Reformulations in Mathematical Programming

Leo Liberti

LIX, École Polytechnique, France

Summary of Talk

- Motivation
- Definitions and results
- Symmetry-breaking narrowing example
- Applications and Perspectives

Pregress notions

- *Mathematical Programming Formulation*: a set of parameters, decision variables, objective function(s) and constraints that precisely define an optimization problem ("model")
- *LP, MILP, NLP, MINLP*: main classes of MP formulations (linear, mixed-integer linear, nonlinear, mixed-integer nonlinear)
- *Branch-and-Bound* (BB): algorithm used for solving MILPs exactly and MINLPs at ε-optimality. *Variable Neighbourhood Search* (VNS): effective metaheuristic based on a given local search neighbourhood
- *General-purpose algorithm*: solution method targeting all problems in a given (large) class (such as e.g. MINLPs): typically, solution algorithms used to solve models

Existing definitions

- "Problem 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:** ignores feasible / locally optimal solutions

Motivation 1

Widespread use of nonlinear modelling

- Solution methods for nonlinear models are not as advanced as for linear ones
- Modelling many real-life problems as linear is innatural / difficult
- Practitioners cannot solve nonlinear models and are not always able to model linearly
- $\Rightarrow$ Inhibits spreading of mathematical programming / optimization techniques in non-specialist industrial settings

Motivation 2

Efficiency/choice of solution algorithms

- Most general purpose solution algorithms compute optima *by means* of the formulation
- Different formulations influence algorithmic behaviour
  1. In BB, alter (tighten) the bound
  2. In VNS, define different (more advantageous) neighbourhoods
- Reformulation may allow the use of a different general purpose solver (e.g. finding feasible solutions for tightly constrained MILPs by reformulation to LCPs [Di Giacomo et al., JOC 2007])

Motivation 3

Solving large-scale NLPs/MINLPs

- Solution methods for nonlinear models are not as advanced as for linear ones (again)
- Instead of solving the original (nonlinear) model, can attempt to reformulate it to a linear one
- The reformulation should be *automatic* (i.e. transparent for the user)

Current status and needs

- Google search:

  `reformulation "mathematical programming"`

  yields 419,000 hits $\Rightarrow$ everyone uses them

- No satisfactory definitions, no general theoretical results (how do we combine simple reformulations into a more complicated one? what is the size/solution difficulty of the complex reformulation?), no reformulation-based literature review, no software!

- Need for:

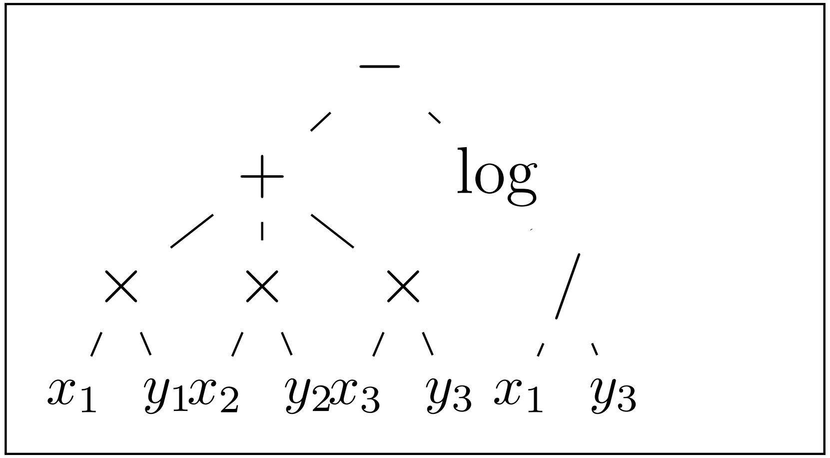
  1. *reformulation theory*
  2. *list of elementary reformulations*
  3. *reformulation software*

