LOCAL BRANCHING

or

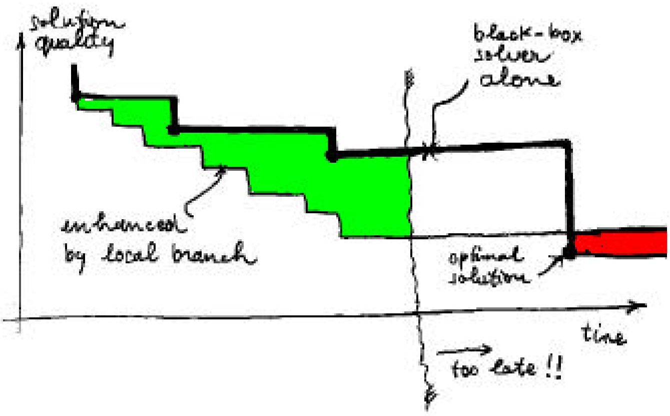
How to enhance the heuristic behaviour of your favourite 0-1 MIP solver

Matteo Fischetti

DEI, University of Padova, fisch@dei.unipd.it

Andrea Lodi

DEIS, University of Bologna, alodi@deis.unibo.it



Presented in AUSSOIS, January 6-11, 2002

0-1 Mixed-Integer Programs

We consider generic Mixed-Integer Linear Programming problems (MIP's) with 0-1 variables

$$\min \; c^T x$$
$$A x \geq b$$
$$x_j \in \{0,1\} \quad , \forall j \in \mathcal{B} \neq \emptyset$$
$$x_j \text{ integer} \quad , \forall j \in \mathcal{G} \; (\supseteq \emptyset)$$

Relevant cases:

- **0-1 ILP's** (generic or with a special structure)
  - set partitioning/covering models (crew scheduling etc.)
  - TSP, VRP , etc.

- **MIP's with no "general integer" variables**

- **MIP's with both general integer and binary variables**, the latter being often used to activate/deactivate costs/constraints (possibly using BIG-M tricks…)

Assumption: once the binary variables have been fixed, the problem becomes (relatively) easy to solve

Hard-to-solve 0-1 MIP's (in practice)

- In many practical cases, generic 0-1 MIP's can be solved in a satisfactory way by general-purpose commercial software which delivers:

    - Provably optimal solution
    - Heuristic solutions with a practically-acceptable error

Most MIPlib instances are of this type!

- Unfortunately, in other cases general-purpose software is not adequate and one has to:

    - Play with the MIP solver parameters ("emphasize integrality" etc.) so as to convince the $#$#?@# solver to deliver, at least, a good solution

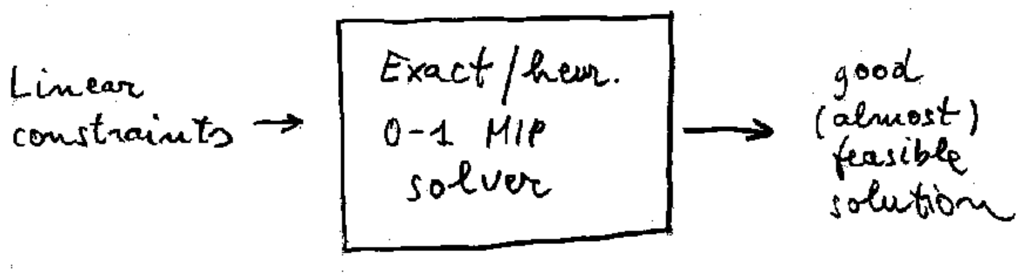
    - Design and use ad-hoc heuristics—thus loosing the advantage of working in a generic MIP framework

Many real-world instances are of this type!

