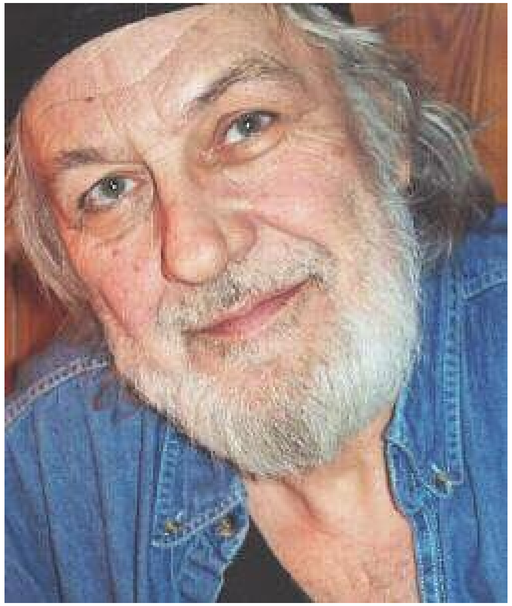
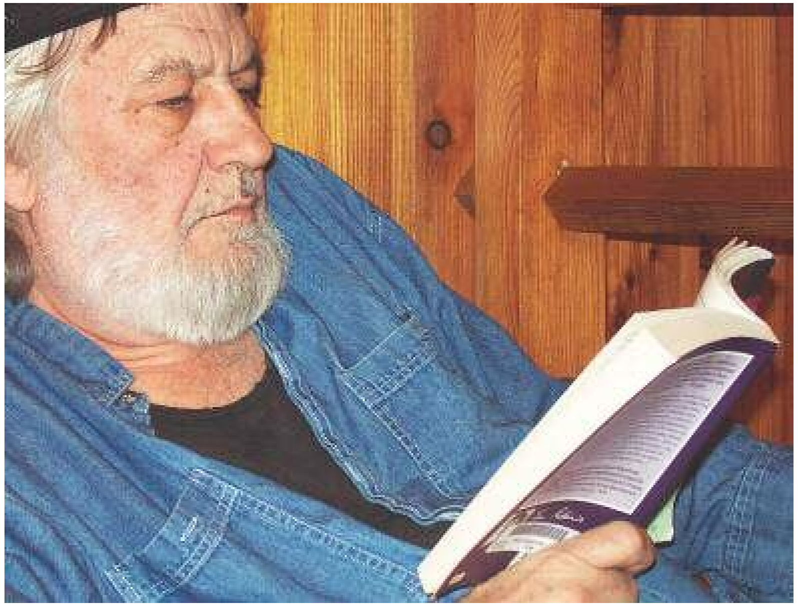
0-1/2 CUTS:
A COMPUTATIONAL STUDY

G. Andreello, DEI, University of Padova

A.Caprara, DEIS, University of Bologna

M. Fischetti, DEI, University of Padova

Thank God! Maybe this time … it works …

Hey … those are MY Aussois pictures …

Hmmm, this BLOSSOM stuff is so boring …
but let me postpone my nap …
maybe new pictures will pop up!!

0-1/2 CUTS

Usual stuff…

- A polyhedron

$$P := \{x \in R^n : Ax \leq b\},\quad \text{where} \quad A \in Z^{mxn}, b \in Z^m$$

- and the convex hull of its integer points

$$P_I := conv\{x \in Z^n : Ax \leq b\}$$

- Chvatàl-Gomory cuts for P_I :

$$\lambda^T Ax \leq \lfloor \lambda^T b \rfloor$$

valid for each $\lambda \geq 0 \;:\lambda^T A \in Z^n$

w.l.o.g. $\lambda \in [0,1)^m : \lambda^T b - \lfloor \lambda^T b \rfloor > 0$

- **0-1/2 cuts:** as before, but $\lambda \in \{0,1/2\}^m$

- **0-1/2 CUT separation:** given $x^* \in P$, find

$$\lambda \in \{0,1/2\}^m : \lambda^T A \in Z^n, \lambda^T A x^* > \lfloor \lambda^T b \rfloor$$

- Can be **rephrased** as: find $\lambda \in \{0,1/2\}^m$ such that
  - $\lambda^T A \in Z^n, \lambda^T b - \lfloor \lambda^T b \rfloor = 1/2$
  - **slack** $s^* := b - Ax^* : \lambda^T s^* < 1/2$

