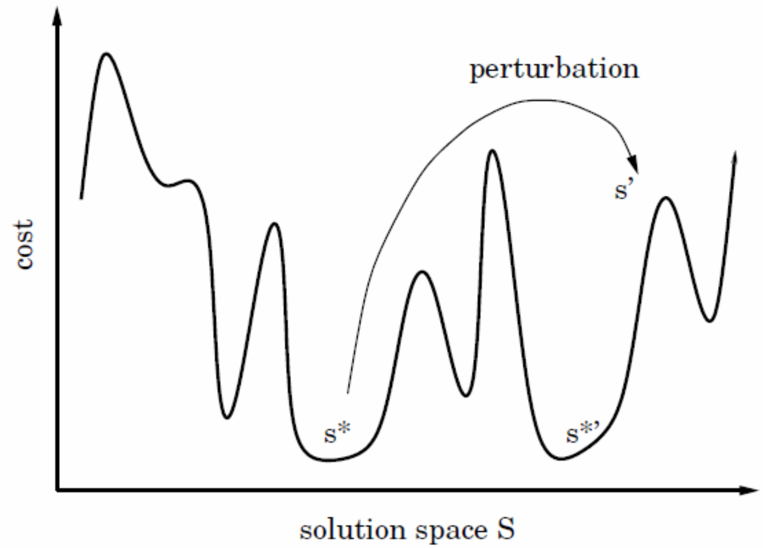
Toward a MIP cut meta-scheme

Matteo Fischetti, DEI, University of Padova

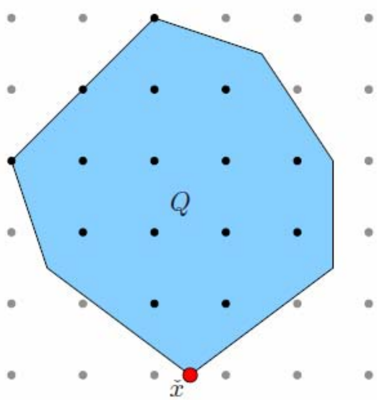


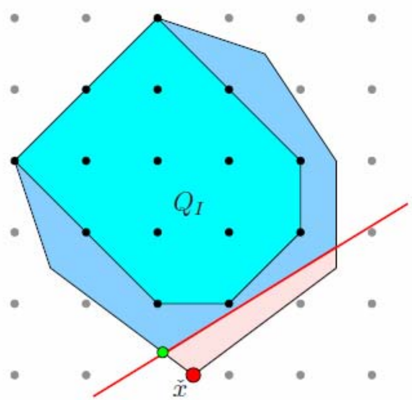
Mixed-Integer Programs (MIPs)

- We will concentrate on general MIPs of the form

  $$\min \{\, c\, x : A\, x = b,\ x \geq 0,\ x_j \textit{ integer } \text{for some } j \,\}$$

- Two main story characters
  - The **LP relaxation** (beauty): easy to solve
  - The **integer hull** (the beast): convex hull of MIP sol.s, hard to describe

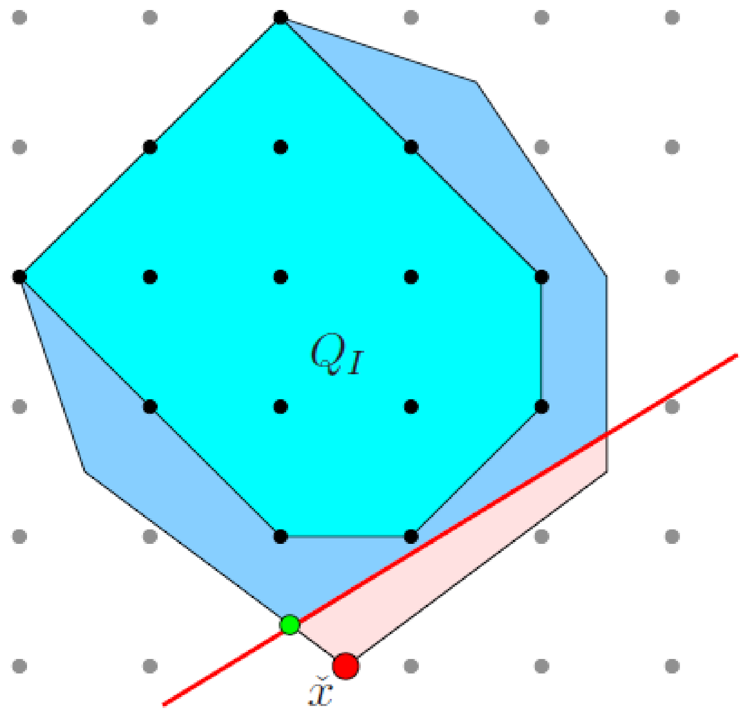




Cutting planes (cuts)

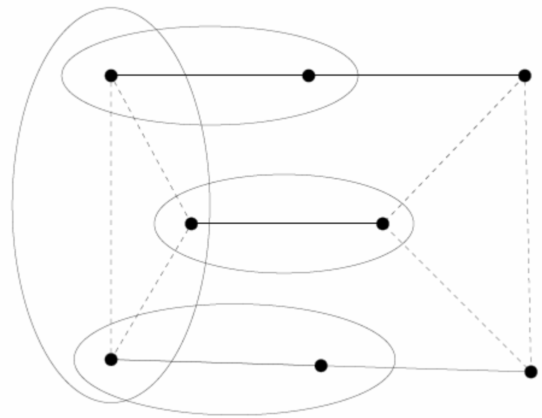
- **Cuts:** linear inequalities valid for the integer hull (but not for the LP relaxation)

- **Questions:**
  - **How to compute?**
  - Are they really useful?
  - If potentially useful, how to use them?

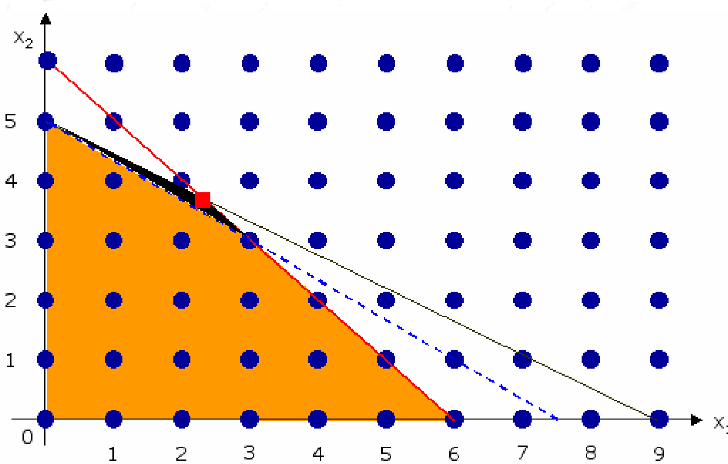


How to compute the cuts?

