Lift-and-Project cuts: an efficient solution method for mixed-integer programs

Sebastian Ceria

Graduate School of Business and

Computational Optimization Research Center

http://www.columbia.edu/~sc244

Columbia University

Mixed Integer Programs

- A ***Mixed* 0-1 *Program***

$$\min cx$$
$$Ax \geq b$$
$$x \geq 0$$
$$x_i \in \{0,1\}, i = 1, \ldots, p$$

- its *LP Relaxation*

$$\min cx$$
$$\tilde{A}x \geq \tilde{b}$$

where $\tilde{A}x \geq \tilde{b}$ includes all bounds

- with *optimal solution* $\bar{x}$

The Lift-and-Project method

- Developed jointly with Egon Balas and Gerard Cornuejols
  - "A lift-and-project method for mixed 0-1 programs", Math Prog
  - "Mixed-integer programming with Lift-and-Project in a branch-and-cut framework", Manag. Science
- For the last two years joint work with Gabor Pataki
- Recent computational experiments with branch-and-cut and multiple cuts with Pasquale Avella and Fabrizio Rossi.
- Based on the work of Balas on Disjunctive Programming
- Related to the work of Lovasz and Schijver (Matrix cuts), Sherali and Adams, Merhotra and Stubbs, Hooker and Osorio, Merhotra and Owen.

Lift-and-Project cuts

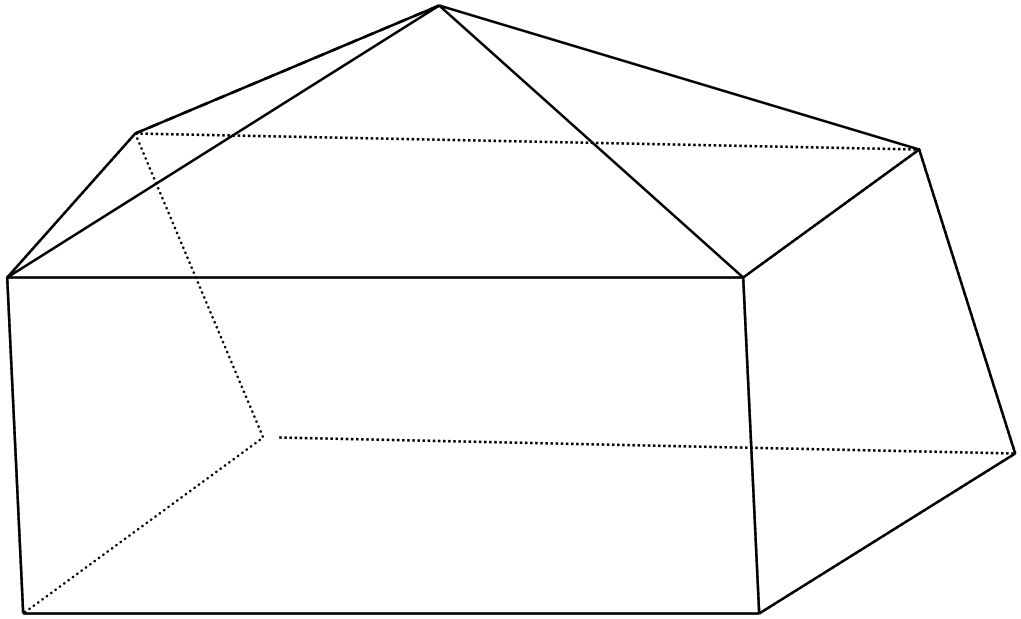
- Generate **cutting planes** for any mixed 0-1 program:
  - **Disjunction**

$$\begin{pmatrix} \tilde{A}x \geq \tilde{b} \\ x_i = 0 \end{pmatrix} \vee \begin{pmatrix} \tilde{A}x \geq \tilde{b} \\ x_i = 1 \end{pmatrix} \qquad \bar{x}_i \notin \{0,1\}$$

