complex (possibly conditional) reformulations, according to the composition laws above — **to do**
- Create a heuristic method for finding out the *best* reformulation given an optimization problem and a solution algorithm — **to do**

Research team

- Myself, obviously
- One full-time senior researcher (Pierre Hansen, funded by Digiteo)
- Two full-time postdocs (F. Tarissan, funded by FP6 Morphex EU project; and S. Cafieri, funded by the ANR “ARS” project)
- Some part-time researchers (F. Messine, Toulouse; L. Létocart, Paris 13)
- A number of external collaborators (C. D’Ambrosio, P. Janes, S. Perron, N. Mladenović, F. Plastria, J. Ninin and others)

Reformulations in Mathematical Programming: Symmetry

The setting

- Most common solution algorithm for finding global optima: **Branch-and-Bound** (BB for MILPs, sBB for MINLPs)
- BB (implicit enumeration): provides a certificate of optimality in the linear case, and of ε-approximation in the nonlinear case
- If the problem has symmetries: many BB nodes will contain (symmetric) optimal solutions $\Rightarrow$ *pruning will occur rarely* $\Rightarrow$ BB converges slowly
- Need a reformulation which is guaranteed to *keep at least one global optimum* (but hopefully excludes a lot of symmetric optima): a **narrowing**

Motivating example

- Consider an instance P:

$$\begin{array}{lllllllll}
\min & x_{11} & +x_{12} & +x_{13} & +x_{21} & +x_{22} & +x_{23} & & \\
 & x_{11} & +x_{12} & +x_{13} & & & & \geq & 1 \\
 & & & & x_{21} & +x_{22} & +x_{23} & \geq & 1 \\
 & x_{11} & & & +x_{21} & & & \geq & 1 \\
 & & x_{12} & & & +x_{22} & & \geq & 1 \\
 & & & x_{13} & & & +x_{23} & \geq & 1
\end{array}$$

  of the covering prob. $\boxed{\min \mathbf{1}x : \forall i \sum_j x_{ij} \geq 1 \wedge \forall j \sum_i x_{ij} \geq 1}$

- The set of solutions is $\mathcal{G}(P) =$

$\{(0,1,1,1,0,0),(1,0,0,0,1,1),(0,0,1,1,1,0),(1,1,0,0,0,1),(1,0,1,0,1,0),(0,1,0,1,0,1)\}$

- $G^* = \text{stab}(\mathcal{G}(P), S_n)$ is the *solution group* **(column permutations keeping $\mathcal{G}(P)$ fixed)**

Symmetries

- For the above instance, G^* is $\langle(2,3)(5,6),(1,2)(4,5),(1,4)(2,5)(3,6)\rangle \cong D_{12}$

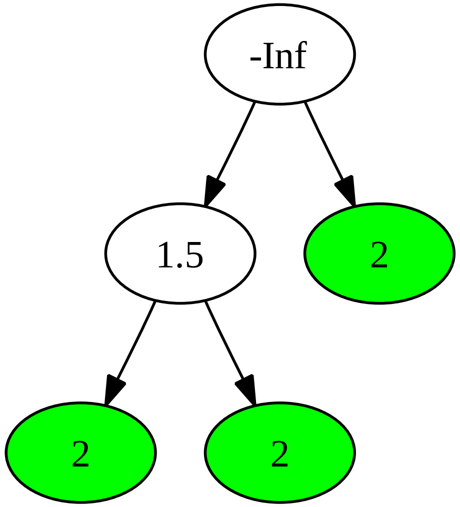
```
gap> S := [[0,1,1,1,0,0],[0,1,0,1,0,1],[0,0,1,1,1,0],
           [1,1,0,0,0,1],[1,0,0,0,1,1],[1,0,1,0,1,0]];
     G:=MatrixAutomorphisms(S); StructureDescription(G);
Group([ (2,3)(5,6), (1,2)(4,5), (1,4)(2,5)(3,6) ]); "D12"
```