- **Problem-specific** classes of cuts (with nice theoretical properties)
  - Knapsack: cover inequalities, …
  - TSP: subtour elimination, comb, clique tree, …



- **General** MIP cuts only derived from the input model
  - Cover inequalities
  - Flow-cover inequalities
  - …
  - **Gomory cuts** (perhaps the most famous class of MIP cuts)



Gomory cuts: basic version

- Basic version for pure-integer MIPs (no continuous var.s): **Gomory fractional cuts**, also known as **Chvàtal-Gomory cuts**

- Given any equation satisfied by the LP-relaxation points $$\sum_j \overline{a}_j x_j = \overline{b}$$
  - 1. **relax** to its ≤ form $$\sum_j \overline{a}_j x_j \le \overline{b}$$
  - 2. **relax** again by rounding down all left-hand-side coeff.s $$\sum_j \lfloor \overline{a}_j \rfloor x_j \le \overline{b}$$
  - 3. **improve** by rounding down the right-hand-side value $$\sum_j \lfloor \overline{a}_j \rfloor x_j \le \lfloor \overline{b} \rfloor$$

- Note: all-integer coefficients (good for numerical stability)

Gomory cuts: improved version

- **Gomory Mixed-Integer Cuts (GMICs):**

$$\sum_j \lfloor \bar{a}_j \rfloor x_j \leq \lfloor \bar{b} \rfloor$$

$$\sum_j (\lfloor \bar{a}_j \rfloor + \epsilon_j) x_j + \sum_{x_j \text{ continuous}} \alpha_j x_j \leq \lfloor \bar{b} \rfloor$$

  - Some left-hand side coefficients can be increased by a fractional quantity $\varepsilon_j \geq 0$ → better cuts, though potentially less numerically stable
  - Can handle continuous variables, if any (a must for MIPs)

GMICs read from LP tableaux

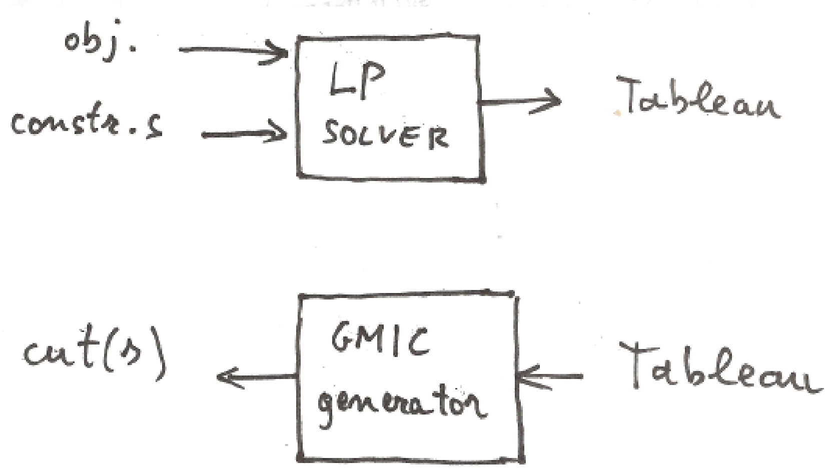
- GMICs apply a simple formula to the coefficients of a starting equation
  - Q. How to define this starting equation (crucial step)?
  - A. The LP optimal tableau is plenty of equations, just use them!

		x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
$-z$	$-\frac{25}{3}$	0	$\frac{4}{3}$	$\frac{19}{6}$	$\frac{9}{2}$	0	0	0	$\frac{7}{6}$
x_5	1	0	1	$-\frac{1}{2}$	$-\frac{3}{2}$	1	0	0	$\frac{3}{2}$
x_1	$\frac{11}{3}$	1	$-\frac{2}{3}$	$-\frac{1}{3}$	0	0	0	0	$\frac{2}{3}$
x_6	$\frac{2}{3}$	0	$\frac{1}{3}$	$\frac{1}{6}$	$-\frac{1}{2}$	0	1	0	$\frac{7}{6}$
x_7	1	0	-3	$\frac{1}{2}$	$\frac{9}{2}$	0	0	1	$-\frac{15}{2}$

The two available modules

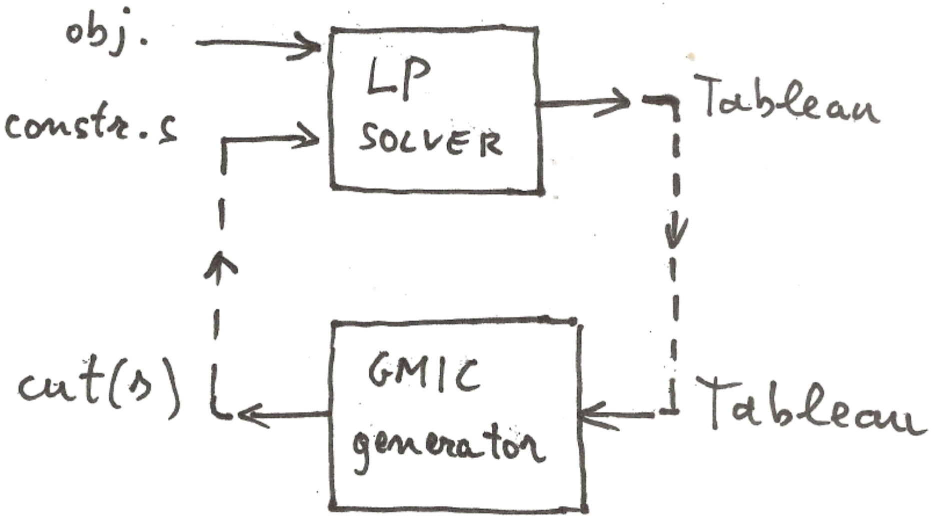
- **The LP solver**
  - Input: a set of linear constraints & objective function
  - Output: an optimal LP tableau (or basis)

- **The GMIC generator**
  - Input: an LP tableau (or a vertex x* with its associated basis)
  - Output: a *round* of GMICs (potentially, one for each tableau row with fractional right-hand side)



How to combine the two modules?

- A natural (??) **interconnection scheme** (Kelley, 1960):