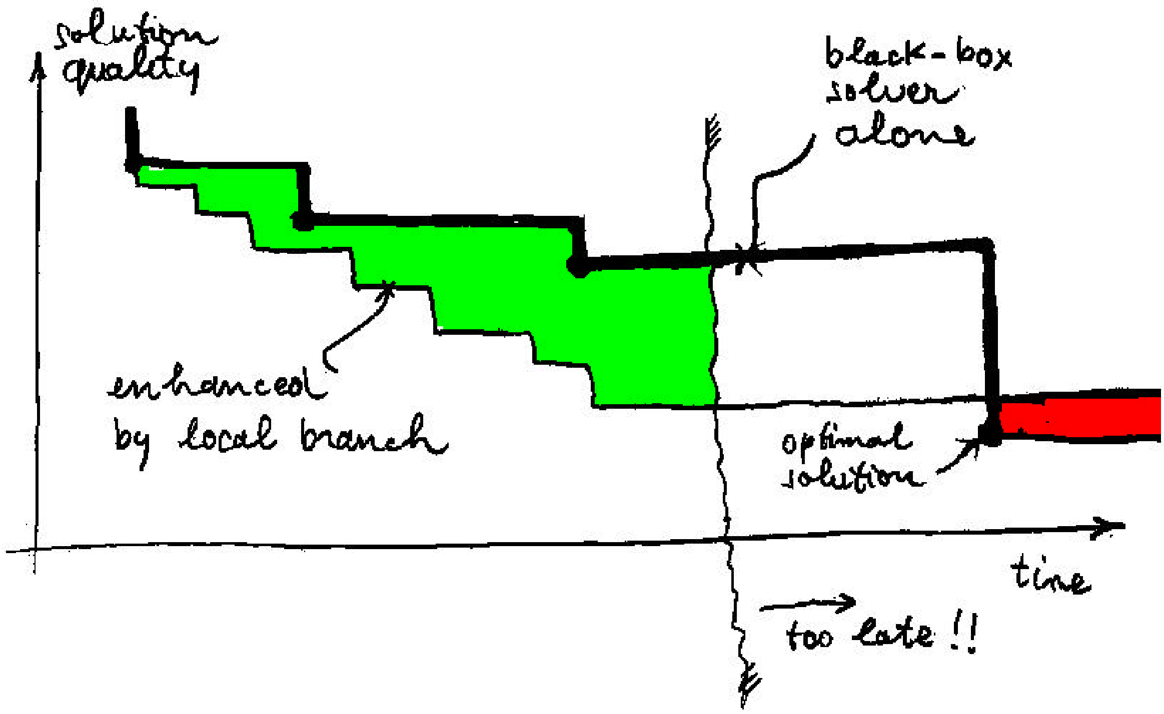
Better heuristics for general 0-1 MIP's strongly required!

