The Feasibility Pump

2.0

Matteo Fischetti - DEI, University of Padova
Domenico Salvagnin - DMPA, University of Padova

Aussois, January 2009

Primal Heuristics

- Crucial when exact methods don't work
- Crucial for the effectiveness of exact methods (=B&C) for mixed integer programming (MIP)
- Need a first feasible solution to enable:
  - bounding!
  - reduced cost fixing
  - other improving heuristics (RINS, LB, etc...)

The Problem

We want to find a feasible solution to:

$$\min \{c^T x : Ax \geq b,\ x_j \text{ integer for } j \in J\}$$

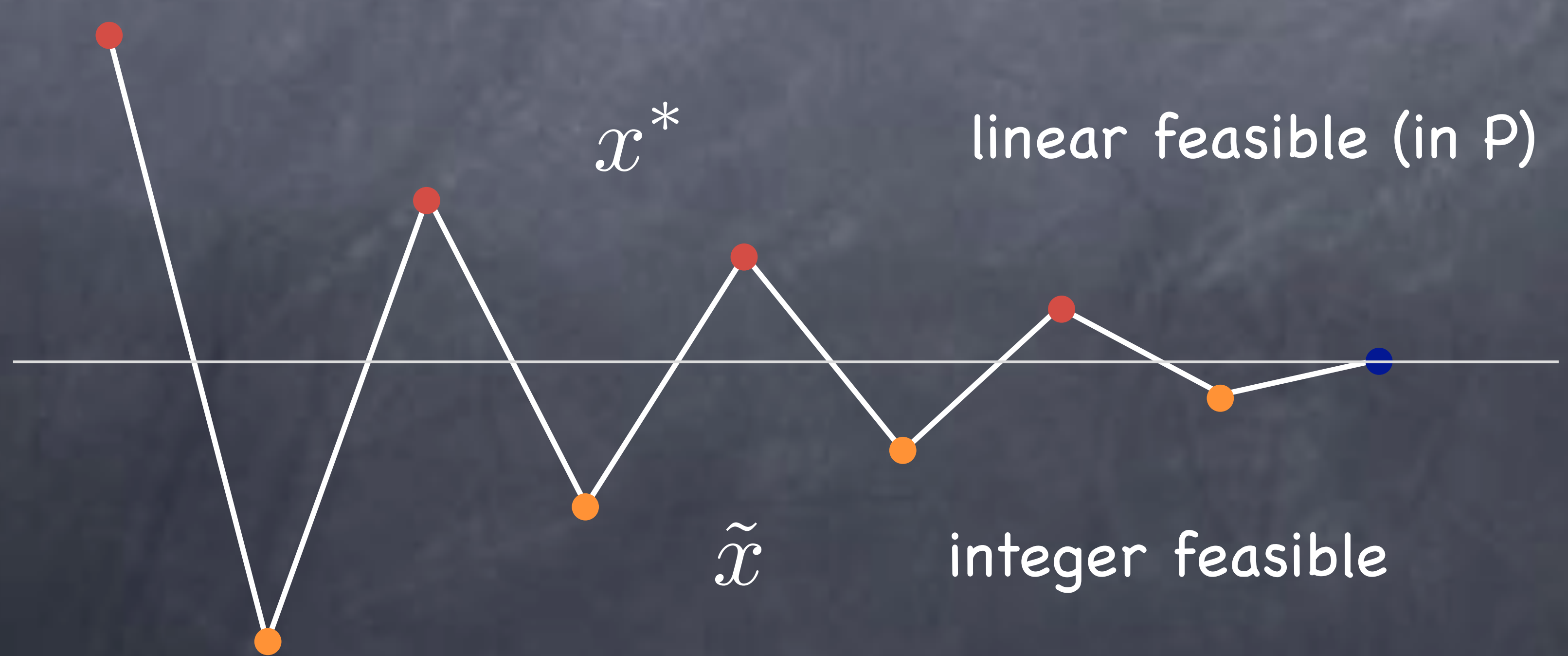
Notation:

- polyhedron $P = \{x : Ax \geq b\}$
- index set of integer variables J

The Feasibility Pump 1.x

Fischetti, Glover, Lodi [2003]
Bertacco, Fischetti, Lodi [2005]
Achterberg, Berthold [2007]

- recent primal heuristic for MIPs
- generates two (hopefully convergent) trajectories of points x^* and $\tilde{x}$ that satisfy feasibility in a complementary way



Basic Scheme I

How to get an integral point from a fractional one?