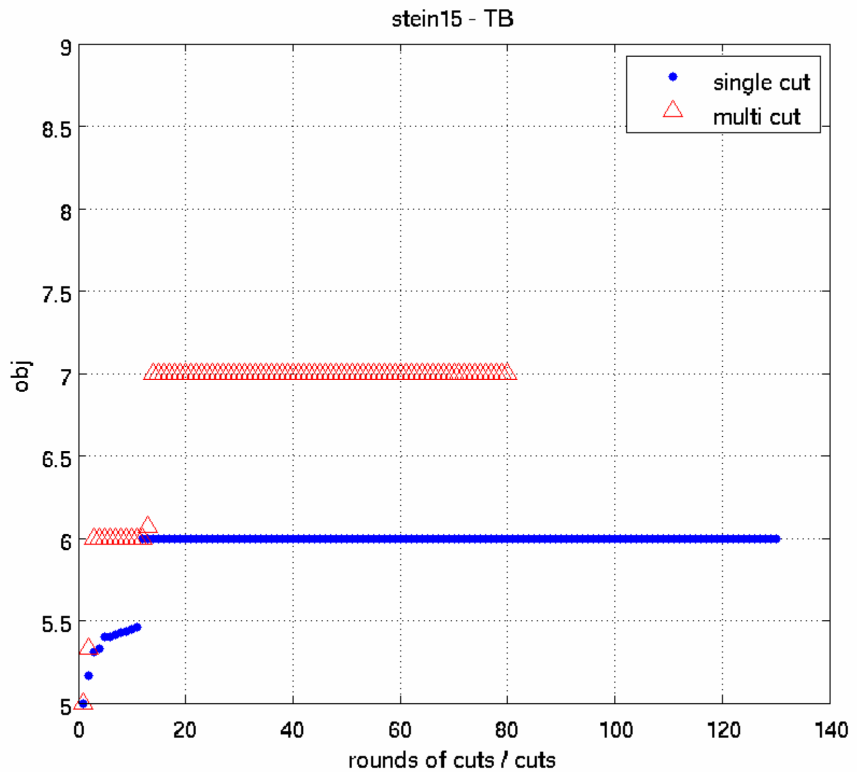


- In theory, this scheme **could** produce a finitely-convergent cutting plane scheme, i.e., an exact solution alg. only based on cuts (no branching)

8. FINITENESS PROOFS

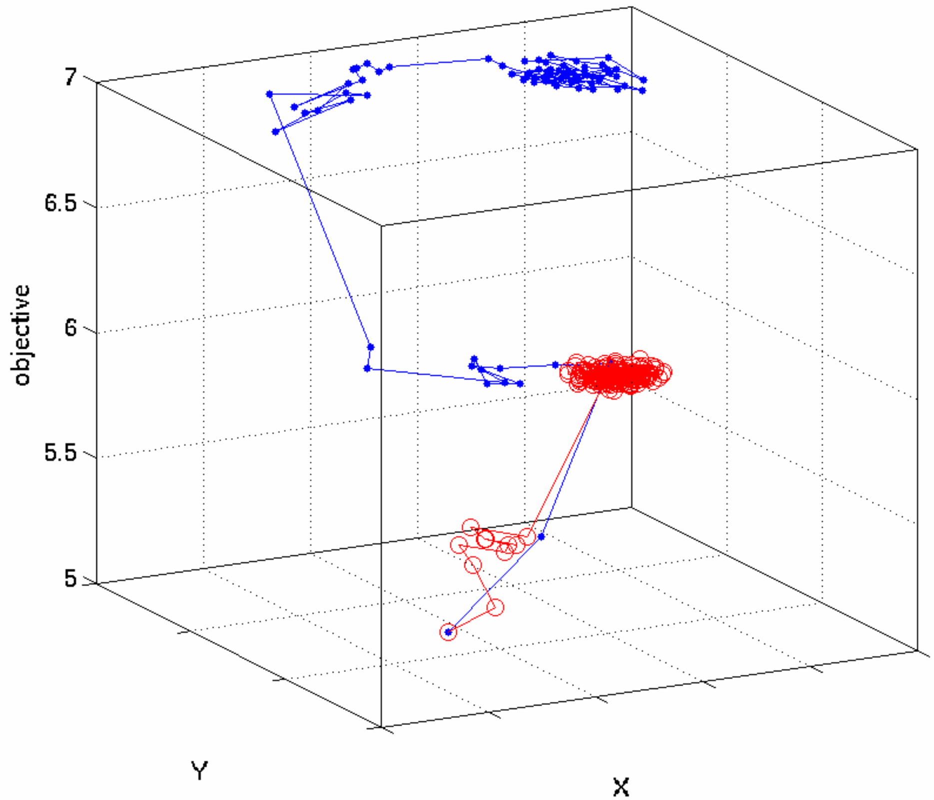
In these proofs we will use the lexicographical dual simplex method described in Section 7. It is not implied that this simplex method need be used in practice or that it is necessary to the proof. It is simply that its use in the proof has reduced the original rather long and tedious proofs to relatively simple ones.

In theory, but … in practice?



- ***Stein15***: toy set covering instance from MIPLIB
  - LP bound = **5**
  - MIP optimum = **8**
- multi cut generates ***rounds*** of cuts before each LP reopt.

LP solution trajectories

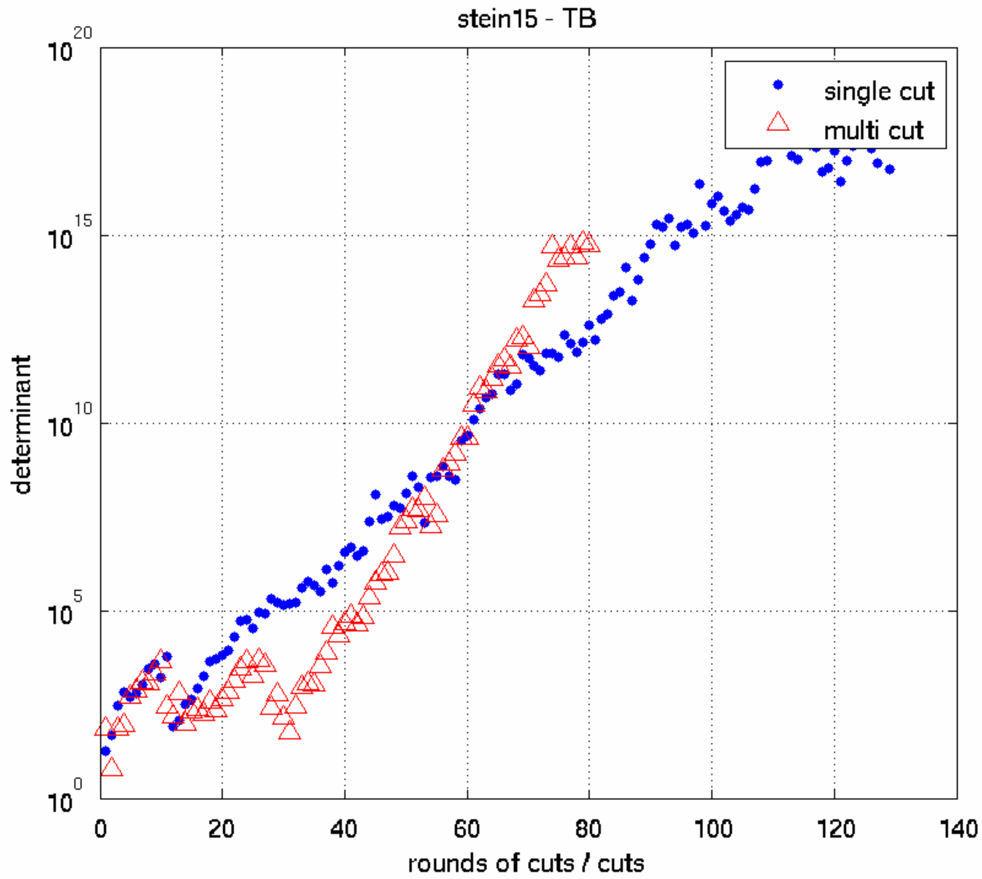


• Plot of the LP-sol. trajectories for **single-cut** (red) and **multi-cut** (blue) versions (multidimensional scaling)

(X,Y) = 2D representation of the x-space (multidimensional scaling)

Both versions collapse after a while → why?