A general heuristic framework

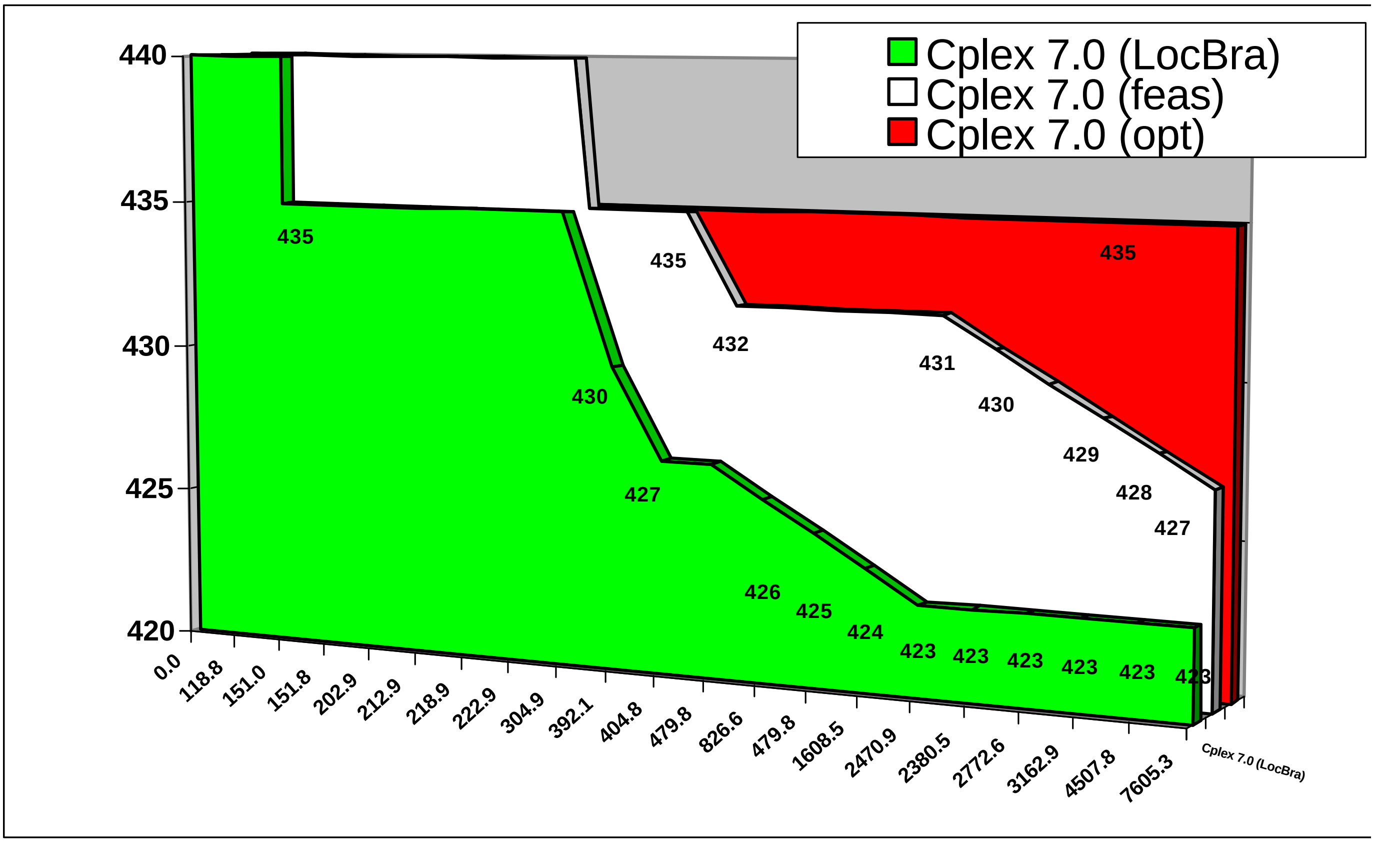
- We aim at embedding a **black-box** (general-purpose or specific) 0-1 MIP solver within an overall **heuristic framework** that "helps" the solver to deliver improved heuristic solutions