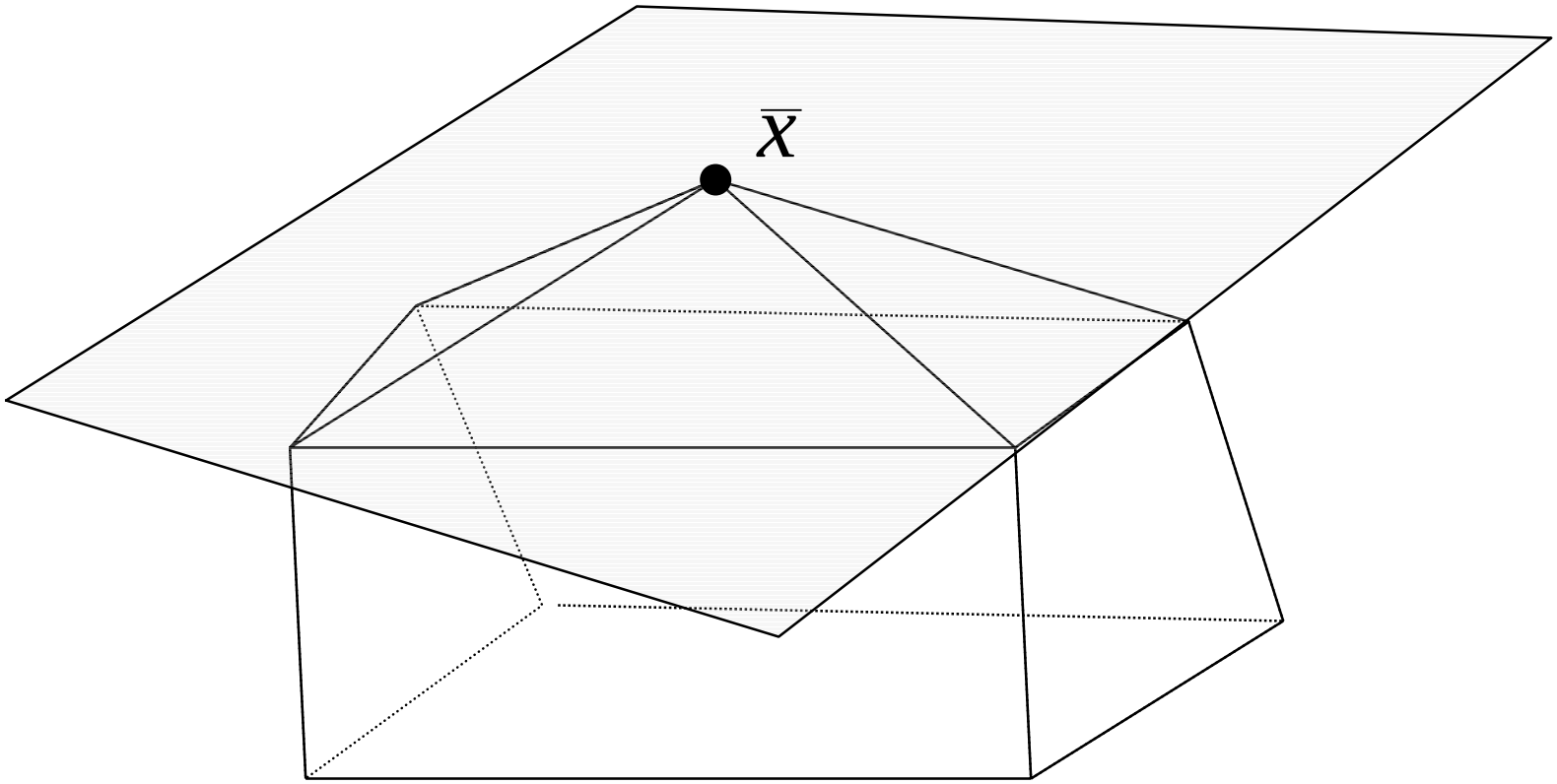
  - Description of

$$P_i = conv\left(\left(\begin{pmatrix} \tilde{A}x \geq \tilde{b} \\ x_i = 0 \end{pmatrix} \cup \begin{pmatrix} \tilde{A}x \geq \tilde{b} \\ x_i = 1 \end{pmatrix}\right)\right)$$