- Develop a **reformulation systematics**

Definitions

- Mathematical expressions as n-ary *expression trees*

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



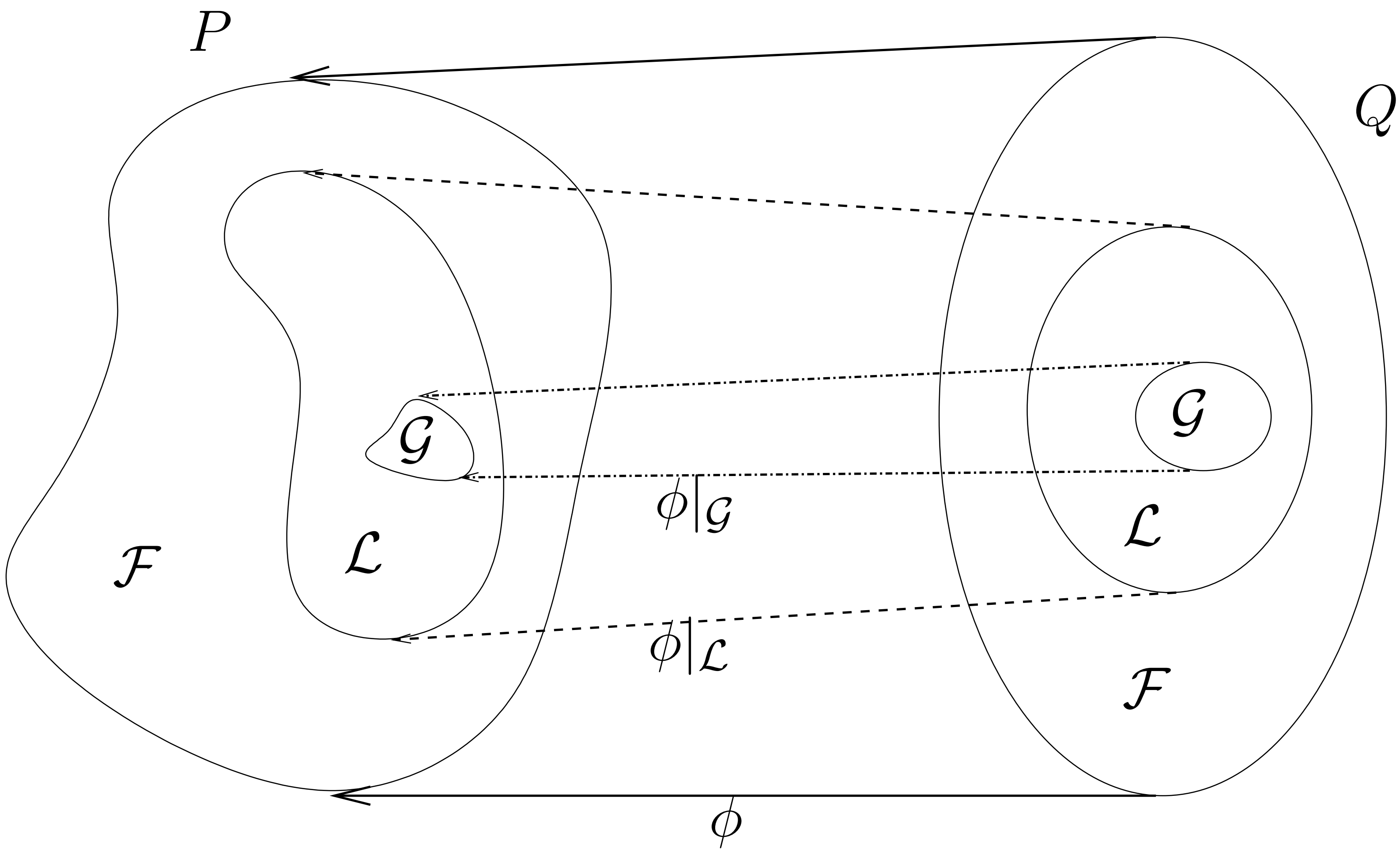
- A *formulation* P is a 7-tuple $(\mathcal{P}, \mathcal{V}, \mathcal{E}, \mathcal{O}, \mathcal{C}, \mathcal{B}, \mathcal{T})$ =(parameters, variables, expression trees, objective functions, constraints, bounds on variables, variable types)
- Constraints are encoded as triplets $c \equiv (e, s, b)$ ($e \in \mathcal{E}$, $s \in \{\leq, \geq, =\}$, $b \in \mathbb{R}$)
- $\mathcal{F}(P) =$ 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.