LP-basis determinant



Exponential growth → unstable behavior!

Intuition about saturation

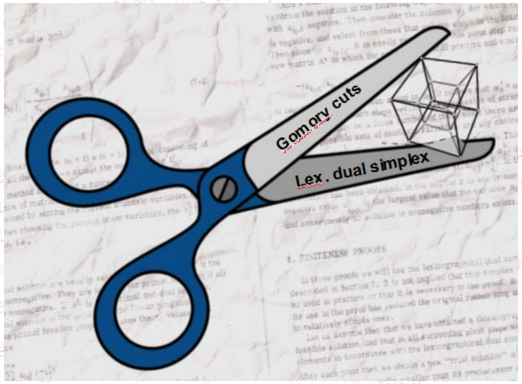
- Cuts work reasonably well on the initial LP polyhedron
  … however they create artificial vertices
  … that tend to be very close one to each other
  … hence they differ by small quantities and
    have “weird entries”
  → very like using a smoothing plane on wood

- LP theory tells that small entries in LP basic sol.s x*
  … require a large basis determinant to be described
  … and large determinants amplify the issue and create numerically unstable tableaux

- Kind of **driving a car on ice** with flat tires :
  - Initially you have some grip
  - … but soon wheels warm the ice and start sliding
  - … and the more gas you give the worse!

Gomory's convergent method

- For pure integer problems (all-integer data) Gomory proved the existence of a finitely-convergent solution method only based on cuts, but one has to follow a **rigid recipe**:
  - use **lexicographic** optimization (a must!)
  - use the objective function as a source for GMICs
  - be **really patient** (don't unplug your PC if nothing seems to happen…)

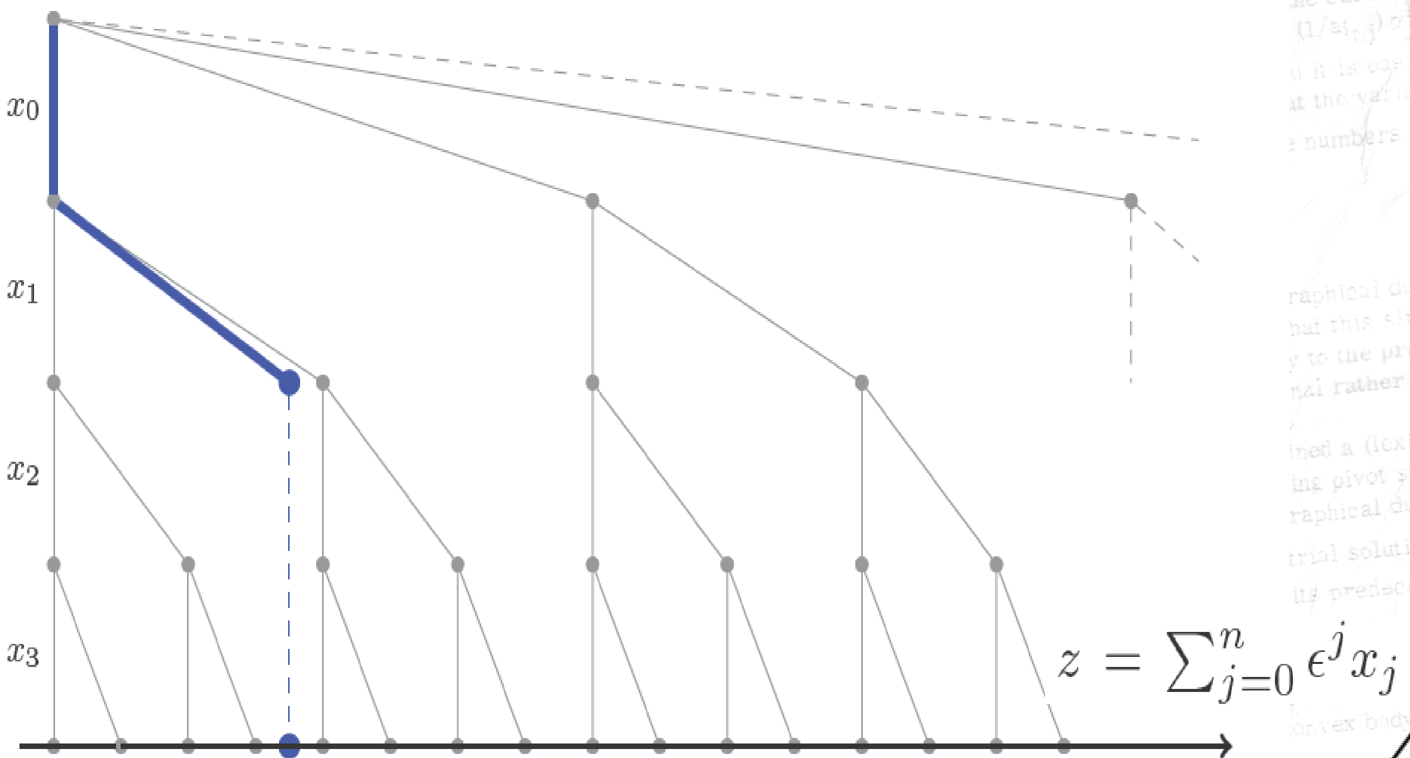


- **Finite convergence** guaranteed by an enumeration scheme hidden in lexicographic reoptimization (this adds anti-slip **chains** to Gomory's wheels…)

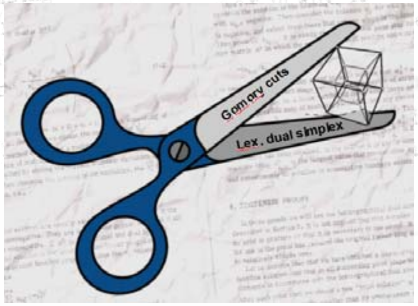
  → **safe but slow (like driving on a highway with chains…)**