  - Choose a set of inequalities valid for P_i that ***cut off*** $\bar{x}$



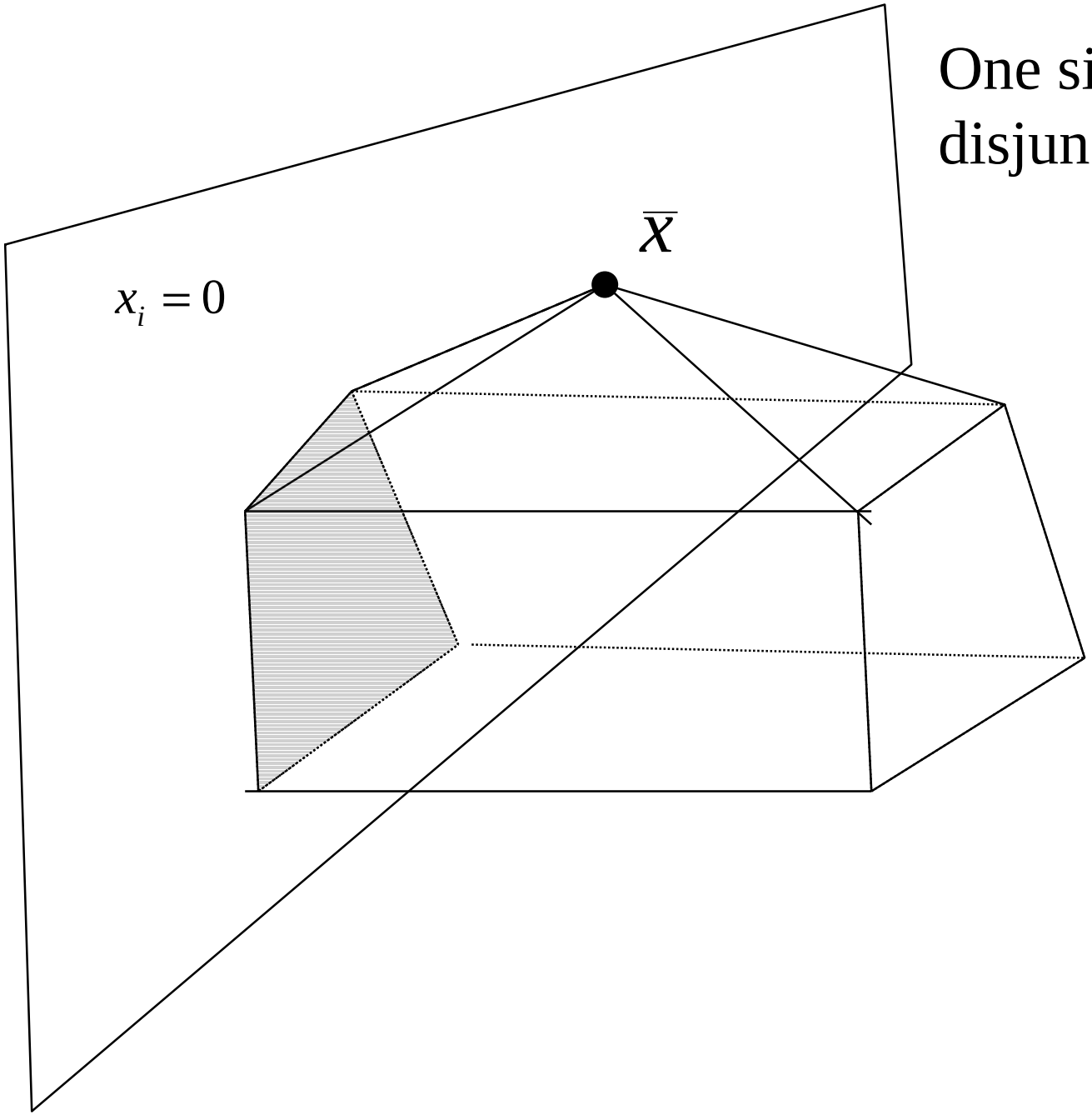
The LP relaxation



The optimal “fractional” solution

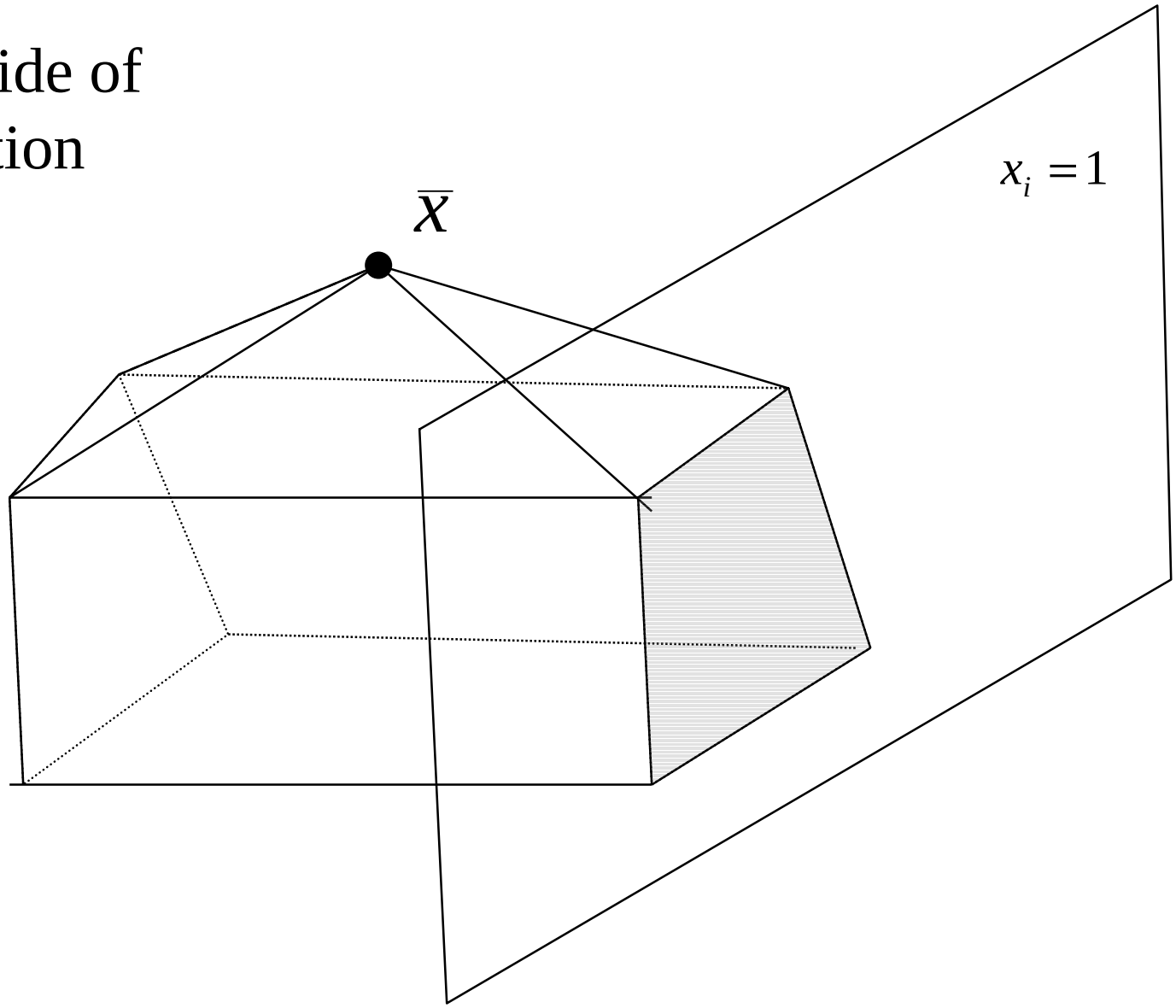
$$\begin{array}{ll} \tilde{A}x \geq \tilde{b} & \tilde{A}x \geq \tilde{b} \\ x_i = 0 & x_i = 1 \end{array}$$