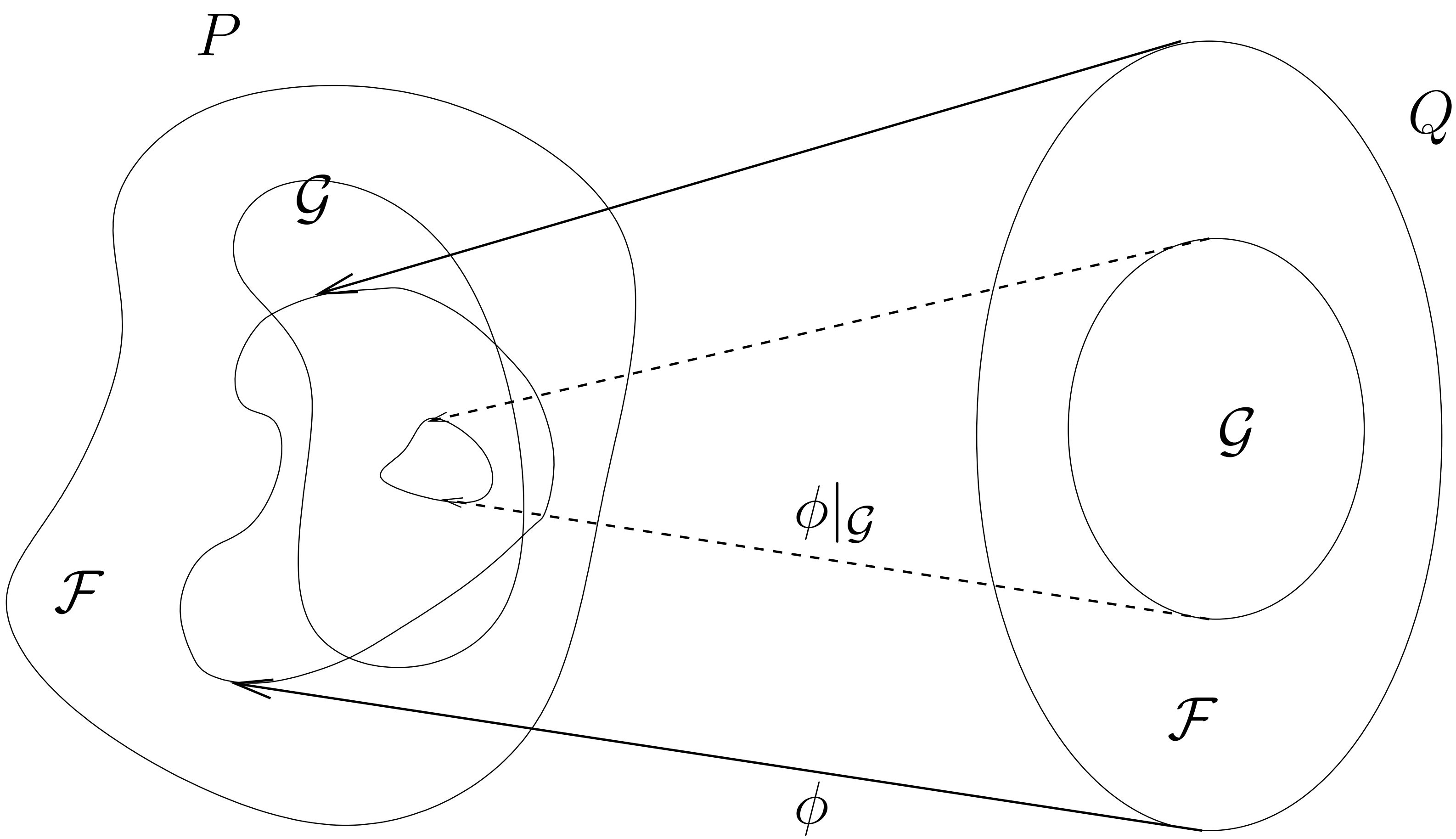
- **Opt-reformulations**: preserve all optimality properties
- **Narrowings**: preserve some optimality properties
- **Relaxations**: drop constraints / bounds / types
- **Approximations**: formulation Q depending on a parameter k such that "$\lim\limits_{k\to\infty} Q(\varepsilon)$" is an opt-reformulation, 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 $\mathcal{F}(P) \subseteq \mathcal{F}(Q)$.