- For all $x^* \in \mathcal{G}(P)$, $G^* x^* = \mathcal{G}(P) \Rightarrow \exists$ only 1 orbit $\exists$ only *one* solution in $\mathcal{G}(P)$ (modulo symmetries)

```
gap> Orbit(G,S[1],Permuted);
[[0,1,1,1,0,0],[1,1,0,0,0,1],[1,0,1,0,1,0],
 [1,0,0,0,1,1],[0,0,1,1,1,0],[0,1,0,1,0,1]]
gap> Orbit(G,S[2],Permuted);
[[0,1,0,1,0,1],[1,0,0,0,1,1],[0,0,1,1,1,0],
 [1,0,1,0,1,0],[0,1,1,1,0,0],[1,1,0,0,0,1]]
gap> Orbit(G,S[3],Permuted);
[[0,0,1,1,1,0],[0,1,0,1,0,1],[1,0,0,0,1,1],
 [1,1,0,0,0,1],[1,0,1,0,1,0],[0,1,1,1,0,0]]
...
```

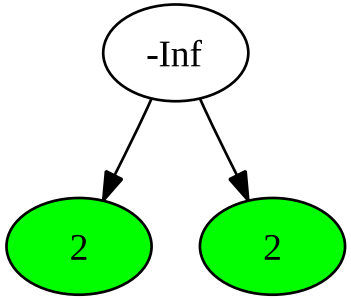
Symmetries

- This is **bad** for Branch-and-Bound techniques: many branches will contain (symmetric) optimal solutions and therefore will not be pruned by bounding $\Rightarrow$ *deep and large BB trees*



$\longleftarrow$ *BB tree for symmetric 6×9 covering inst. with* $G^* = C_2$

BB tree using symmetry breaking constraints $\longrightarrow$



- If we knew G^* in advance, we might add constraints eliminating (some) symmetric solutions out of $\mathcal{G}(P)$
- Can we find G^* (or a subgroup thereof) *a priori*?
- What constraints provide a valid *reformulation* of P excluding symmetric solutions of $\mathcal{G}(P)$?

Symmetries and formulation

- The cost vector $c^{\mathsf{T}} = (1,1,1,1,1,1)$ is fixed by all (column) permutations in S_6
- The vector $b = (1,1,1,1,1)$ is fixed by all (row) permutations in S_5
- Consider P's constraint matrix:

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

- Let $\pi \in S_6$ be a column permutation such that $\exists$ a row permutation $\sigma \in S_5$ with $\sigma(A\pi) = A$
- Then permuting the variables/columns in P according to π does not change the problem formulation

The problem group

- For a MILP with $c = \mathbf{1}_n$ and $b = \mathbf{1}_m$,

$$G_P = \{\pi \in S_n \mid \exists \sigma \in S_m \ (\sigma A \pi = A)\} \quad (1)$$

  is called the *problem group* of P

- In the example above, we get $G_P \cong D_{12} \cong G^*$

```
gap> A := [[1,1,1,0,0,0],[0,0,0,1,1,1],
           [1,0,0,1,0,0],[0,1,0,0,1,0],[0,0,1,0,0,1]];
     G:=MatrixAutomorphisms(A); StructureDescription(G);
Group([ (1,4)(2,5)(3,6), (2,3)(5,6), (1,2)(4,5) ]); "D12"
```

Thm.

For a covering/packing problem P, $G_P \leq G^*$.

- Result can be extended to all MILPs [Margot: 2002, 2003 (Math. Prog.); 2007 (DO)]

Related results in MILP

- **Isomorphism pruning** [*Margot 02,03*], involves addition of linear inequalities of packing type *locally* to selected nodes of the BB tree (as well as var. fixing)
- **Orbitopes** [*Kaibel et al. 07,08*]: "polytopes modulo symmetries" for C_n and S_n groups only
- **Fundamental domains** [*Friedman 07*]: given a (discrete) domain X and a group G acting on X, a fundamental domain is a subset F of X such that $GF = X$ (determination of smallest FDs w.r.t. given ordering vectors c)
- **Orbital branching** [*Ostrowski et al. 07,08*] branching scheme taking advantage of the problem group (yields fewer branching disjunctions)