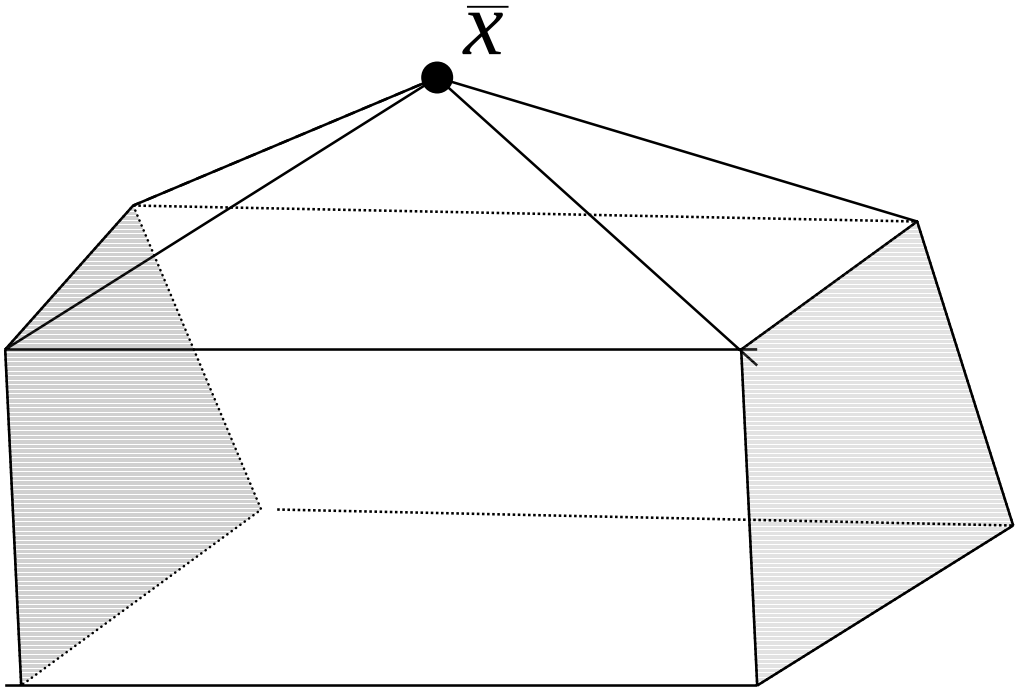
The disjunction



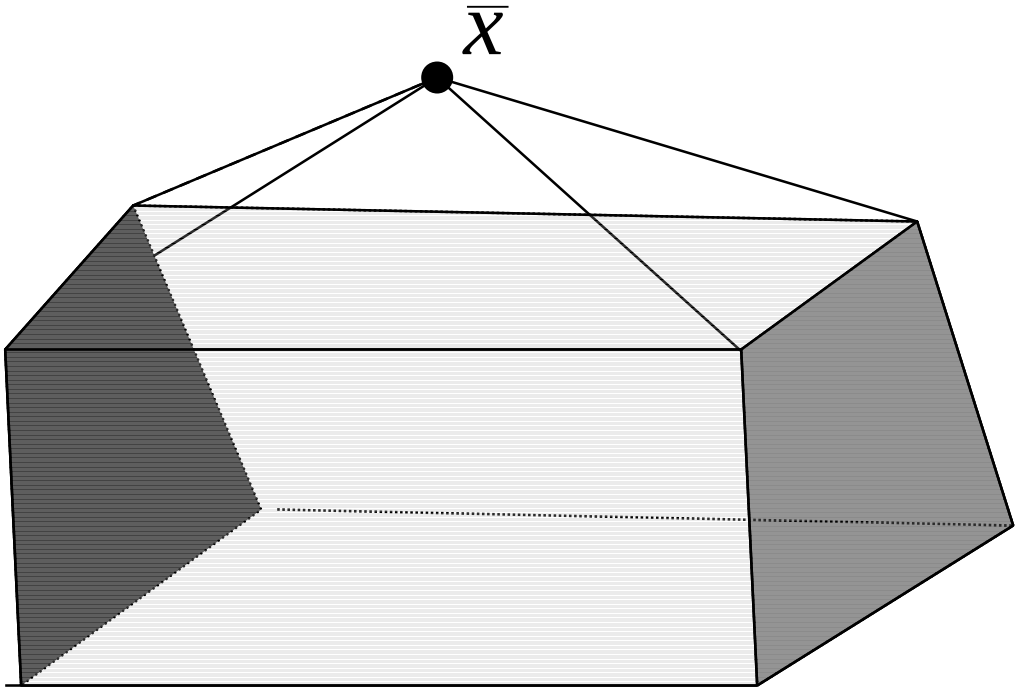
One side of the disjunction

The other side of the disjunction

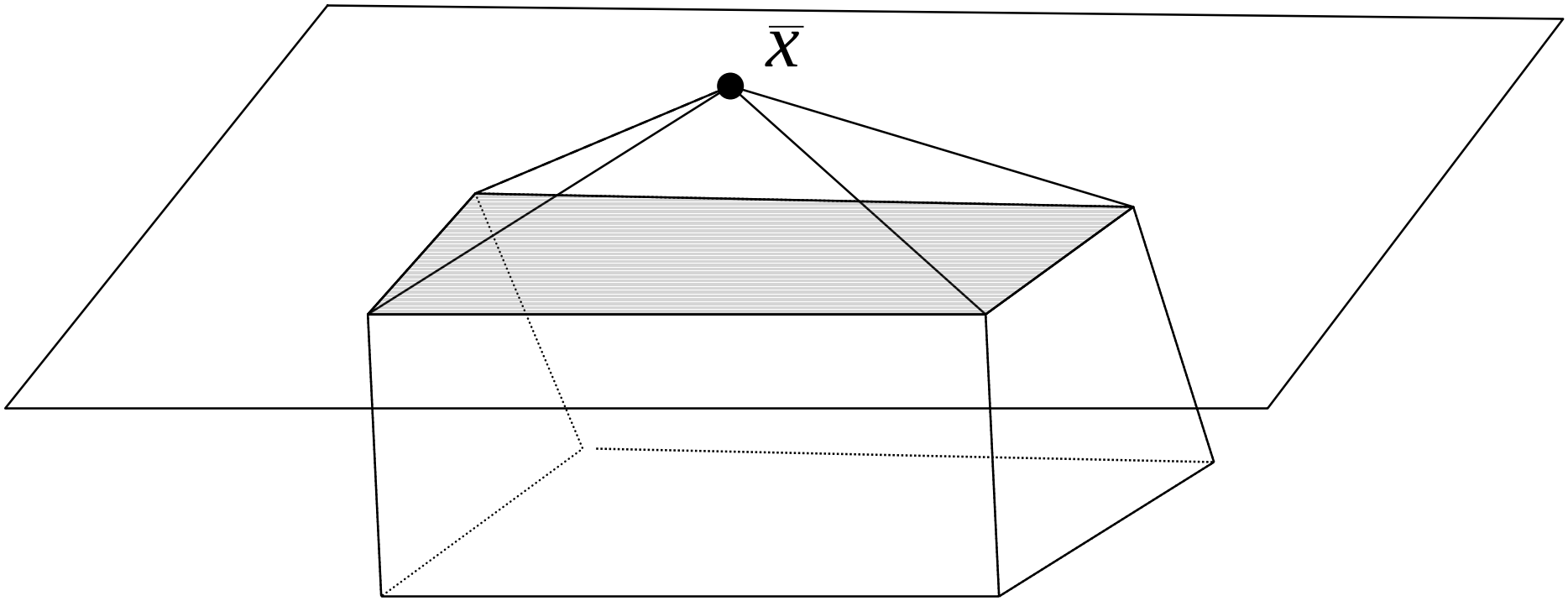




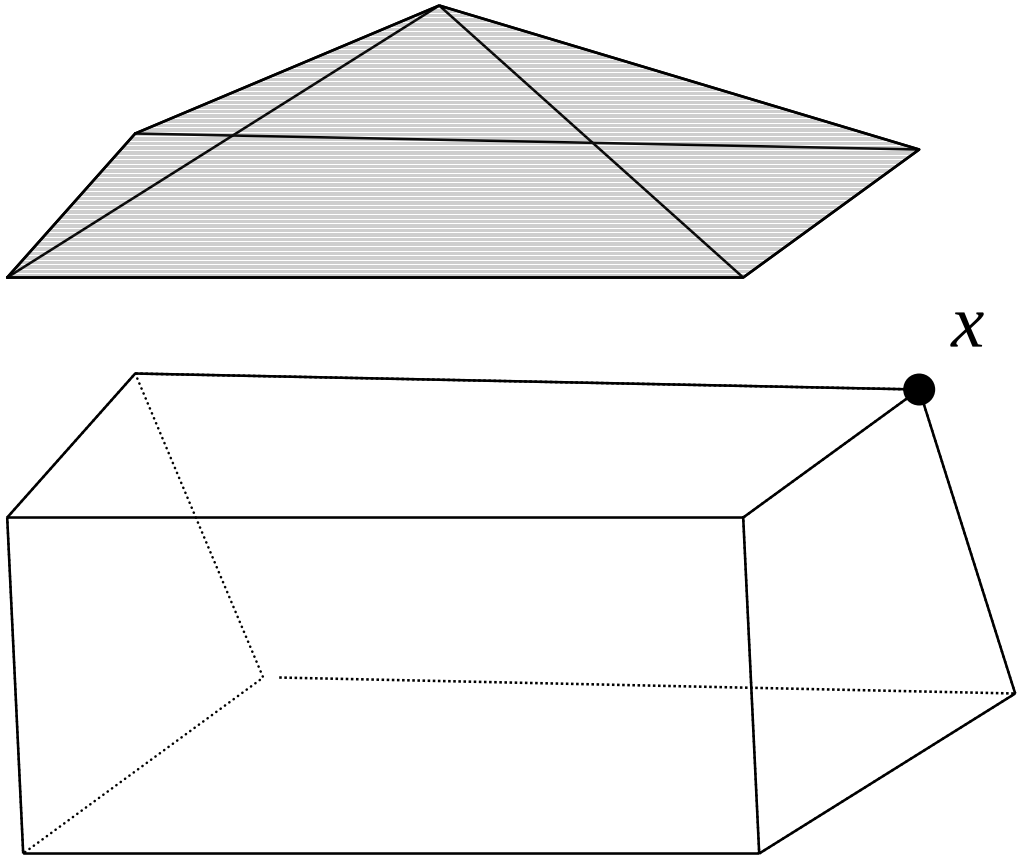
The union of the disjunctive sets



The convex-hull of the
union of the disjunctive sets



One facet of the convex-hull
but it is also a cut!



The new “feasible” solution!

How do we get disjunctive cuts in practice?

- A cut $\alpha x \geq \beta$ is valid for P_i *if and only if* $(\alpha, \beta) \in P_i(K)$, i.e. if it satisfies (Balas, 1979)

$$\alpha = u^1 \tilde{A} + v^1 e_i \qquad \alpha = u^2 \tilde{A} + v^2 e_i$$

$$\beta \leq u^1 \tilde{b} + v^1 0 \qquad \beta \leq u^2 \tilde{b} + v^2$$

$$u^1, u^2 \geq 0$$

- Hence, we have a **linear description** of the inequalities valid for P_i.

The Disjunctive Programming Approach