Approximations

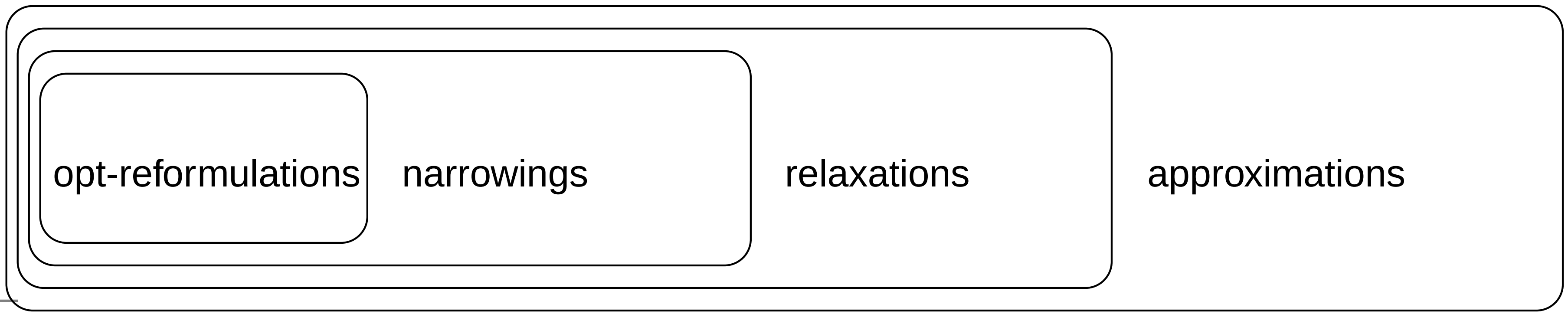
Q is an *approximation* of P if there exist: (a) an auxiliary problem Q^* of P; (b) a sequence $\{Q_k\}$ of problems; (c) an integer $k' > 0$; such that:

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.

Fundamental results

- Opt-reformulation, narrowing, relaxation, approximation are all transitive relations
- An approximation of any type of reformulation is an approximation
- A reformulation consisting of opt-reformulations, narrowings, relaxations is a relaxation
- A reformulation consisting of opt-reformulations and narrowings is a narrowing
- A reformulation consisting of opt-reformulations is an opt-reformulation



The SYMMBREAK2 narrowing 1/7

- SYMMBREAK2 motivating example
- Consider the mathematical program P:

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

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

$$
\begin{array}{l}
\{(0,1,1,1,0,0), \quad (1,0,0,0,1,1), \quad (0,0,1,1,1,0), \\
\ (1,1,0,0,0,1), \quad (1,0,1,0,1,0), \quad (0,1,0,1,0,1)\}
\end{array}
$$

The SymmBreak2 narrowing 2/7

- The group G^* of automorphisms of $\mathcal{G}(P)$ is $\langle(1,4)(2,5)(3,6),(1,5)(2,4)(3,6),(1,4)(2,6)(3,5)\rangle \cong D_{12}$
- For all $x^* \in \mathcal{G}(P)$, $Gx^* = \mathcal{G}(P) \Longrightarrow$ $\exists$ only *one* solution in $\mathcal{G}(P)$ (modulo 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*
- If we knew G^* in advance, we might add constraints eliminating (some) symmetric solutions out of $\mathcal{G}(P)$
- . . . in other words, look for a *narrowing* of P
- Can we find G^* (or a subgroup thereof) *a priori*?
- What constraints provide a valid narrowing of P excluding symmetric solutions of $\mathcal{G}(P)$?