Plain rounding of the components in J to the nearest integer

$$\tilde{x}_j = \begin{cases} [x_j^*] & j \in J \\ x_j^* & j \notin J \end{cases}$$

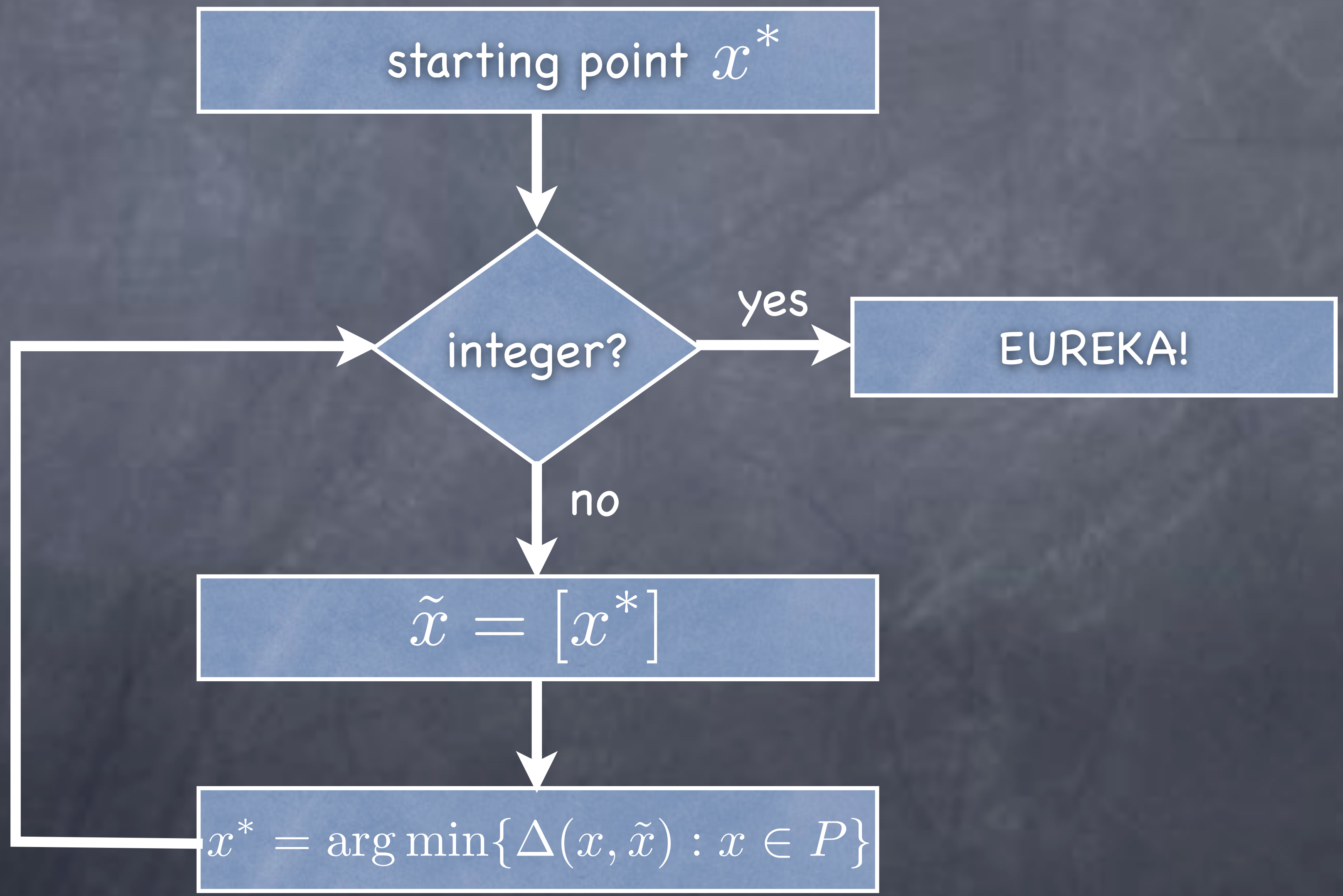
How to get an LP-feasible point from an integral one?