- A theorem of Balas allows us to represent the convex-hull on a higher dimensional space. It's projection is used to generate the cuts.

$$\tilde{A}x^1 \geq \tilde{b}\,x_0^1 \qquad \tilde{A}x^2 \geq \tilde{b}\,x_0^2$$
$$x_i^1 = 0 \qquad x_i^2 = x_0^2$$

$$x^1 + x^2 = x$$
$$x_0^1 + x_0^2 = 1$$

Normalization constraints, objective function and geometric interpretation

- We generate a cutting plane by:

  i) Requiring that inequality **be valid**, i.e. $(\alpha, \beta) \in P_i(K)$;

  ii) Requiring that it **cuts-off** the current fractional point

$$\max \beta - \alpha \bar{x}$$

$$\alpha = u^1 \tilde{A} + v^1 e_i \qquad \alpha = u^2 \tilde{A} + v^2 e_i$$

$$\beta \leq u^1 \tilde{b} + v^1 0 \qquad \beta \leq u^2 \tilde{b} + v^2$$

$$u^1, u^2 \geq 0$$

- This linear program is unbounded (the inequality can be **scaled** arbitrarily, and if there is a cut, the objective can be made $+\infty$)

Duality and geometric interpretation

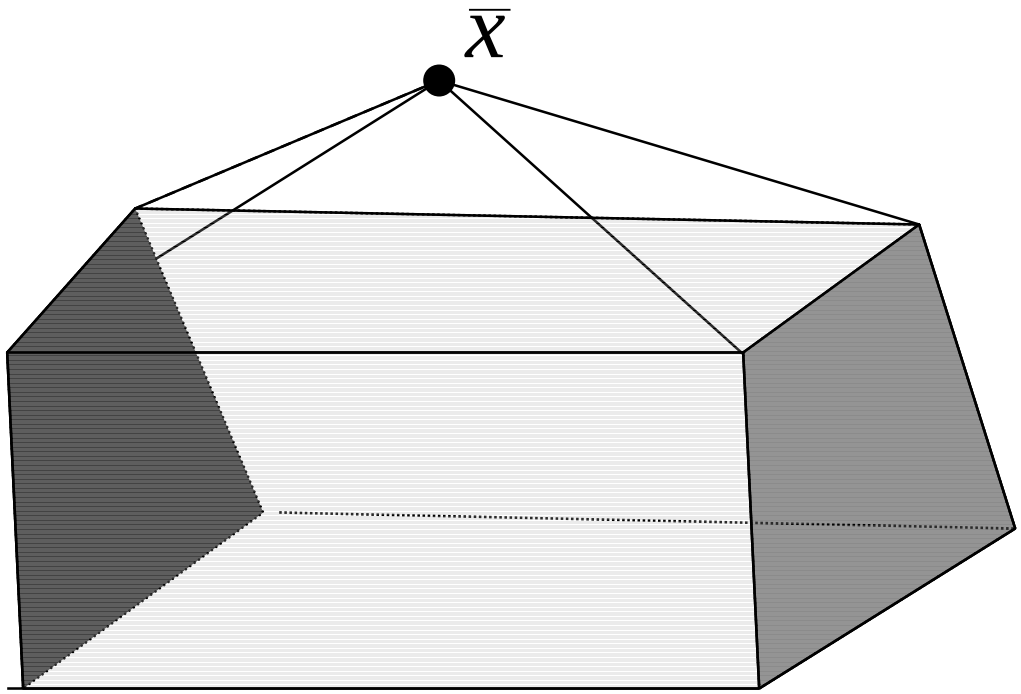
- The “primal” problem is **infeasible**...