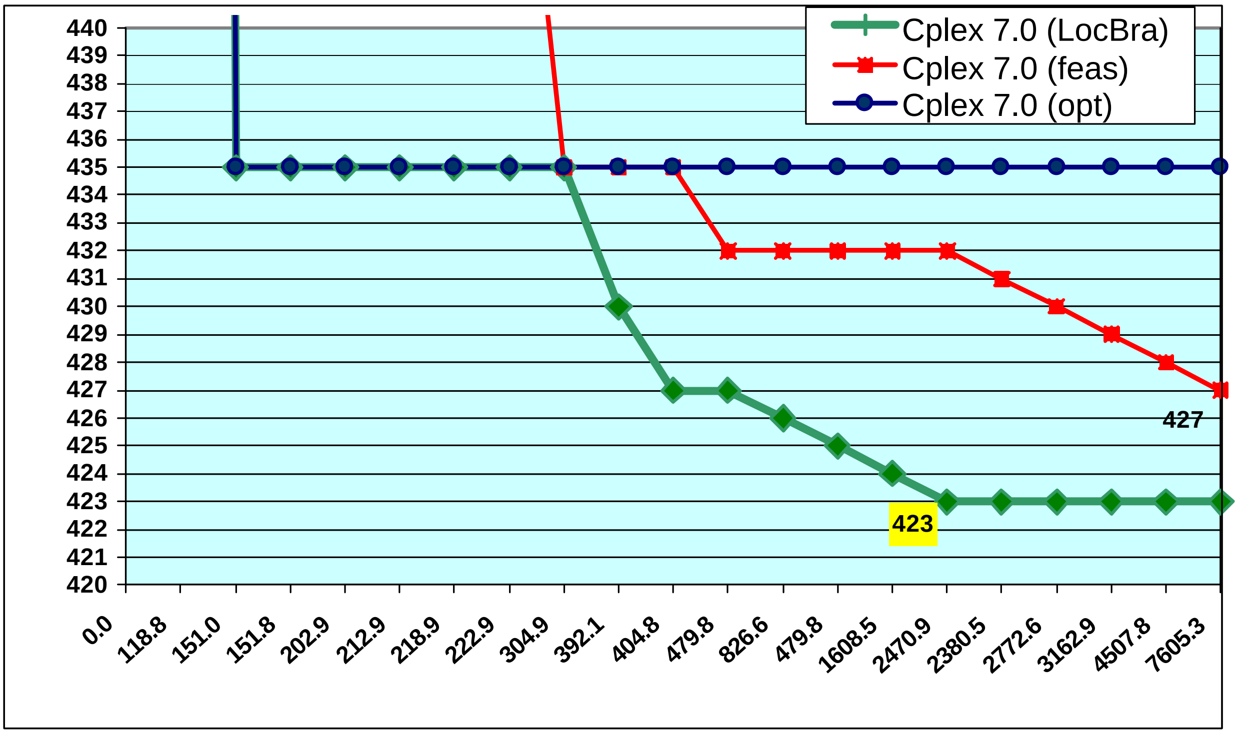


The available black-box module



The desired "Italian flag"





The Local Branch heuristic on a hard MIPLIP problem (seymour.lp)

MIPLIP problem seymour.lp

CPLEX 7.0: MIP emphasis: optimality

- Elapsed b&b time = 151.02 sec. : 435.0
- Final Sol. after 18000.0 sec.s : 435.0

CPLEX 7.0: MIP emphasis: integer feasibility

- Elapsed b&b time = 118.85 sec. : 459.0
- Elapsed b&b time = 202.98 sec. : 456.0
- 455.0
- 454.0
- Elapsed b&b time = 222.97 sec. : 453.0
- Elapsed b&b time = 304.97 sec. : 435.0
- Elapsed b&b time = 479.85 sec. : 432.0
- Elapsed b&b time = 2380.52 sec. : 431.0
- Elapsed b&b time = 2772.62 sec. : 430.0
- Elapsed b&b time = 3162.93 sec. : 429.0
- Elapsed b&b time = 4507.88 sec. : 428.0
- Elapsed b&b time = 7605.32 sec. : 427.0
- Final Sol. after 18000.0 sec.s : 427.0

CPLEX 7.0 & Local Branching (k=10)

- Local Branch Time = 151.8 sec. : 435.0
- Local Branch Time = 392.1 sec. : 430.0
- Local Branch Time = 404.8 sec. : 427.0
- Local Branch Time = 826.6 sec. : 426.0
- Local Branch Time = 1122.3 sec. : 425.0
- Local Branch Time = 1608.5 sec. : 424.0
- Local Branch Time = 2470.9 sec. : 423.0

MIPLIP: problem arki001.lp

CPLEX 7.0: MIP emphasis: optimality

- Elapsed b&b time = 21.12 sec. 7,594,629.2
- 7,590,295.2
- 7,590,247.2
- Elapsed b&b time = 212.32 sec. 7,585,194.4
- Elapsed b&b time = 1897.90 sec. 7,584,116.1
- Elapsed b&b time = 2088.58 sec. 7,583,895.3
- 7,583,878.4
- Elapsed b&b time = 2450.85 sec. 7,582,953.8
- Elapsed b&b time = 2613.20 sec. 7,582,840.6
- Elapsed b&b time = 4160.22 sec. 7,582,751.4
- Elapsed b&b time = 6216.88 sec. 7,582,634.8
- Elapsed b&b time = 7161.85 sec. 7,582,414.4
- Elapsed b&b time = 7161.85 sec. 7,582,302.6
- Elapsed b&b time = 14322.80 sec. 7,582,202.7
- Elapsed b&b time = 16237.02 sec. 7,582,031.3
- Elapsed b&b time = 16237.02 sec. 7,582,024.4