Find the point in P closest w.r.t the L_1 norm Δ

$$x^* = \arg\min\{\Delta(x, \tilde{x}) : x \in P\}$$

Basic Scheme II

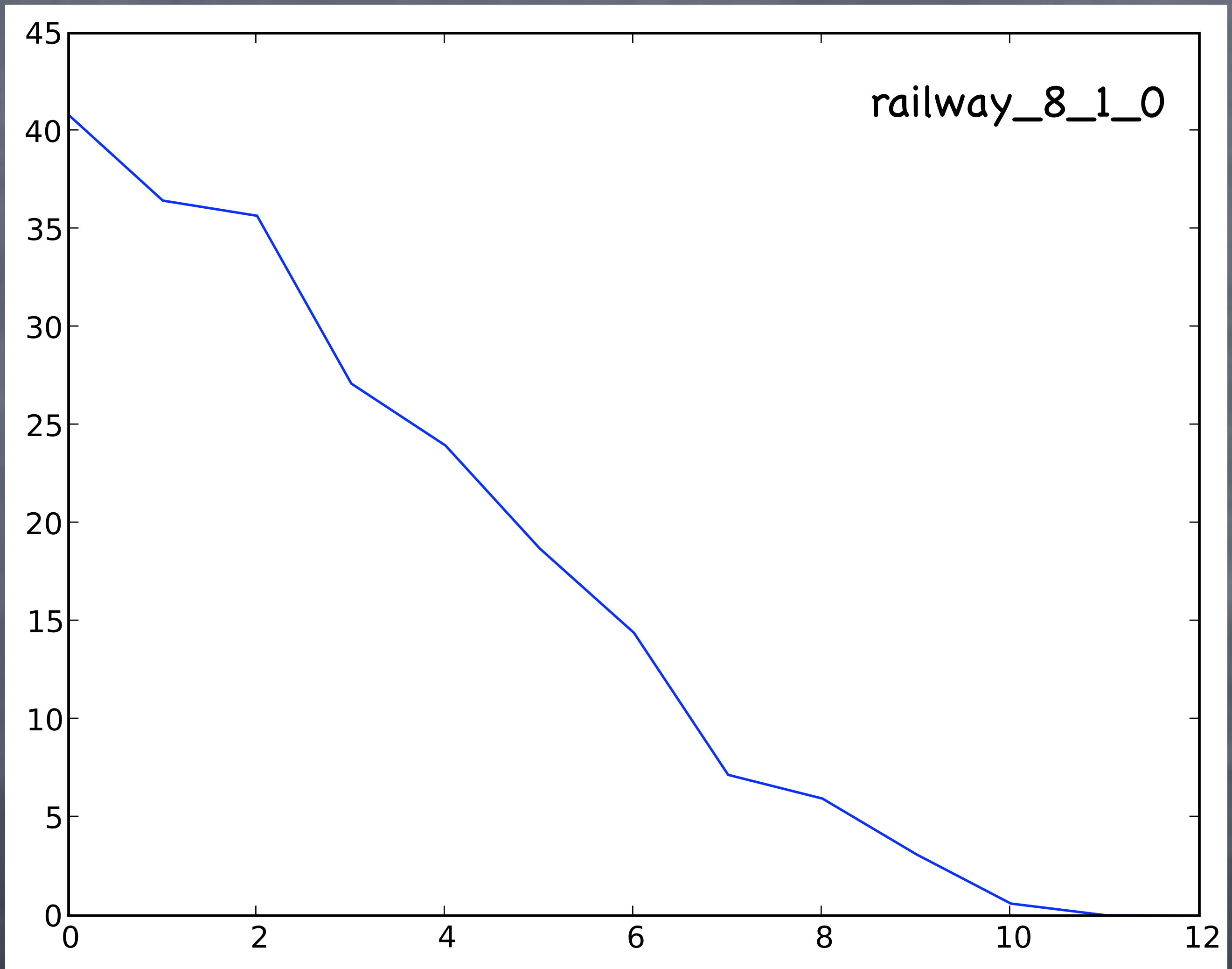
A simplified algorithm may look like this:



Cycling

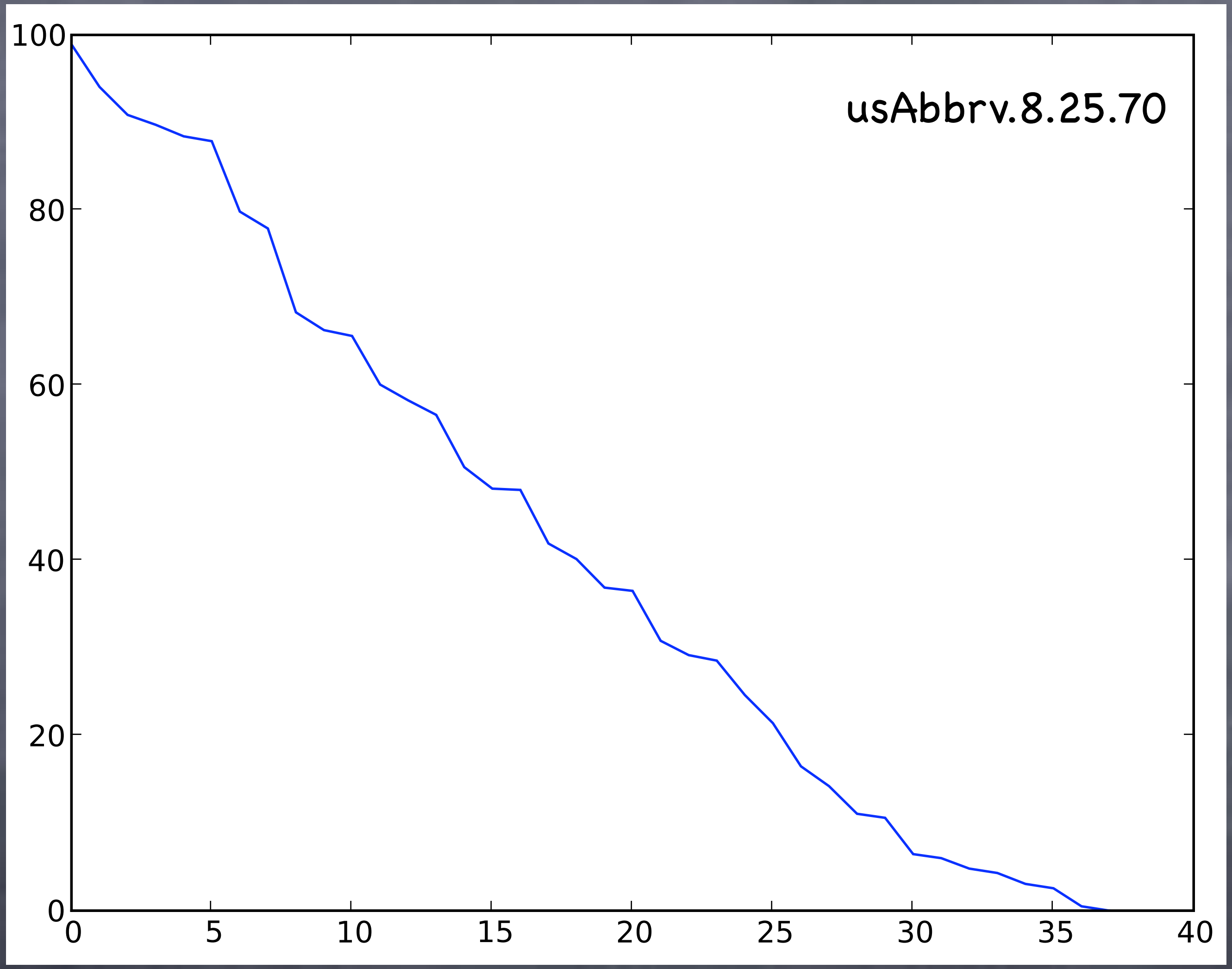
- What happens if we generate an already discovered integer solution?
- We are in a cycle!!!
- Need some anticycling mechanisms:
  - weak random perturbations
  - strong kicks (random restarts)