$$\tilde{A}x^1 \geq \tilde{b}\,x_0^1 \qquad \tilde{A}x^2 \geq \tilde{b}\,x_0^2$$

$$x_j^1 = 0 \qquad x_j^2 = x_0^2$$

$$x^1 + x^2 = \tilde{x}$$

$$x_0^1 + x_0^2 = 1$$



$\bar{x}$ is not in the convex-hull of the
union of the sets obtained by fixing the variable to 0 and 1

Normalization constraints

- We add a **normalization constraint** for the cut generation linear program, which is equivalent to **relaxing** the "primal problem". We **limit the norm** of the cut, and **relax** the fact that the point needs to be in the convex-hull.

$$\max \beta - \alpha \bar{x} \qquad \min \left\| x^* - \tilde{x} \right\|_*$$

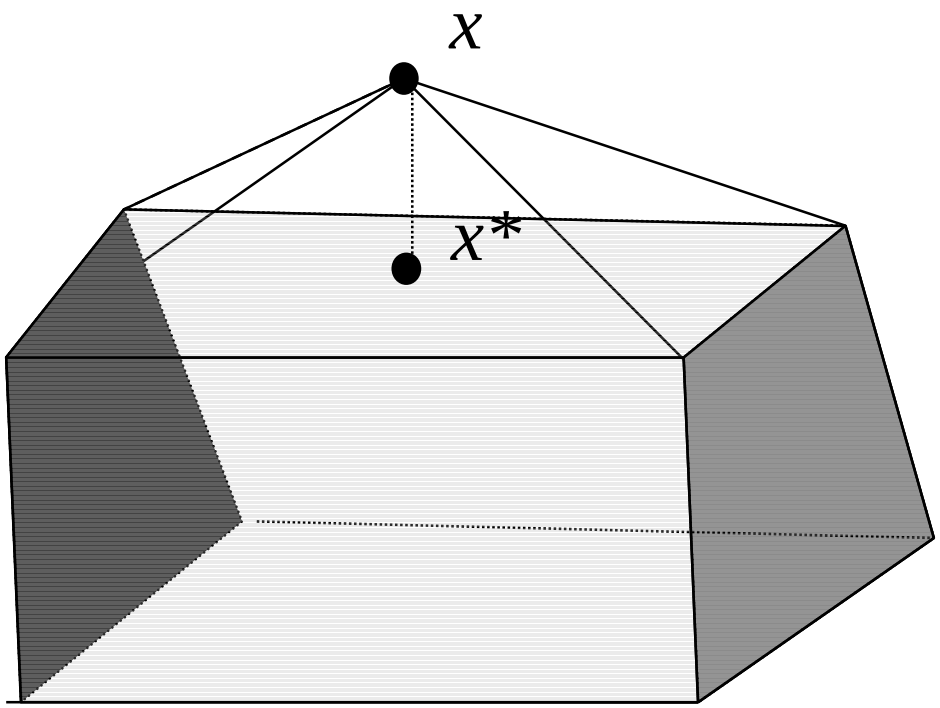
$$\alpha = u^1 \tilde{A} + v^1 e_i \qquad \alpha = u^2 \tilde{A} + v^2 e_i \qquad \tilde{A} x^1 \geq \tilde{b}\, x_0^1 \qquad \tilde{A} x^2 \geq \tilde{b}\, x_0^2$$

$$\beta \leq u^1 \tilde{b} + v^1 0 \qquad \beta \leq u^2 \tilde{b} + v^2 \qquad x_j^1 = 0 \qquad x_j^2 = x_0^2$$

$$u^1, u^2 \geq 0 \qquad x^1 + x^2 = x^*$$

$$\left\| \alpha \right\| \leq 1 \qquad x_0^1 + x_0^2 = 1$$

Geometric interpretation



Normalization constraints (cont)

- We normalize by **restricting the multipliers** in the cut generation linear program, which is equivalent to **relaxing the original constraints** in the "primal problem":