Related results in CP

- Much more work in CP than in MILP
- **Definitions**: Cohen et al., *Symmetry Definitions for Constraint Satisfaction Problems*, CP 2005. Relations between constraint and solution groups

---

Survey

F. Margot, *Symmetry in Integer Linear Programming*, to appear in “50 Years of Integer Programming”, Springer.

My contributions

1. **MILPs** (COCOA08 paper):
   - A MILP-based method for finding subgroups of the problem group
   - Some static symmetry-breaking constraints (narrowing reformulation)
2. **MINLPs** (new material):
   - Definition of the problem group
   - Reduction to GRAPH ISOMORPHISM
   - Orbit-based static symmetry-breaking constraints (narrowing reformulation)

Symmetries in MINLPs

- **Consider the following MINLP P:**

$$\left.\begin{array}{rrcl} \min & f(x) & & \\ & g(x) & \leq & 0 \\ & x & \in & X. \end{array}\right\} \qquad (2)$$

where X may contain integrality constraints on x

- For a row permutation $\sigma \in S_m$ and a column permutation $\pi \in S_n$, we define $\sigma P \pi$ as follows:

$$\left.\begin{array}{rrcl} \min & f(x\pi) & & \\ & \sigma g(x\pi) & \leq & 0 \\ & x\pi & \in & X. \end{array}\right\} \qquad (3)$$

- Define $G_P = \{\pi \in S_n \mid \exists \sigma \in S_m \; (\sigma P \pi = P)\}$

Representing $g(x\pi)$

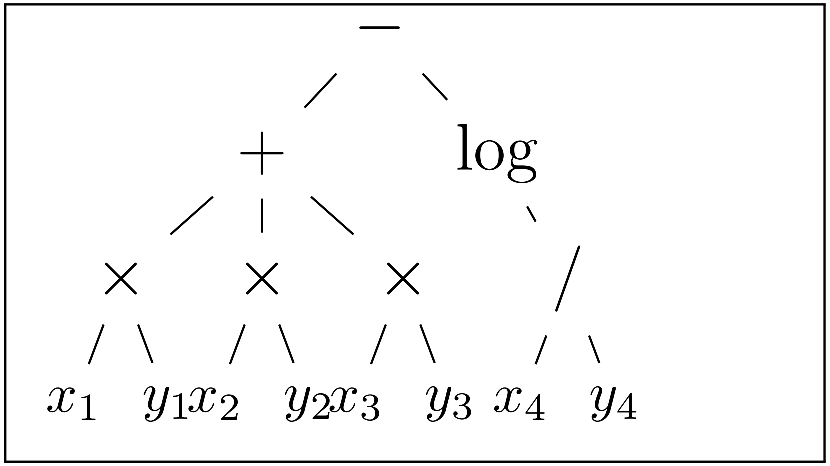
- In the linear case, writing $Ax\pi$ is easy — how do we deal with $g(x\pi)$?

  How do we decide whether $g_i(x) = g_h(x\pi)$ for $i, h \le m$?