Real Examples



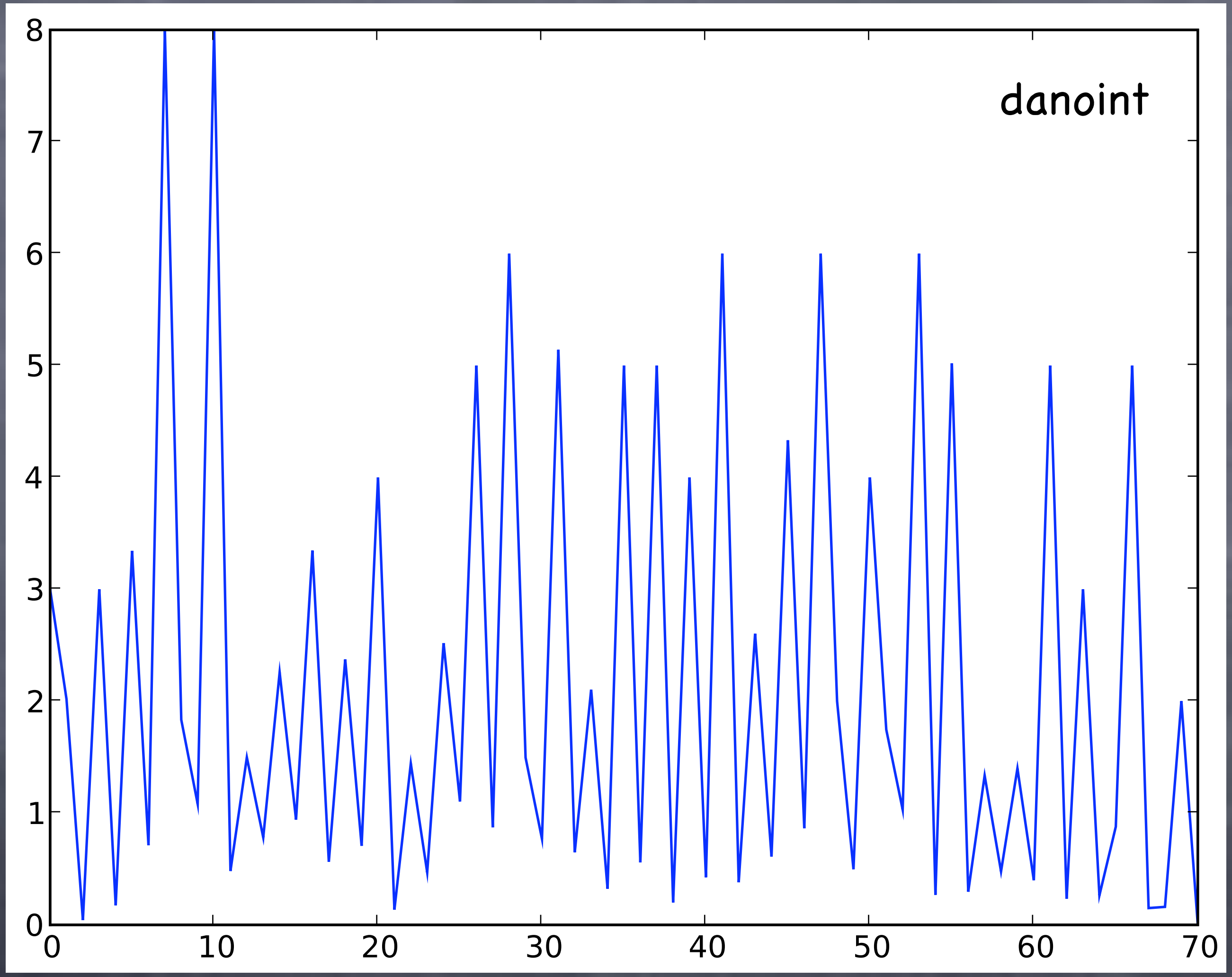
Good

Real Examples



Fair

Real Examples



Bad!

Rounding: are we serious?

Many advantages:

+ extremely simple and fast
+ [x] is the nearest integer point to x
+ convergence **IN ABSENCE** of cycles

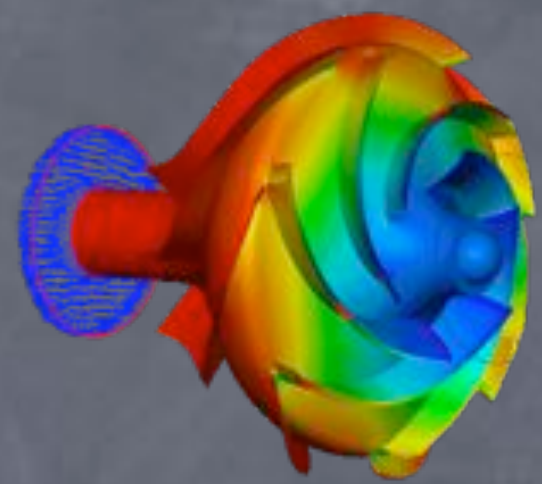
BUT:

- prone to cycling (many different continuous x may map to same integral [x])
- completely forgets P

FP 1.x: quick summary

- simple primal heuristics
- convergent in absence of cycles
- cycles are a big problem
- often feasible solutions are found because of frequent perturbations rather than by design
- simple rounding is quite blind and may fail on trivial instances (e.g. knapsacks, set covering)

FP 2.0: Inference!