- or equivalently: find $2\lambda =: \mu \in \{0,1\}^m$ such that

$\mu^T A$ is **even**, $\mu^T b$ is **odd**, and $\mu^T s^* < 1$

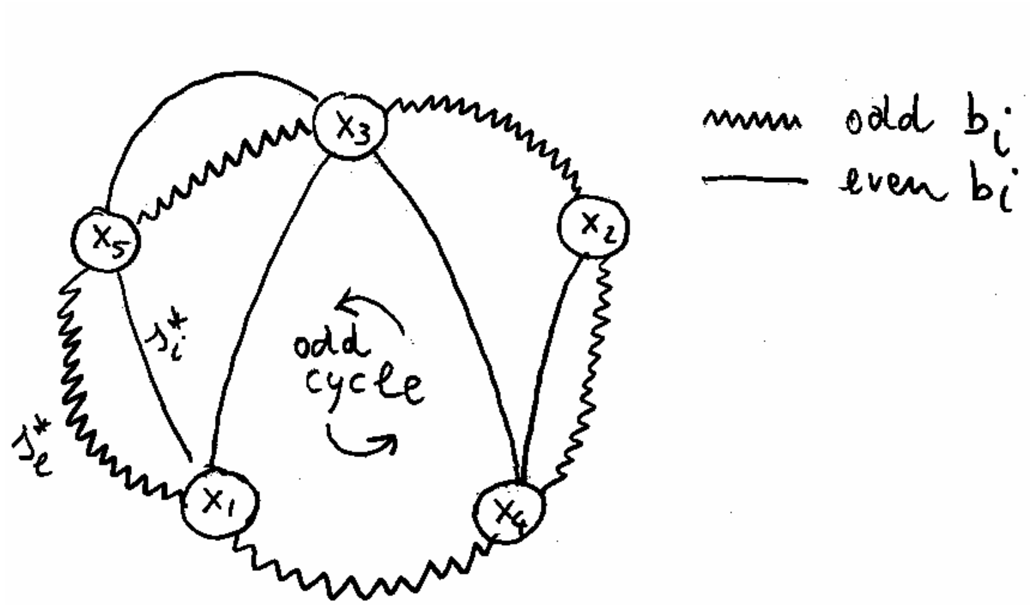
0-1/2 cuts express parity requirements of the integer solution…

- **0-1/2 CUT separation:** NP-hard in general, but polynomially solvable in some special cases

- In particular when **each row of matrix A contains no more than 2 odd coefficients**

- In this case, each inequality of the original system $a_i^T x \le b_i$ is characterized by:

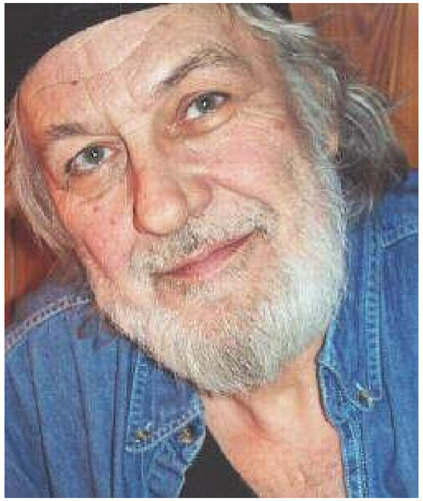
  - The **two** specific variables for which the coefficient is **odd**
  - an **odd/even** r.h.s. b_i
  - **slack** $s_i^* := b_i - a_i^T x^*$



- **0-1/2 CUT separation:** Find and **odd cycle** in the **separation graph** of weight less than 1 (polynomial)

WHAT if some inequalities have 3 or more odd coefficients?