CPLEX 7.0 & Local Branching (k=10)

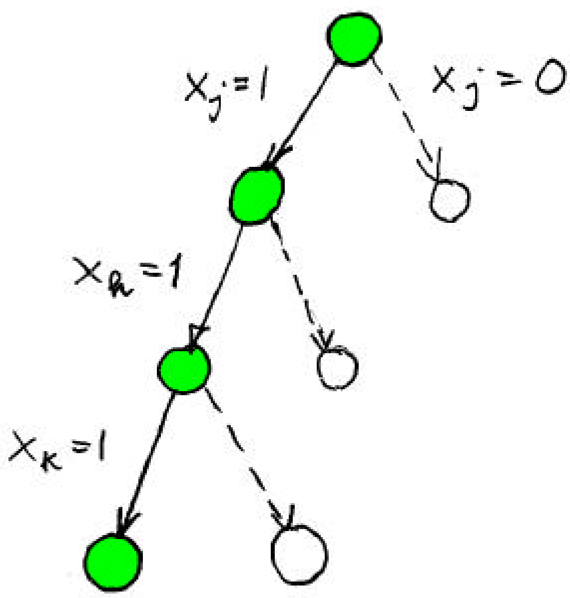
- Time = 21.6 best sol = 7,594,629.28
- Time = 471.9 best sol = 7,581,831.57
- Time = 922.0 best sol = 7,581,267.36
- Time = 2279.9 best sol = 7,580,980.56
- Time = 2730.1 best sol = 7,580,947.53
- Time = 3630.4 best sol = 7,580,937.90
- Time = 4080.5 best sol = 7,580,925.20

Variable-fixing strategy (hard version)

- A commonly-used (often quite effective) heuristic framework
- Let x^H be an (almost) feasible "target solution", and let

$$S = \{ j \in B : x_j^H = 1 \}$$

denote its **binary support** (binary var.s at value 1).



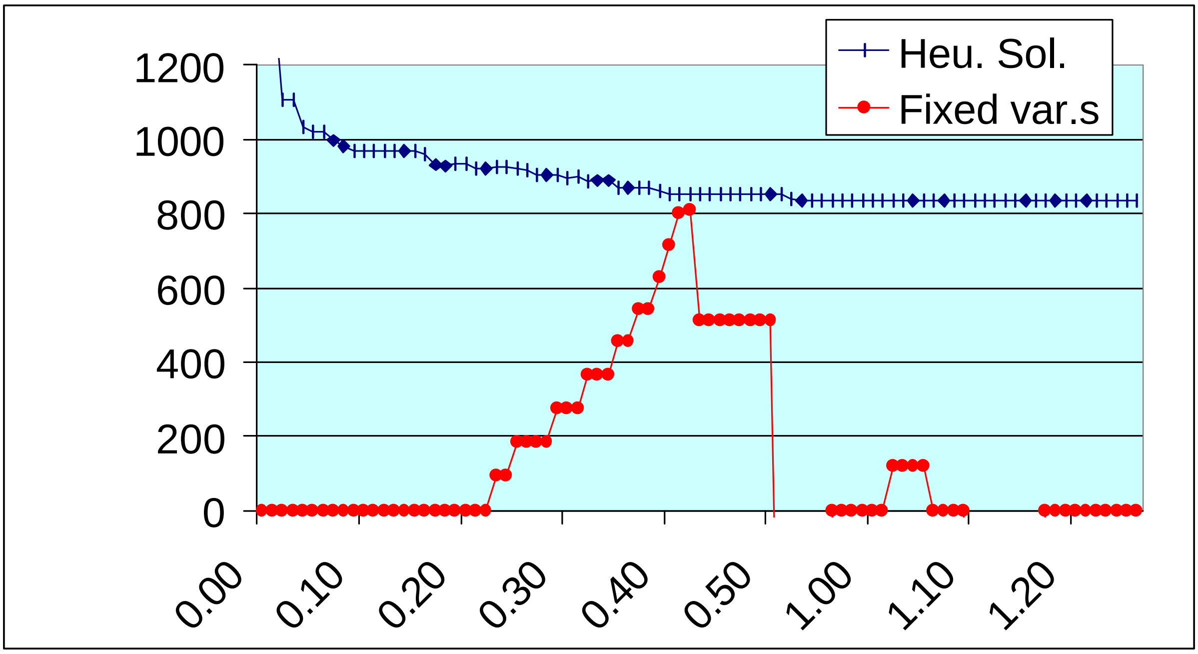
- **Heuristic depth-first search** of the branching tree:
  - iteratively **fix to 1** certain "highly efficient" variables in S (**green nodes**)
  - apply the **black-box module** to some **green nodes** only
  - only limited **backtracking** allowed