- Rounding a variable means fixing it temporarily to a value
- Propagate the rounding of an integer variable before rounding the remaining ones
- Advantages:
  - use information from the linear constraints
  - hopefully still fast
  - akin to diving, but without solving LPs

Bound Strengthening

Savelsbergh [1994] Maros [2003] Hooker [2006] Achterberg [2007]

Given a linear constraint with positive coefficients:

$$LB \leq \sum a_j x_j \leq UB$$

and original bounds $[l_j, u_j]$ on the variables, we can compute the minimum and maximum activities:

$$L_{\min} = \sum a_j l_j \qquad L_{\max} = \sum a_j u_j$$

and update variable bounds:

$$x_j \leq l_j + \frac{UB - L_{\min}}{a_j} \qquad x_j \geq u_j + \frac{LB - L_{\max}}{a_j}$$

(can be rounded for integer variables and
generalized to constraints with negative coefficients)

Constraint Propagation

Schulte [2000] Actherberg[2007]

- How do we organize constraint propagation?
- We use a propagator-based approach
- Basic scheme:

```
K: set of variables
Ci: constraint i
Ci ⊃ j: Ci involves variable j
Q: set of constraints to propagate

Q ← {Ci: i = 1,…,m}
while Q not empty:
   Ci ← pop(Q)
   K ← propagate(Ci)
   Q ← Q ∪ {Ci: Ci ⊃ j for some j in K}
```

Constraint Propagation II

- Actual implementation is more involved due to optimizations (incremental propagation and specific constraint structure exploitation)
- What's the complexity of all of this?
  - polynomial for pure binary problems
  - pseudo-polynomial for general integer
  - may not converge in a finite number of propagations for continuous variables!

NEED TO STOP PROPAGATIONS AT SOME POINT!

FP 2.0: some remarks...

- propagation can be time consuming, but is typically fast enough
- the final result depends on:
  - how we choose the next variable to round
  - the order of constraint propagators
- no dominance exists between the two versions of the FP, but FP 2.0 is strictly better on a single feasible shot

Results: testbed

- Testbed: 43 binary and 29 general integer instances from MIPLIB 2003
- Turned into feasibility problems by adding bounds on the optimal value, as a relative gap [10%, 100%, None] from the best known solution
- variables sorted by increasing fractionality before propagation
- 3 different seeds for random perturbation
- two iteration limits (IL): 20 and 250

Computational Results

Improvements w.r.t. FP 1.x

			Gap			
Instances	IL	Measure	None	100%	10%	Overall
binary	20	#found	15%	30%	19%	21%
		iterations	32%	27%	7%	19%
		time	5%	8%	3%	5%
	250	#found	7%	18%	26%	14%
		iterations	46%	44%	11%	25%
		time	31%	57%	10%	31%
general integers	20	#found	31%	100%	54%	53%
		iterations	26%	19%	6%	16%
		time	3%	9%	-5%	2%
	250	#found	14%	41%	42%	27%
		iterations	23%	23%	8%	17%
		time	4%	27%	23%	22%

Computational Results

Solution Quality

Testbed	IT	Gap	FP1 better	FP2 better	equal
binary	20	10%	4	8	31
		100%	5	19	19
		None	7	19	17
	250	10%	7	7	29
		100%	8	20	15
		None	9	21	13
general integers	20	10%	2	5	22
		100%	3	8	18
		None	8	12	9
	250	10%	2	6	21
		100%	5	11	13
		None	5	17	7
Overall	**-**	**-**	**65**	**153**	**214**

Conclusions

- Propagation can be very effective if embedded into the FP scheme (both for binary and general integer instances!)
  - reduced number of iterations
  - increased success rate and solution quality, in a comparable amount of time
- What's next?
  - exploit higher level modeling tools when available
  - turn attention to the LPs (maybe FP 3.0...)

Paper available (for Jack) at:
http://www.dei.unipd.it/~fisch/papers/