Eureka, you RELAX



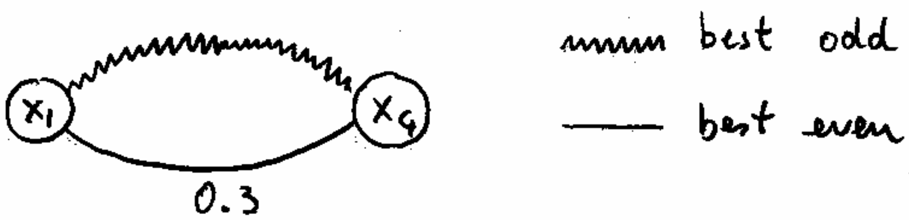
Admittedly, it does not sound as good as
"Eureka, you shrink" **but, you know, HE copyrighted everything…**

WHAT if some inequalities have 3 or more odd coefficients? Relax it…

$$x_1 + 2x_2 + x_3 - x_4 \le 1 \quad \rightsquigarrow \text{slack } 0.1$$

$$x_3 \le 1 \quad \rightsquigarrow \text{slack } 0.2$$

$$x_1 + 2x_2 + 2x_3 - x_4 \le 2 \quad \rightsquigarrow \text{slack } 0.3$$



- **Our heuristic 0-1/2 CUT separation:** do it systematically, by dynamic programming (polynomial)
- **Nice theoretical** applications**:** Linear Ordering, Clique Partitioning, etc.
- **General separation tool** to be invoked within a ILOG-Cplex 8.1 callback for general 0-1 ILP's

Cut Selection Criteria

Cuts $\gamma == \underline{a}^T \underline{x} \le b$ can be characterized in terms of:

- **Euclidean distance** from the LP solution x^*

$$dist(\underline{x}^*, \gamma) = \frac{| \underline{a}^T \underline{x}^* - b |}{\|\underline{a}\|}$$

- **Angle** between two cuts γ_i and γ_j (=1 parallel, =0 orthogonal)

$$angle(\gamma_i, \gamma_j) = \frac{\left| \underline{a}_i^T \underline{a}_j \right|}{\|\underline{a}_i\| \|\underline{a}_j\|}$$

A set of cuts $\{\gamma_i : i = 1,\ldots,k\}$ is supposed to be effective if

(1) $\forall i \quad : dist(\underline{x}^*, \gamma_i) \ge \min\_dist$

(2) $\forall i, j \quad : angle(\gamma_i, \gamma_i) \le \max\_angle$

Comparison of two algorithms X and Y over a given test bed

- The **sum of solution times**: useful, but
  - no distinction between solved and time-out instances
  - time-out instances is likely to dominate the sum.

- A **detailed table** with n. of nodes, cpu times, % gaps, etc. may be difficult to understand for **humans** (other than Paolo, of course …)

- **à *la CPLEX*** (Bixby et al.): decide a time-limit and:

  1. discard the instances solved in less than 20 seconds (say) by both X and Y (too easy)
  2. compute the average **speedup time(X) / time(Y)** on the instances solved by both X and Y
  3. count the n. of instances solved by X but not by Y, and compute the average time(X)
  4. count the n. of instances solved by Y but not by X, and compute the average rime(Y)
  5. count the n. of instances not solved by X or Y

Average: geometric mean is more robust than the arithmetic one when the standard deviation is high

Arithmetic mean: $\frac{1}{n}\left(\sum_i speedup_i\right)$ Vs. **Geometric** mean: $\left(\prod_i speedup_i\right)^{\frac{1}{n}}$

Our test bed

- IP formulations of about 100 **SAT** problems from DIMACS Challenge (**feasibility** problems)
- Objective function randomly generated (sum of some of the constraints of the formulation)
- Optimization stopped as soon as the first integer feasible solution is found

Main features:

- constraint matrix A has 0-1 coefficients (nice combinatorial structure expected)
- number of constraints is larger than number of variables
- **standard cuts from ILOG-CPLEX arsenal are ineffective (more combinatorial ones needed)**

Computational results

Reference configuration for all runs:
Default ILOG CPLEX 8.1 with a null cut callback

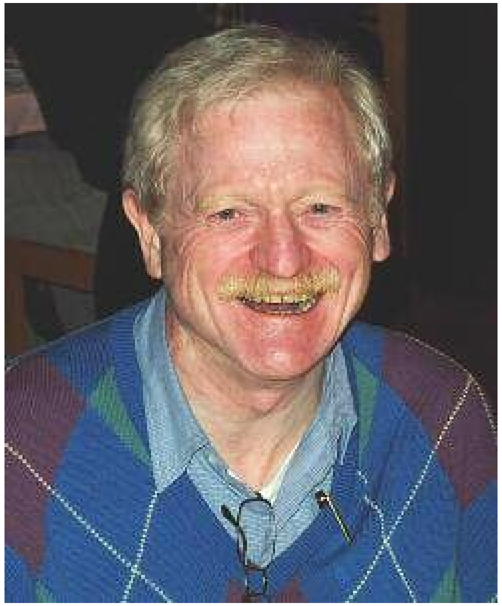
(null) cut callbacks force CPLEX to disable some preprocessing