Advantages:

- Problem size quickly reduced: the black-box solver can concentrate on smaller and smaller "**core problems**"
- The black-box solver is applied over and over on different subproblems (**diversification**)

<u>Example:</u>

Black-box = Crew scheduling solver based on a set covering model (dynamic column-generation & Lagrangian heuristics)



Crew scheduling heuristic TURNI using variable fixing

<u>Disadvantages:</u>

- How to choose the "highly efficient variables in S" to be fixed?
- Wrong choices at early levels are typically very difficult to detect, even when **lower bounds** are computed along the way

Feasible solutions only available after several fixings, at a **deep level** in the branching tree

The lower bound does not help at early levels: it often stays quiet for several fixings, and **bumps** suddenly after an apparently innocent late fixing

<u>How to reach a sufficiently-deep branching level with a good lower bound?</u>

Variable-fixing strategy (soft version): local branching

- As before, let x^H be a (almost) feasible "target solution" and denote by $S^H = \{j \in B : x_j^H = 1\}$ its binary support
- Don't decide the actual variables in S^H to be fixed (a difficult task!), but just their **number** $|S^H| - k$
- Introduce the **local branching** constraint

$$x(S^H) := \sum_{j \in S^H} x_j \geq |S^H| - k$$

$$\left(\text{or} \quad \sum_{j \in S^H} (1 - x_j) \leq k \right)$$

so as to define a convenient **k-OPT neighbourhood** $N(x^H, k)$ of the target solution x^H (the larger k, the larger the neighbourhood)

"Akin to k-OPT for TSP"

- Search $N(x^H, k)$ by means of the black-box module
- Repeat with a different target solution (if available) and/or with a modified k (basic idea: to be elaborated in the sequel…)

> **Conjecture:** a small value of k drives the black-box solver towards integrality as effectively as fixing a large number of variables, but with a **much larger degree of freedom** → better solutions can be found.

Local branching in an exact solution framework

- **Black-box module** = generic **exact** branch-and-cut (or branch-and-bound) MIP solver , e.g. Cplex or XPRESS

- **General scheme (sketch)**

1. set $h := 0$ and choose a convenient value for parameter k
2. run the MIP solver (initial upper bound $= +\infty$) for a while, until a **first feasible solution** $x^{(h)}$ is found, and let