The SymmBreak2 narrowing 3/7

- 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 SymmBreak2 narrowing 4/7

- For a packing or covering problem with $c = \mathbf{1}_n$ and $b = \mathbf{1}_m$,

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

is called the *problem symmetry group* of P

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

  Thm.

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

- Result can be extended to all MILPs [Margot02, Margot03, Margot07]
- Extension to MINLPs under way

The SymmBreak2 narrowing 5/7

Thm.

Assume:

- $\exists x^* \in \mathcal{G}(P)$ with $1 \leq \mathrm{supp}(x^*) < n-1$;
- $|G_P| > 1$.

Let $\gamma = (\gamma_1, \ldots, \gamma_k)$ with $k > 1$ be a cycle in the disjoint cycle representation of $\pi \in G_P$. Then adjoining the constraints:

$$\forall 2 \leq j \leq k \quad x_{\sigma_1} \leq x_{\sigma_k} \tag{2}$$

to P results in a nontrivial narrowing Q of P (i.e. one s.t. $|\mathcal{G}(Q)| < |\mathcal{G}(P)|$).

The SYMMBREAK2 narrowing 6/7

- *Good news*: there are automatic ways to find permutations in G_P

  One formulates an auxiliary mathematical program the solution of which encodes $\pi \in G_P$ (incidentally if $\pi = e$ this proves $G_P = \{e\}$)

- *Bad news*: the CPU time required to find permutations of G_P is prohibitively high (for now)
- *Good news*: once some $\pi \in G_P$ is known, adding constraints (2) for the longest disjoint cycle of π yields a narrowing Q computationally as tractable as P
- *Bad news*: there is an element of arbitrary choice in (2), namely that x_{σ_1} is a minimum element within $x[\sigma]$
- ...found no way (yet) to eliminate this arbitrary choice without adding more variables to Q

The SYMMBREAK2 narrowing 7/7

Very preliminary computational results on a small set of instances (some from MILPLib, some from Margot's website):

Instance	*Group*	γ	BBn(P)	BBn(Q)
`enigma`	C_2	2	3321	**269**
`jgt18`	$C_2 \times S_4$	6	**573**	1300
`oa66234`	S_3	2	0	0
`oa67233`	$C_2 \times S_4$	6	6	**0**
`oa76234`	S_3	2	0	0
`ofsub9`	$C_3 \times S_7$	21	1111044	**980485**
`stein27`	$((C_3 \times C_3 \times C_3) \ltimes PSL(3,3)) \ltimes C_2$	24	**1084**	1843
`sts27`	$((C_3 \times C_3 \times C_3) \ltimes PSL(3,3)) \ltimes C_2$	26	1317	**968**

Results are promising but not exciting
Need to improve narrowing efficacy

Other applications

- RcLin opt-reformulation: applied in (L., 4OR, 2007) to the Graph Partitioning Problem (GPP), the Multiprocessor Scheduling Problem with Communication Delays (MSPCD) and the Quadratic Assignment Problem (QAP): CPU improvement 2 Orders of Magnitude (OMs)
- RRLTRelax relaxation:
  1. used in (L. &Pantelides, JOGO, 2006) to drastically tighten the convex relaxation of pooling and blending problems from the oil industry: sBB nodes improvements 2-5 OMs
  2. use in (Lavor et al., EPL, 2007 and L. et al., DAM, accepted) to be able to compute molecular orbitals solving Hartree-Fock systems by sBB (impossible without it)
- InnerApprox approximation: found feasible solutions of a large-scale (25-50K bin vars/constrs) convex MINLP occurring in a sphere covering problem arising in the configuration of gamma-ray radiotherapy units (using CPLEX)

Perspectives

- Principal Investigator for the Automatic Reformulation Search (ARS) project funded by ANR, and part of a WP in the EU project "Morphex": extend the reformulation library and implement a prototype of the automatic reformulation software
- Reformulation techniques offer high didactical value when teaching modelling courses
- My bet : successful algorithms for large scale MINLPs will *have* to employ automatic reformulation techniques to some extent
- My regret : there is a widespread belief that reformulations are "just" modelling tricks, and to dismiss them as implementation details, even though computational results improvements due to reformulations are major.

The end

Thank you