- **Answer**: consider the *expression DAG* representation of g

  $$\sum_{i=1}^{3} x_i y_i - \log(x_4/y_4)$$

  List of expressions $\equiv$ expression DAG sharing variable leaf nodes

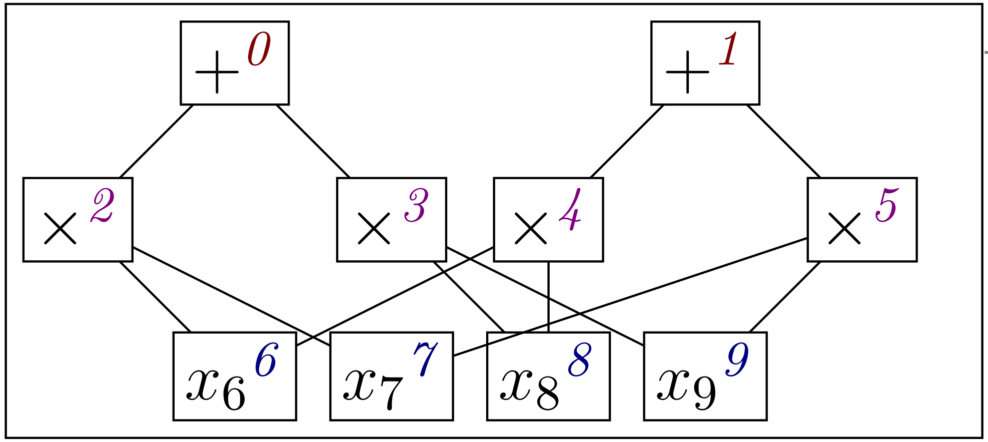


- *Every function $g : \mathbb{R}^n \to \mathbb{R}^m$ is represented by a DAG whose leaf nodes are variables and constants and whose intermediate nodes are mathematical operators*

- Look for relationships between the DAGs representing $g(x)$ and $\sigma g(x\pi)$

Example

$$\begin{aligned} \mathsf{C}_0 : x_6x_7 + x_8x_9 &= 1 \\ \mathsf{C}_1 : x_6x_8 + x_7x_9 &= 1 \end{aligned}$$



- G_{DAG} = **set of automorphisms of expression DAG fixing: (a) root node set having same constr. direction and coeff. (constraint permutations), (b) operators with same label and rank and (c) leaf node set (variable permutations)**

```
Dreadnaut version 2.4 (32 bits).
> n=10 g 2 3; 4 5; 6 7; 8 9; 6 8; 7 9.  f=[0:1|2:5|6:9] x
(4 5)(6 7)(8 9)      !variable permutations
(2 3)(6 8)(7 9)      !operator permutations
(0 1)(2 4)(3 5)(7 8) !constraint permutation
```

- G_P is the projection of G_{DAG} to variable indices $\langle (6,7)(8,9), (6,8)(7,9), (7,8) \rangle \cong D_8$

Node colors 1

Colors on the DAG nodes are used to identify those subsets of nodes which can be permuted

1. Root nodes (i.e. constraints) can be permuted if they have the same RHS
2. Operator nodes (including root nodes) can be permuted if they have the same DAG rank and label
3. If an operator node is non-commutative, then the order of the children node must be maintained
4. Constant nodes can be permuted if they have the same DAG rank level and value
5. Variable nodes can be permuted if they have the same bounds and integrality constraints

Node colors 2

- Formalize by equivalence relations on sets: $\mathscr{R}$ =roots, $\mathscr{O}$ =operators, $\mathscr{C}$ =constants, $\mathscr{V}$ =variables
- Let $\mathcal{V}$ be the set of all nodes of the DAG; for all $x, y \in \mathcal{V}$:
  1. $x \sim_R y$ if $x, y \in \mathscr{R} \wedge \text{RHS}(x) = \text{RHS}(y)$ or $x, y \notin \mathscr{R}$
  2. $x \sim_O y$ if $x, y \in \mathscr{O} \wedge \text{level}(x) = \text{level}(y) \wedge \text{label}(x) = \text{label}(y) \wedge (\text{order}(x) = \text{order}(y)$ if x, y noncommutative$)$ or $x, y \notin \mathscr{O}$
  3. $x \sim_C y$ if $x, y \in \mathscr{C} \wedge \text{value}(x) = \text{value}(y) \wedge \text{level}(x) = \text{level}(y)$ or $x, y \notin \mathscr{C}$
  4. $x \sim_V y$ if $x, y \in \mathscr{V} \wedge \text{limits}(x) = \text{limits}(y) \wedge \text{integer}(x) = \text{integer}(y)$ or $x, y \notin \mathscr{V}$
- *Define an integral function* color : $\mathcal{V} \to \mathbb{N}$ *s.t.* $\forall\, x, y \in \mathcal{V}$ $(\text{color}(x) = \text{color}(y)$ iff $x \sim_R y \wedge x \sim_O y \wedge x \sim_C y \wedge x \sim_V y)$
- color *is itself an equivalence relation (call it $\sim$) and partitions $\mathcal{V}$ in disjoint sets* $V_1, \ldots, V_p$