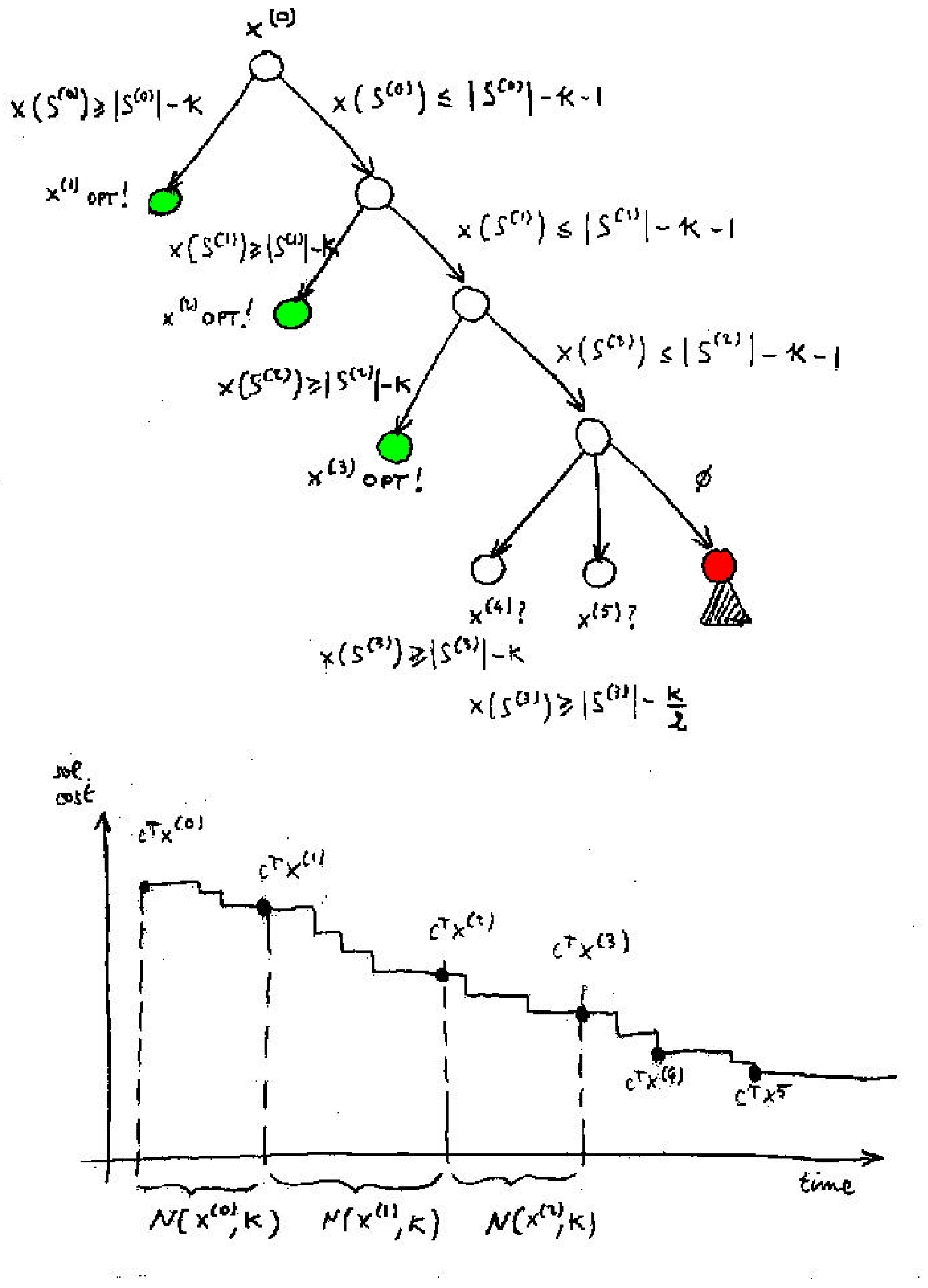
$$S^{(h)} := \{ j \in B : x_j^{(h)} = 1 \}$$

be its binary support

3. add the local branching constraint $x(S^{(h)}) \geq |S^{(h)}| - k$ to the current MIP, and try to solve it exactly within a certain time limit (initial upper bound = $c^T x^{(h)}$)
4. let $x^{(h+1)}$ be the best solution found so far
5. remove the last local branching constraint $x(S^{(h)}) \geq |S^{(h)}| - k$
6. **GREEN flag** : if **current problem solved to proven optimality or infeasible**, then

   add the "valid cut" $x(S^{(h)}) \leq |S^{(h)}| - k - 1$, set

   $h := h + 1$ and repeat from step 3 (increase k if pr. infeas.)
7. if you feel lucky, reduce k and repeat from step 3
8. **RED flag** : give-up and run the MIP solver on the current problem



Example: local branching in an exact solution framework

Local branching in a heuristic solution framework

- Easy adaptation of the previous framework: when the RED FLAG situation occurs, use a **diversification** mechanism to find a (worse) solution $x^{(h+1)}$ to replace the current-best solution, and continue.