Alternative CPLEX 8.1 configurations

	Solved	**Speedup**	**Solved by neither**	**Solved only by reference**	**Solved only by current**
Default w/out callback	19	0.89	16	4	0
Default w/out cuts	19	1.10	18	3	0
Null callbk w/out cuts	16	1.42	20	1	2

But this is Cplex alone; what about 0-1/2 cuts embedded into Cplex?

Much worse than Cplex alone, I presume !!



0-1/2 CUTS

- **Skip_step:** Cut separation applied at each **skip_step** backtrackings (Cplex-like)
- **Static cuts:** Cuts are never removed from the LP (Cplex-like)
- **Internal Pool:** used to store the 0-1/2 generated in previous calls, so as to increase the degree of freedom in the choice of the (few) cuts to add statically to the LP
- **Cut recombination**: allow for high-rank 0-1/2 cuts

0-1/2 CUT: Initial configurations

Skip_step=1, no other criteria, 5x or 0.3x n. of row in the original LP

	Solved	**Speedup**	**Solved by neither**	**Solved only by Cplex 8.1**	**Solved only by 0-1/2 CUT**
Lots of cuts (5x)	19	0.35	14	5	8
Few cuts (0.3 x)	18	1.07	17	1	5

Fixing max_angle= 0.3 and min_dist = 0.2

Tuning skip_step

skip_step = *, max_angle = 0.3, min_dist = 0.2

Skip_step	Solved	Speedup	Solved by neither	Solved only by Cplex 8.1	Solved only by 0-1/2 CUT
1	16	2.76	4	4	6
4	15	3.80	8	5	9
5	17	3.42	6	4	9
7	13	2.35	9	6	12
10	16	3.03	8	2	8
13	17	2.37	7	4	9
16	15	3.31	6	6	9

Tuning max_angle (=0 only orthogonal cuts, =1 all cuts)

skip_step = 4, max_angle= *, min_dist = 0.2

Max_angle	Solved	Speedup	Solved by neither	Solved only by Cplex 8.1	Solved only by 0-1/2 CUT
0.001	14	3.71	16	4	9
0.1	13	3.91	16	5	9
0.3	15	3.80	8	5	9
0.3	13	3.14	13	5	12
0.5	15	3.05	15	2	10
0.7	16	2.34	15	2	10
0.9	14	3.16	18	4	7
1.0	14	3.67	18	4	8

Tuning min_dist

skip_step = 4, max_angle= 0.1, min_dist = *

Min_dist	**Solved**	**Speedup**	**Solved by neither**	**Solved only by Cplex 8.1**	**Solved only by 0-1/2 CUT**
0.00	15	2.59	18	3	8
0.05	14	3.35	15	5	11
0.10	14	2.84	14	6	12
0.15	14	3.35	15	5	11
0.20	13	3.91	16	5	9
0.25	13	3.99	15	5	11

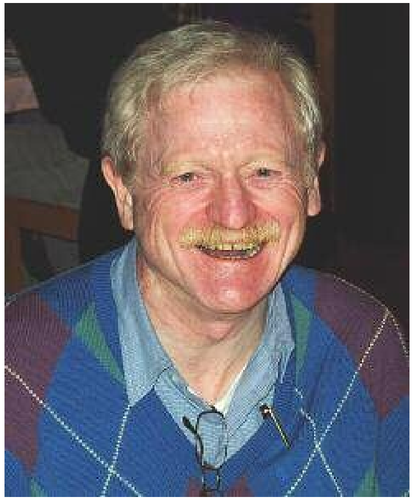
Conclusions

On specific ILP instances with a strong combinatorial structure, the use of 0-1/2 cuts can result into a considerable speed-up

STONGLY RECOMMENDED FOR HARD ILP'S

Any comment?

I hope they will not expect I include this $%£@# in my next book!!



I hope they will not expect I include this $%£@# in the new Cplex release!!