MINLP problem groups

- Let P be a MINLP and $D = (\mathcal{V}, \mathcal{A})$ be the DAG of P
- Let G_{DAG} be the group of automorphisms of D that fix each class in $\mathcal{V}/\sim$
- Define $\phi : G_{\text{DAG}} \to S_n$ by $\phi(\pi) =$permutation on $\mathscr{V}$ (set of variable nodes) induced by π; then

  Thm.

  ϕ is a group homomorphism and Im$\phi \cong G_P$
- Hence can find G_P by computing Imϕ
- Although the complexity status (**P**/**NP**-complete) of the GRAPH ISOMORPHISM problem is currently unknown, `nauty` is a practically efficient software for computing G_{DAG}
- Also, MILPs are MINLPs! (can apply same methods)

Symmetries in the MIPLib3

Instance	G_P
`air03.mod`	$(C_2)^{13}$
`arki001.mod`	S_{38}
`enigma.mod`	C_2
`gen.mod`	$(C_2)^2 \times D_8 \times S_6$
`fiber.mod`	C_2
`harp2.mod`	C_2
`mas74.mod`	$C_2 \times C_2$
`mas76.mod`	$C_2 \times C_2$
`misc03.mod`	S_3
`misc06.mod`	$(C_2)^2 \times (S_3)^2 \times S_4$
`misc07.mod`	S_3
`mitre.mod`	$(C_2)^7$
`noswot.mod`	C_2
`nw04.mod`	C_2
`p0201.mod`	C_2
`p0282.mod`	$(C_2)^3 \times (S_3)^3$
`p0548.mod`	$(C_2)^7$
`p2756.mod`	$(C_2)^{32}$
`qiu.mod`	$C_2 \times S_4$
`rgn.mod`	S_5
`rout.mod`	S_5
`stein27.mod`	$((C_3)^3 \ltimes PSL(3,3)) \ltimes C_2$
`swath.mod`	S_{79}
`vpm1.mod`	S_{48}
`vpm2.mod`	$(S_3)^2 \times S_4 \times S_5$

Instance	*Error*
`mkc.mod`	RAM (36 gens)
`seymour.mod`	RAM (78 gens)

All others: $G_P = \{e\}$

All instances have been pre-solved by AMPL

Symmetries in the MIPLib2003

Instance	G_P
`arki001.mod`	S_{48}
`fiber.mod`	C_2
`glass4.mod`	C_2
`mas74.mod`	$C_2 \times C_2$
`mas76.mod`	$C_2 \times C_2$
`misc07.mod`	S_3
`mzzv11.mod`	$(C_2)^{155}$
`mzzv42z.mod`	$(C_2)^{110}$
`noswot.mod`	C_2
`opt1217.mod`	C_2
`p2756.mod`	$(C_2)^{32}$
`protfold.mod`	$(C_2)^2$
`qiu.mod`	$C_2 \times S_4$
`rout.mod`	S_5
`timtab1.mod`	C_2
`timtab2.mod`	C_2

Instance	*Error*
`mkc.mod`	RAM (36 gens)
`seymour.mod`	RAM (78 gens)
`swath.mod`	RAM (922 gens)
`atlanta-ip`	CPU time
`dano3mip`	CPU time
`mod011.mod`	CPU time
`sp97ar.mod`	CPU time
`t1717.mod`	CPU time