$$\max \beta - \alpha \bar{x}$$

$$\alpha = u^1 \tilde{A} + v^1 e_i \qquad \alpha = u^2 \tilde{A} + v^2 e_i$$

$$\beta \le u^1 \tilde{b} + v^1 0 \qquad \beta \le u^2 \tilde{b} + v^2$$

$$u^1, u^2 \ge 0$$

$$\sum_i u_i^1 + \sum_i u_i^2 + v^1 + v^2 \le k$$

$$\min t$$

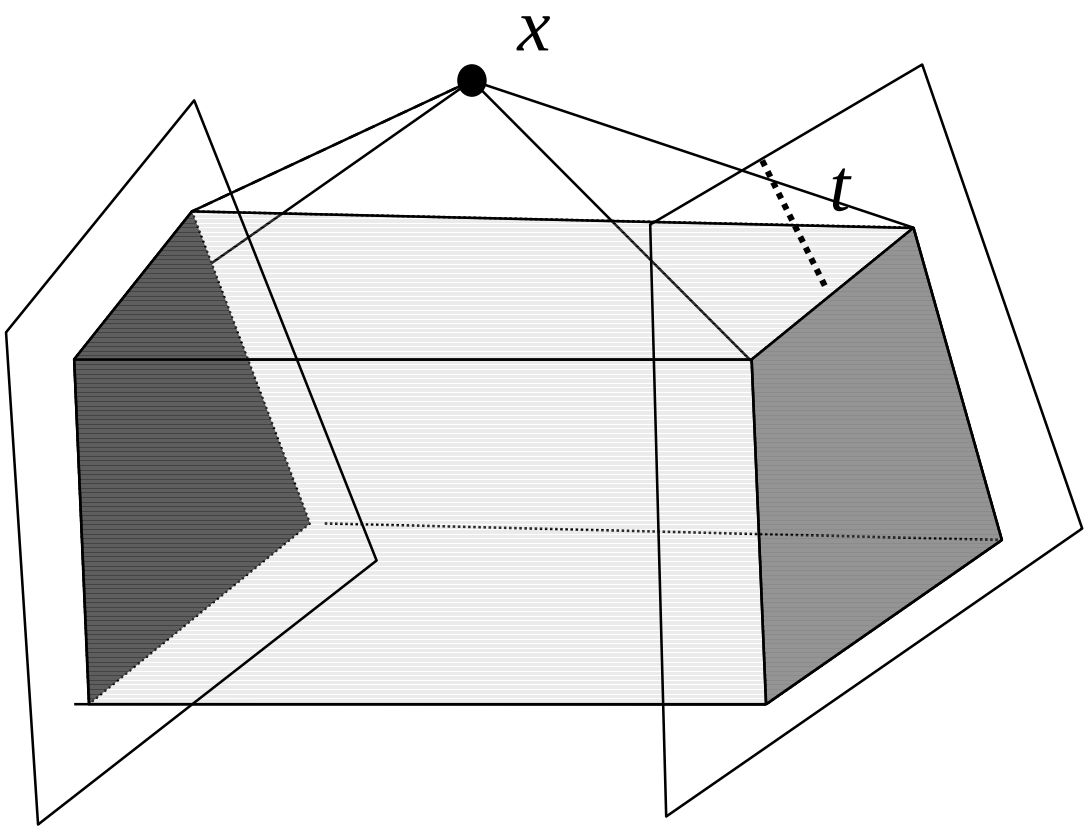
$$\tilde{A} x^1 + te \ge \tilde{b} x_0^1 \qquad \tilde{A} x^2 + te \ge \tilde{b} x_0^2$$

$$x_j^1 = 0 \qquad x_j^2 = x_0^2$$

$$x^1 + x^2 = \tilde{x}$$

$$x_0^1 + x_0^2 = 1$$

Geometrical interpretation



Cut generation LP

- A “look” at the matrix

u^1		v^1	u^2		v^2	α	β	
A^T	I	$-I$	e^i				$-I$		0
b^T		-1^T						-1	0
			A^T	I	$-I$	e^i	$-I$		0
			b^T		-1^T	1		-1	0

- Dimension of CLP:
  - rows: $2n+2$
  - columns: $2m+ 5n + 3$

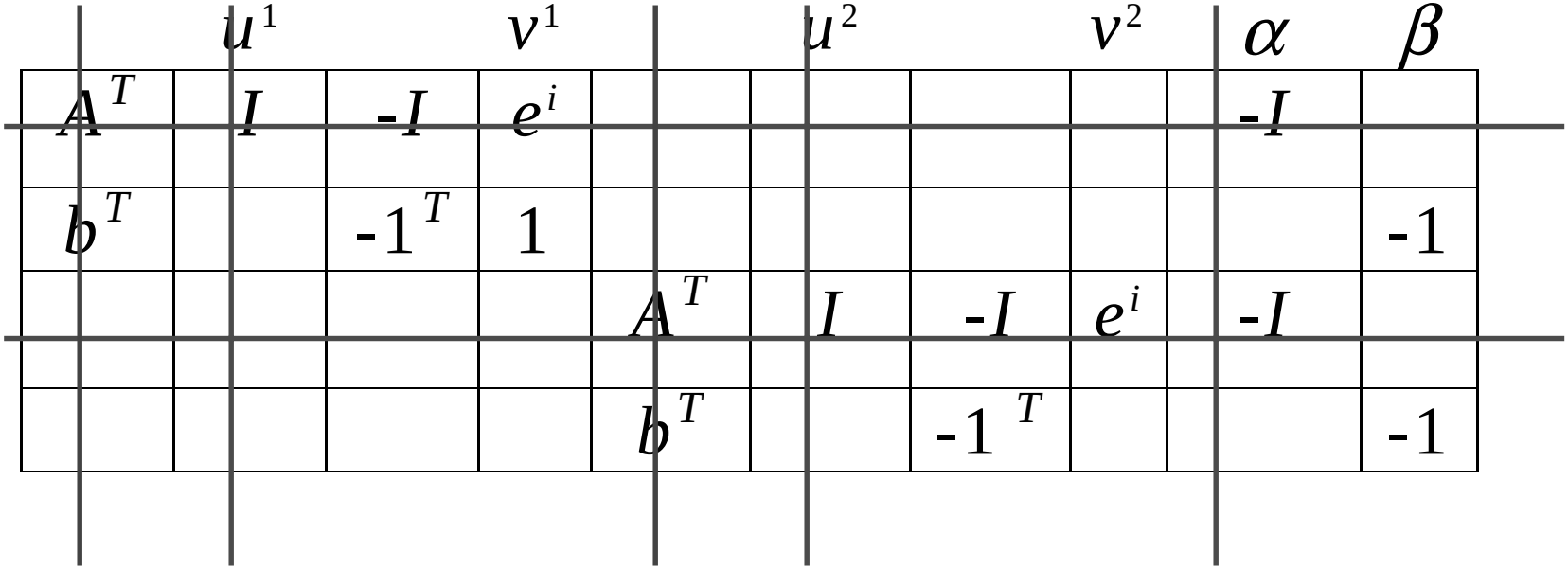
Solving the cut generation LP efficiently: Lifting

- Working on a **subspace of the "fractional" variables** (variables which are not at their bounds).
- It's a theorem about the **solution** of these linear programs, in order to solve the linear program on the full space of variables (more constraints and variables) we solve the problem in the subspace, and generate a solution to the full space by using a simple formula.
- In practice it **reduces** the computation time significantly (typically the number of variables between bounds is much smaller than the total number of variables.
- **It also allows for using the cuts in branch-and-cut**

Solving the cut generation LP efficiently: Constraint selection

- We can select to work with a **subset of the original constraints** (for example only the tight constraints).
- But if we only take the tight constraints, then we get the **intersection cut**! (Balas, Glover 70's).
- Since the intersection cut is readily available from the tableau, we could use the intersection cut as a starting basis for solving the cut generation linear program.
- In practice this combination **reduces the computation time significantly**, although it may yield weaker cuts.