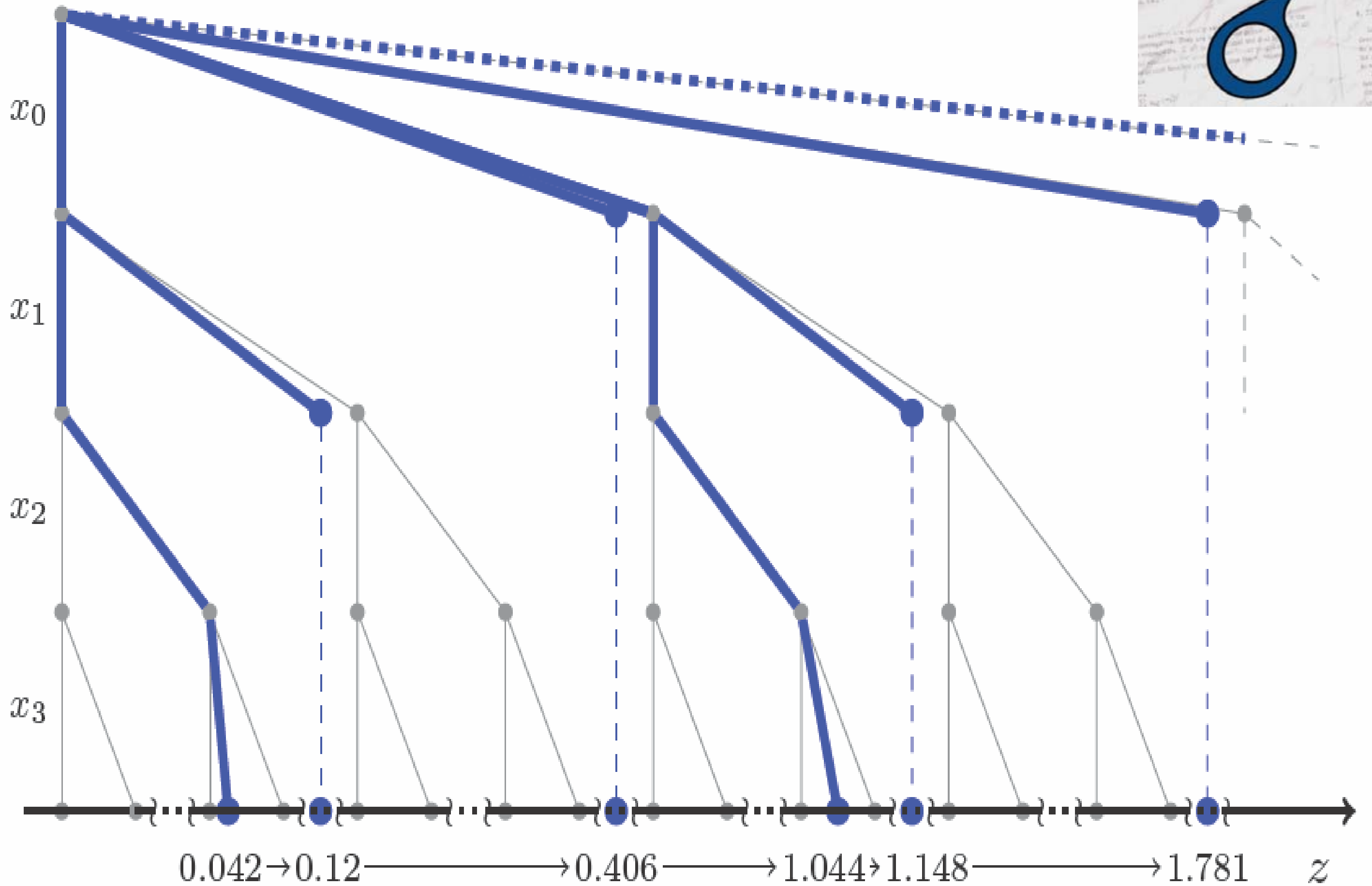
The underlying enumeration tree

- Any LP solution x^* can be visualized on a lex-tree ($x_o = c\ x$ = objective)
- The structure of the tree is fixed (for a given lex-order of the var.s)
- Leaves correspond to integer sol.s of increasing lex-value (left to right)

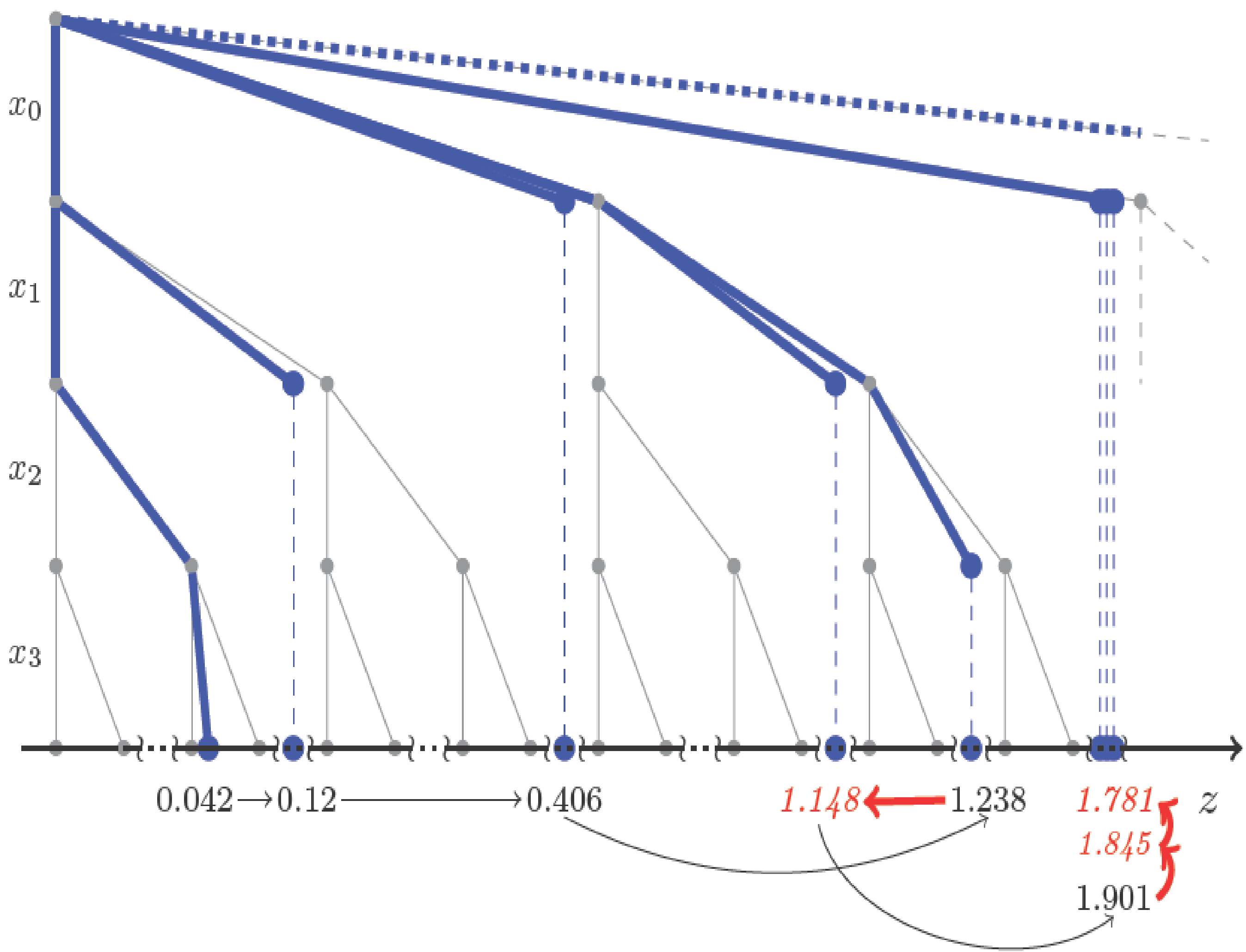


The "good" Gomory (+ lex)