All others: $G_P = \{e\}$

AMPL presolver disabled

Symmetries in the MINLPLib

Instance	G_P
cecil_13.mod	$(C_2)^9$
elf.mod	S_3
gastrans.mod	C_2
gear.mod	D_8
gear2.mod	D_8
gear3.mod	D_8
gear4.mod	D_8
hmittelman.mod	C_2
lop97icx.mod	$(C_2)^7 \times S_{762}$
nuclear14.mod	S_6
nuclear24.mod	S_6
nuclear25.mod	S_5
nuclear49.mod	S_7
nuclearva.mod	S_3
nuclearvb.mod	S_3
nuclearvc.mod	S_3
nuclearvd.mod	S_3
nuclearve.mod	S_3
nuclearvf.mod	S_3
nvs09.mod	S_{10}
product.mod	S_{50}
risk2b.mod	$(C_2 \times S_3 \times S_6 \times S_{13})^3$
super2.mod	$(C_2)^9 \times (S_3)^2$
super3.mod	$(C_2)^9 \times (S_3)^2$
synheat.mod	S_4

Instance	*Error*
qap.mod	CPU time
qapw.mod	CPU time

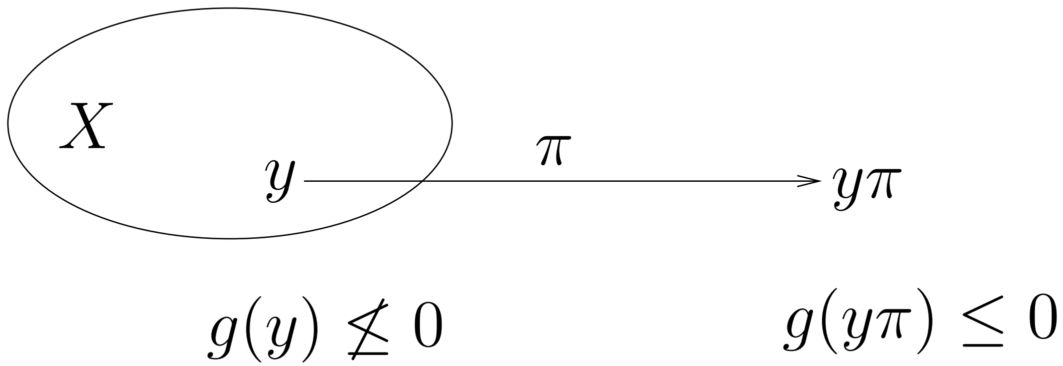
All others: $G_P = \{e\}$

All instances have been pre-solved by AMPL

Breaking symmetries

Defn.

Given a permutation $\pi \leq S_n$ acting on the component indices of the vectors in a given set $X \subseteq \mathbb{R}^n$, the constraints $g(x) \leq 0$ (that is, $\{g_1(x) \leq 0, \ldots, g_q(x) \leq 0\}$) are *symmetry breaking constraints* (SBCs) with respect to π and X if there is $y \in X$ such that $g(y\pi) \leq 0$.



Defn.

Given a group G, $g(x) \leq 0$ are SBCs w.r.t G and X is there is $y \in XG$ such that $g(y) \leq 0$.

Usually $y\pi$ is an optimum, but not all optima satisfy the SBCs

SBCs and narrowings

Adjoining SBCs to an MP formulation provides a valid narrowing

Thm.

If $g(x) \leq 0$ are SBCs for any subgroup G of G_P and $\mathcal{G}(P)$, then the problem Q obtained by adjoining $g(x) \leq 0$ to the constraints of P is a narrowing of P.

Notation: $g[B](x) \leq 0$ *if* $g(x)$ *only involve variable indices in* B

Conditions allowing adjunctions of many SBCs

Thm.

Let $\omega, \theta \subseteq \{1, \ldots, n\}$ be such that $\omega \cap \theta = \emptyset$. Consider $\rho, \sigma \in G_P$, and let $g[\omega](x) \leq 0$ be SBCs w.r.t. $\rho, \mathcal{G}(P)$ and $h[\theta](x) \leq 0$ be SBCs w.r.t. $\sigma, \mathcal{G}(P)$. If $\rho[\omega], \sigma[\theta] \in G_P[\omega \cup \theta]$ then the system of constraints $c(x) \leq 0$ consisting of $g[\omega](x) \leq 0$ and $h[\theta](x) \leq 0$ is an SBC system for $\rho\sigma$.