How variable and constraint selection affect the cut generation problem



- Rows and columns deleted by (original) variable selection
- Columns deleted by constraint selection

Solving the cut generation LP efficiently: Multiple cut generation

- ***First idea***: let $(\alpha^*, \beta^*, u^*, v^*)$ be the first optimal solution of CLP
  - Choose a subset S of multipliers u^*
  - Add to CLP the constraint $u(S) = 0$
  - Reoptimize CLP
- A slight variation, do some pivoting so as to get alternative solutions to the cut generation linear program

Solving the cut generation LP efficiently: Multiple cut generation (cont)

- ***Second idea***:(**strong** cutting)
  - Generate other points x_1 ,…, x_k and also cut them off with the same cut-generation LP.
  - The fractional point only affects the objective function of the cut generation LP, so, in a way, it is like getting alternative solutions to the LP (these solutions may not cut-off the solution to the linear programming relaxation)
  - The other points are generated by doing some pivots in the original problem (pivots that give solutions that are close to optimal)
  - One could also think of generating an objective which is a convex combination of these (thus separating a convex combination of x_1 ,…, x_k)

More extensive computational results

- The test-bed
  - “All” problems from MIPLIB 3.0, excluding the ones that solve in less than 100 nodes with a good commercial solver using the consistently best setting (CPLEX 5.0, best bound, strong branching).
  - From here we chose the ones that take more than half an hour on a Sun Ultra 167 MHZ. There are 23 problems in this set. For these problems the strong branching setting works better than the default setting.

Cut and branch

- Generate **disjunctive** cuts from 0-1 disjunctions in **rounds** (a set of cuts generated for different disjunctions without resolving the relaxation), add them to the formulation. After every round, drop the non-tight cuts. This makes the LP-relaxation tighter, and the LP harder to solve.
- Run branch-and-bound (CPLEX 5.0, MIP solver SB-BB and XPRESS-MP, default parameters) on the **strengthened formulation**, and compare it with the run on the original formulation.

Cut and branch with 5 rounds of cuts

Problem	C&B Time	C&B Nodes	CPLEX 5.0 Time	CPLEX 5.0 Nodes
10 teams	5747	1034	5404	2265
Gesa2	1721	6464	9919	86522
gesa2_o	668	4739	12495	111264
Modglob	435	5623	+++	+++
p6000	805	1264	1115	2911
pp08a	178	1470	+++	+++
pp08aCUTS	134	607	50791	1517658
qiu	27691	15239	35290	27458
vpm2	974	18267	8138	481972

The cuts work but 5 rounds is too much

Problem	C&B Time	C&B Nodes	CPLEX 5.0 Time	CPLEX 5.0 Nodes
air04	5084	120	2401	146
air05	4099	213	1728	326
mod011	+++	+++	22344	18935

Rerunning these problems with only 2 rounds

Problem	C&B Time	C&B Nodes	CPLEX 5.0 Time	CPLEX 5.0 Nodes
Air04	1536	110	2401	146
Air05	1411	141	1728	326
Mod011	63481	24090	22344	18935

The rest of the problems

Problem	C&B Time	C&B Nodes	CPLEX 5.0 Time	CPLEX 5.0 Nodes
arki001	13642	12536	6994	21814
misc07	4133	14880	2950	15378
pk1	6960	150243	3903	130413
rout	40902	190531	19467	133075

Some comparisons

Problem	C&B(CP) Time	C&B (CP) Nodes	C&B (XP) Time	C&B (XP) Nodes
10 teams	5747	1034	1120	5983
gesa2	1721	6464	3248	145352
gesa2_o	668	4739	2024	95349
Misc07	4133	14880	997	32284
modglob	435	5623	2439	235106
p6000	805	1264	2123	12924
pp08a	178	1470	256	784
pp08aCUTS	134	607	116	9073
qiu	27691	15239	7016	47951
vpm2	974	16267	1413	136053

- Problems **not solved** with any of the two methods:
  - noswot, set1ch, harp2, seymour, dano3mip, danoint, fastxxx.
- Performance of our cuts on these problems:
  - Poor: noswot, harp2 (high symmetry) dano3mip, danoint (already contains many cuts)
  - Fairly good : set1ch, seymour.
- Not tried
  - fastxxx

Conclusions

- A **robust** mixed-integer programming solver that uses general disjunctive cutting planes.
- The cuts can be used within branch-and-cut with **very good** performance.
- The computational experiments indicate that it is very important to be able to generate **good "sets" of cuts**, rather than individual cuts.
- How do we use disjunctions more efficiently?
- Which are good disjunctions?
- Which are bad ones?

Branch & cut MISC07

3 rounds at the root node
Cuts every 10 nodes
1 round for nodes different from the root
LP Solver: XPRESS-MP 10

Alternative	Nodes	Time
2	22241	3543
3	21415	3286
4	20584	3020
5	16946	2584
6	15338	2610

Cut and branch takes 35426 nodes and 4028 seconds with two alternative solutions

Cut and branch pp08a

Alternative	2 rounds	5 rounds	Nodes	Time
1	6010	6835	25067	850
2	6010	6837	18111	640
3	6012	6869	16763	650
4	6078	6898	14324	630
5	6079	6901	7030	580
6	6165	6985	5020	540
7	6167	6985	4982	605