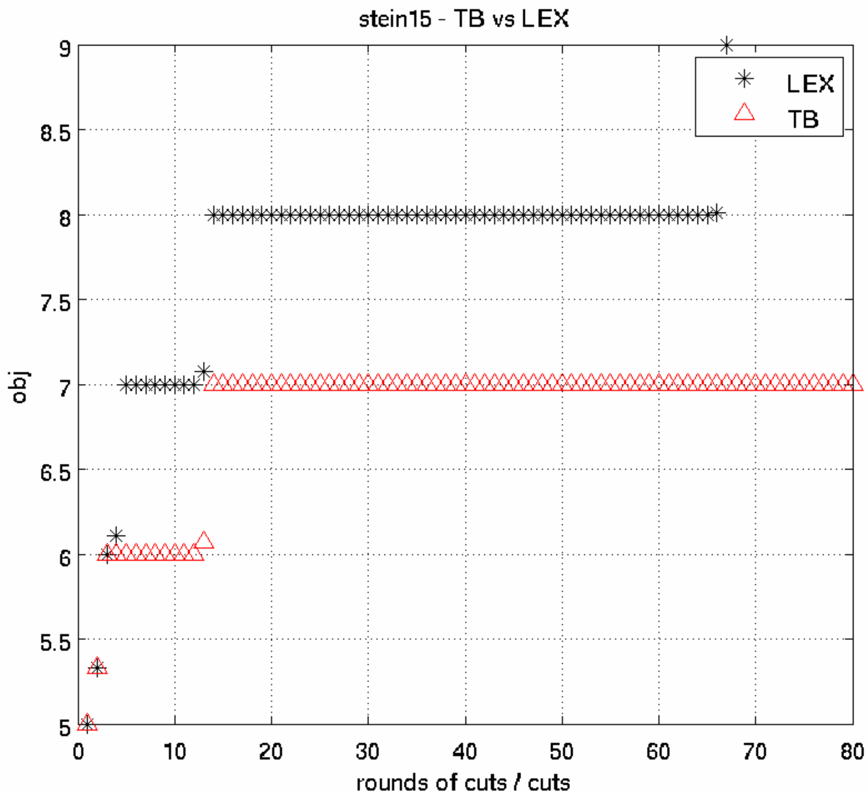


The “bad” Gomory (no lex)



lex-value z may decrease → risk of loop in case of naïve cut purging!

Good Gomory: Stein15 (LP bound)

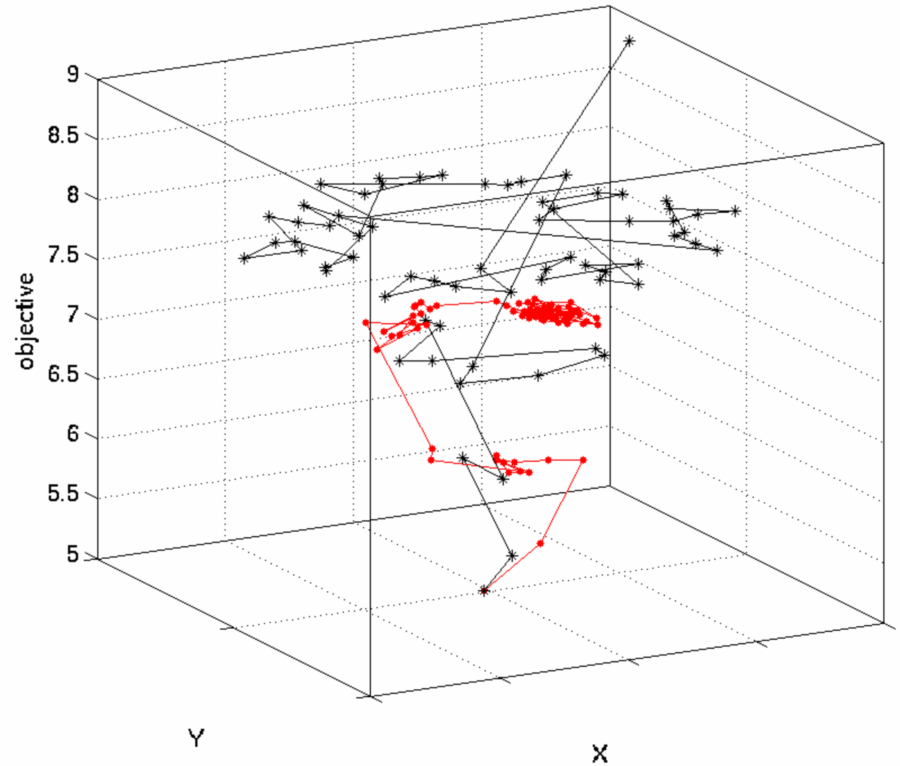


LP bound = **5**; ILP optimum = **8**

TB = no-lex **multi-cut** vers. (as before)

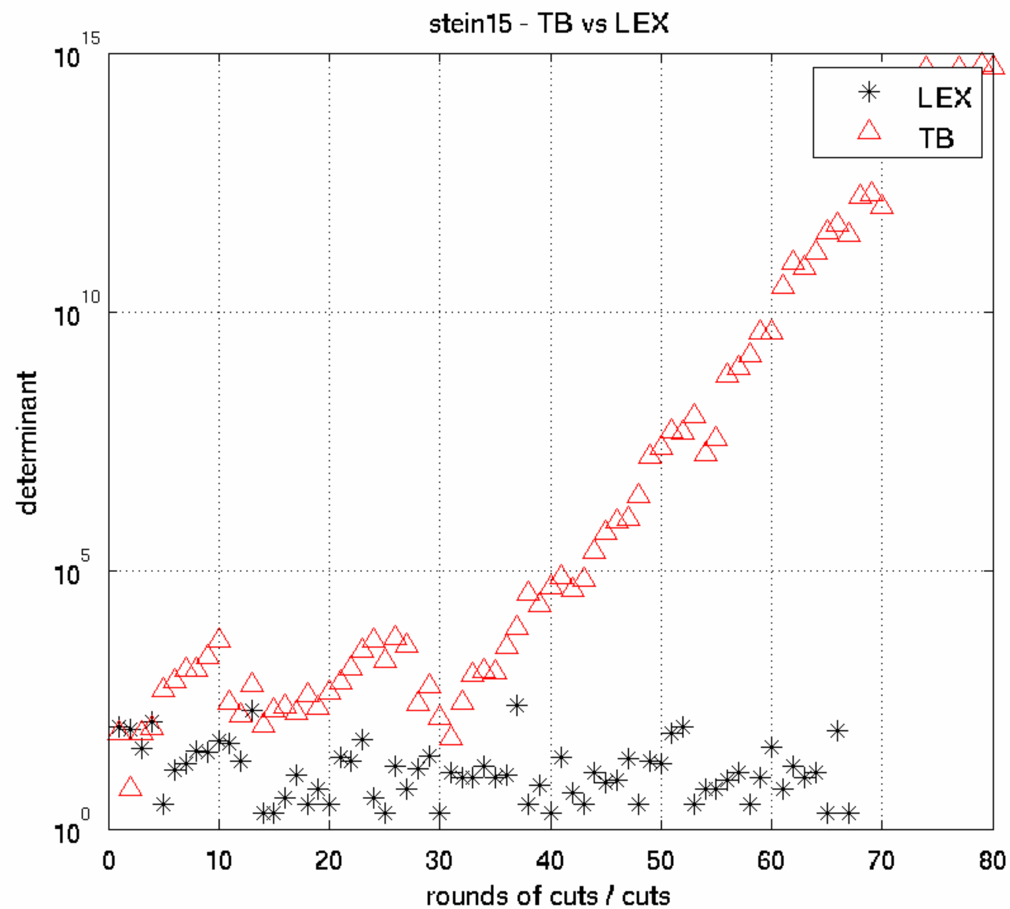
LEX = single-cut with **lex-optimization**