SBCs from orbits

Let Ω be the set of nontrivial orbits of the regular action of G_P on $\{1, \ldots, n\}$

Thm.

Let $\omega \in \Omega$. The constraints

$$\forall j \in \omega \smallsetminus \{\min \omega\} \quad x_{\min \omega} \leq x_j. \tag{4}$$

are SBCs with respect to G_P.

Notation: $G[\omega]$ *is the transitive constituent of* G *on its orbit* ω

Thm.

Provided $G_P[\omega] = \mathsf{Sym}(\omega)$, the following constraints:

$$\forall\ j \in \omega^- \quad x_j \leq x_{j^+} \tag{5}$$

are SBCs with respect to G_P.

Automatic SBC generation

1. Transform MINLP from AMPL input format into a DAG representation (ROSE)
2. Compute node colors according to relation $\sim$ defined above (ROSE)
3. Compute G_{DAG} (nauty)
4. Compute Imϕ (gap)
5. Compute nontrivial orbits Ω (gap)
6. Generate SBCs (4) or (5) according to the structure of $G_P[\omega]$, where ω is the longest orbit in Ω (gap)
7. If conditions hold, try to generate compatible SBCs from other orbits (gap)

ROSE=Reformulation/Optimization Software Engine; nauty=Graph Isomorphism software; gap=Group Theory software; data flow provided by Unix scripts

Tests

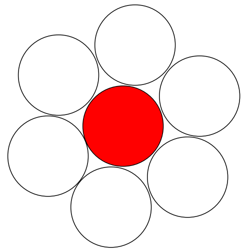
- Computed group structures for 669 instances in MIPLib3 $\cup$ MIPLib2003 $\cup$ GlobalLib $\cup$ MINLPLib
- Out of 18% instances with nontrivial groups, 74 could be solved by BB algorithms (CPLEX for MILPs; Couenne, BARON for (MI)NLPs)
- *Narrowing1*: only use (4) for longest orbit
- *Narrowing2*: try to generate as many and as tight SBCs as possible
- Test 1: over all instances
- Test 2: over a selection of 6 difficult instances with long BB runs

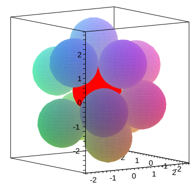
	Original problem			*Narrowing1*			*Narrowing2*		
T.	*CPU*	*Best gap*	*Nodes*	*CPU*	*Best gap*	*Nodes*	*CPU*	*Best gap*	*Nodes*
1	157263	69 2.26E4%	21.44M	152338	70 2.26E4%	14.23M	153470	72 2.26E4%	15.72M
2	815018	5 242.88%	12.26M	888089	5 219.14%	14.63M	786406	5 217.05%	11.28M

The KNP group

- Kissing Number Problem (decision version): Given integers $D, N > 1$, can N unit spheres be adjacent to a given unit sphere in $\mathbb{R}^d$?
- Formulation:

$$
\begin{array}{ll}
\max_{x,\alpha} & \alpha \\
\forall i \le N & ||x_i||^2 = 1 \\
\forall i < j \le N & ||x_i - x_j||^2 \ge \alpha \\
\multicolumn{2}{l}{\alpha \in [0,1], \forall i \le N \; x_i \in [-1,1]^D}
\end{array}
$$





- If $\alpha \ge 1$, answer YES, otherwise NO
- *The group Aut$(\mathcal{G}(P))$ has infinite (uncountable) cardinality: each feasible solution can be rotated by any angle in $\mathbb{R}^D$*; however, the **problem group** G_P is finite (permutations of spheres and/or dimensions)
- Conjecture (formulated by software): $G_P \cong S_D$
- Rewrite constraint: $||x_i - x_j||^2 = \sum_{k \le D}(x_{ik} - x_{jk})^2 = \sum_{k \le D}(x_{ik}^2 + x_{jk}^2 - 2x_{ik}x_{jk}) = 2(D - \sum_{k \le D} x_{ik}x_{jk})$
- **Conjecture becomes**: $G_P \cong S_D \times S_N$ (eventually proved correct)

The end

Thank you