Good Gomory: Stein15 (LP sol.s)



Plot of the LP-sol. trajectories for **TB** (**red**) and **LEX** (**black**) versions

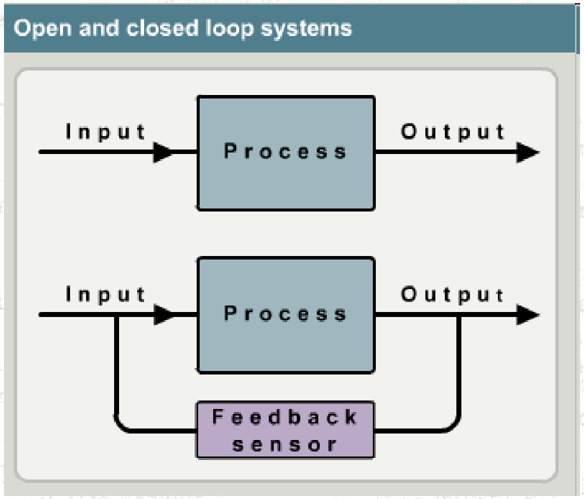
Good Gomory: Stein15 (determinant)



TB = multi-cut vers. (as before) **LEX** = single-cut with lex-opt.

So, what is wrong with Gomory?

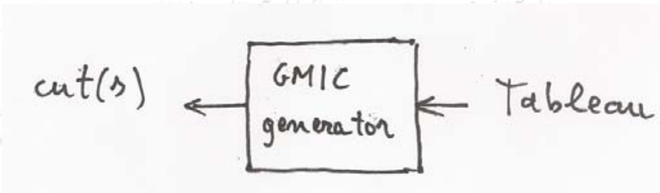
- GMICs are not bad by themselves
- What is problematic is their use in a naïve Kelley's scheme
- A main issue with Kelley is the closed-loop nature of the interconnection scheme
- Closed-loop systems are intrinsically prone to instability…
- … unless a filter (like lex-reopt) is used for input-output decoupling



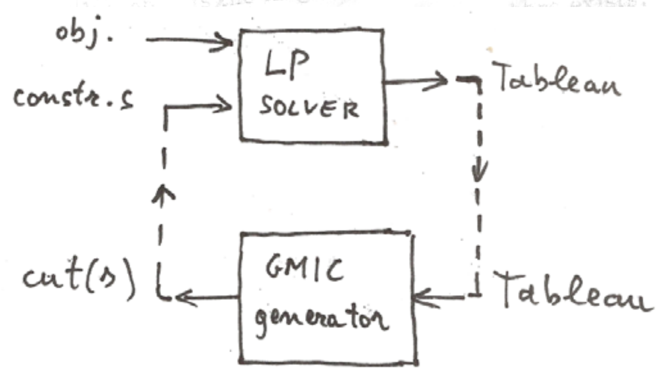
Brainstorming about GMICs

- Ok, let's think "laterally" about this cutting plane stuff

- We have a cut-generation module that needs an LP tableau on input



- … but we cannot short-cut it directly onto the LP-solver module (soon the LP determinant burns!)



- **Shall we forget about GMICs and look for more fancy cuts,**
- **… or we better design a different scheme to exploit them?**

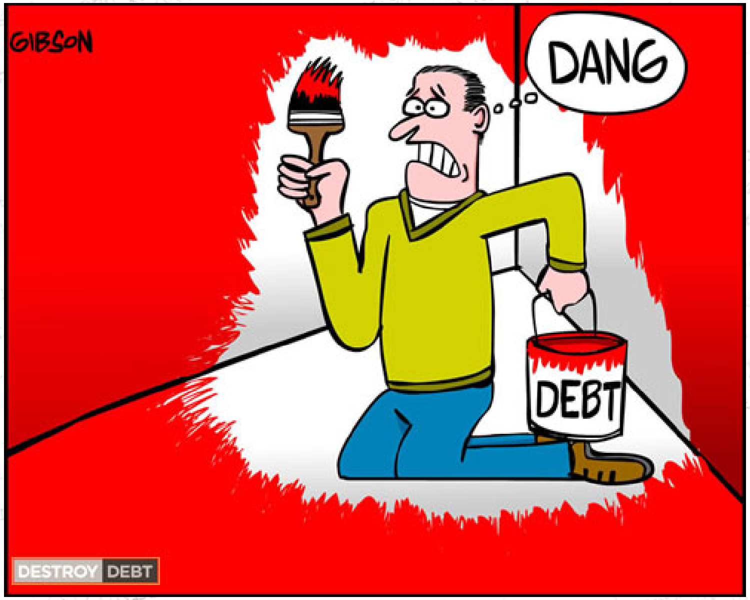
Brainstorming about GMICs

- This sounds like *déjà vu*…

  … we have a **simple module** that works well in the beginning

  … but soon it **gets stuck in a corner**

- … Where did I hear this?

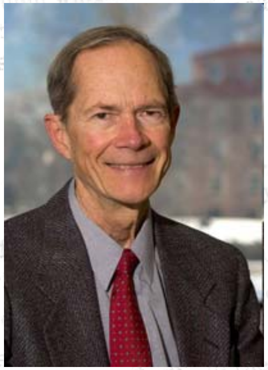


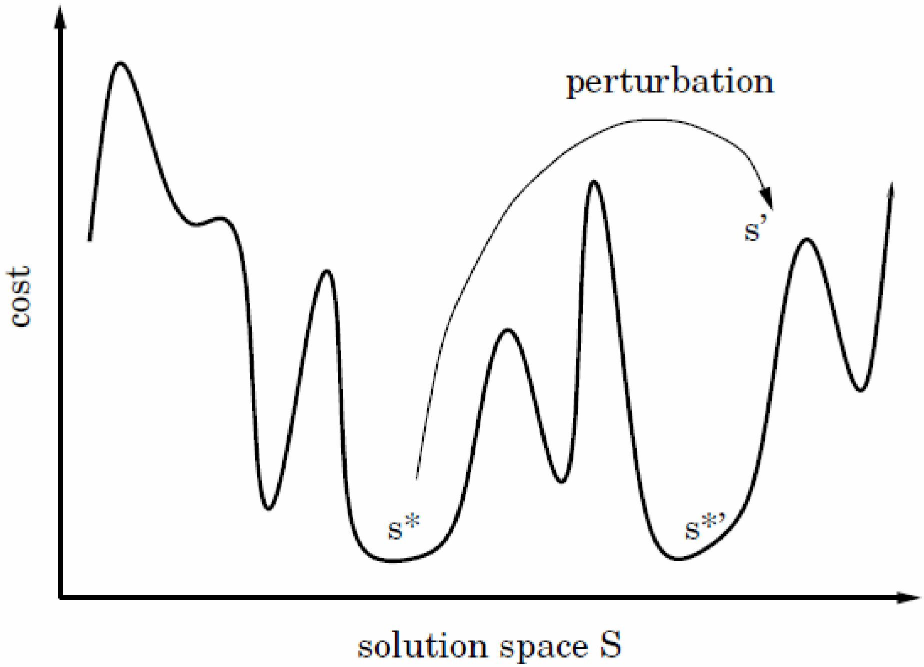
- **Oh yeah! It was about heuristics and metaheuristics…**

We need a META-SCHEME for cut generation !

Toward a meta-scheme for MIP cuts

- We stick with **simple** cut-generation modules; if we get into trouble…
  … we don't give-up but apply a **diversification step** (isn't this the name, Fred?) to perturb the problem and explore a different "**cut neighborhood**"





A diving meta-scheme for GMICs

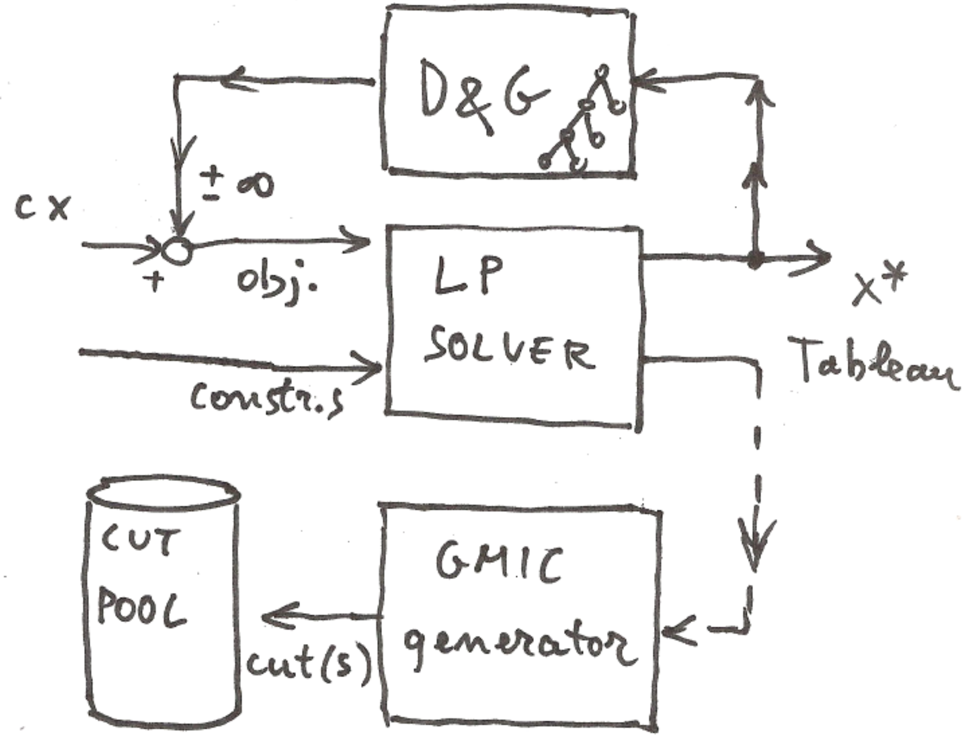
- A main source of feedback is the presence of previous GMICs in the LP → avoid modifying the input constr.s, use the obj. function instead

- A kick-off (very simple) scheme:

  Dive & Gomory

  Idea: Simulate enumeration by adding/subtracting a bigM to the **cost** of some var.s and apply a classical GMIC generator to each LP

  … but **don't add the cuts to the LP** (just store them in a cut pool for future use…)



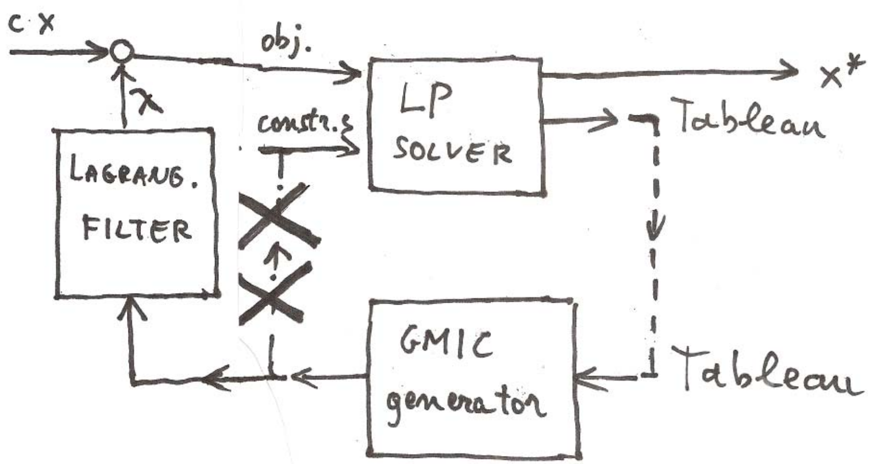
D&G results

method	MIPLIB 2003 cl.gap	time (s)
1gmi	18.3%	0.54
Lift&Project	30.7%	95.23
Dive&Gomory	31.5%	7.45

cl.gap = integrality gap (MIP opt. – LP opt.) closed by the methods

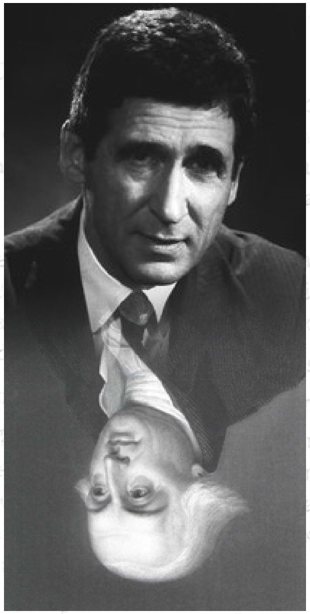
A Lagrangian filter for GMICs

- As in Dive&Gomory, diversification can be obtained by changing the objective function passed to the LP-solver module so as to produce LP tableaux that are only **weakly correlated** with the LP optimal solution x* that we want to cut



- A promising framework is ***relax-and-cut*** where GMICs are not added to the LP but immediately relaxed in a **Lagrangian** fashion

→ very interesting results to be reported by Domenico (Salvagnin) in his Friday's talk about "**LaGromory cuts**"…



Thank you for your attention...

… and of course for not sleeping…

… (is it over … already?)