- Diversification by **Variable Neighbourhood Search** (Hansen & Mladenovic, 1998):

Find a solution $x^{(h+1)}$ close enough to $x^{(h)}$, but outside the current k-OPT neighbourhood, e.g.

$$x^{(h+1)} \in N(x^{(h)}, k + k/2) \setminus N(x^{(h)}, k)$$

  - Run the MIP solver (initial upper bound $= +\infty$) to find the first feasible solution $x^{(h+1)}$ of the current problem amended by the **diversification constraint**

$$k + 1 \leq |S^{(h)}| - x(S^{(h)}) \leq k + k/2$$

"Akin to a random 3-OPT move after several 2-OPT moves for TSP"

Variants and extensions (under investigation)

- **<u>Tabu search by branch-and-cut:</u>**

  - The black-box MIP solver is used to explore a suitable neighbourhood $N(x^H, k)$ of the current target solution x^H
  - the neighbourhood is defined by a **<u>restricted</u>** **MIP model** defining a proper subset of the feasible solution space, e.g.

    - not too far from the current target solution:

$$x(S^H) \geq |S^H| - K$$

    - tabu moves:

$$x(S^H) \leq |S^H| - 1$$

    - big-M coefficient reductions of the type

$$\sum_{j \in F} x_j \leq |F| \, y_j \;\Rightarrow\; \sum_{j \in F} x_j \leq \left[\sum_{j \in F} x_j^H \right] y_j$$

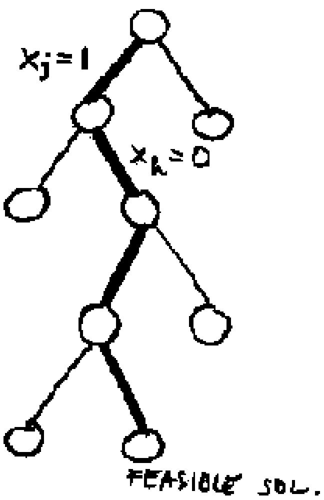
    - possibly: problem-specific constraints …

- **Working with percentage gap closed**

$$\begin{cases} \min \; c^T x \\ \sum_{j \in S^H} x_j \geq |S^H| - k \\ Ax \geq b \\ 0 \leq x_j \leq 1, \; j \in \mathcal{B} \end{cases} \qquad \begin{cases} k := \left\lceil \max \sum_{j \in S^H} x_j \right\rceil \\ c^T x \leq \alpha \, LB + (1-\alpha) UB \\ Ax \geq b \\ 0 \leq x_j \leq 1, \; j \in \mathcal{B} \end{cases}$$

"Fixed k" "% gap closed"

- **Smart backtracking**



$$F_1 := \{ j \in \mathcal{B} : x_j^* = 1 \}$$

$$F_0 := \{ j \in \mathcal{B} : x_j^* = 0 \}$$

$$\underbrace{\sum_{j \in F_0} x_j + \sum_{j \in F_1} (1 - x_j)}_{\text{n. of flipped var.s}} \leq K$$

MIPLIP problem seymour.lp

CPLEX 7.0: MIP emphasis: optimality

- Elapsed b&b time = 151.02 sec. : 435.0
- Final Sol. after 18000.0 sec.s : 435.0

CPLEX 7.0: MIP emphasis: integer feasibility

- Elapsed b&b time = 118.85 sec. : 459.0
- Elapsed b&b time = 202.98 sec. : 456.0
- 455.0
- 454.0
- Elapsed b&b time = 222.97 sec. : 453.0
- Elapsed b&b time = 304.97 sec. : 435.0
- Elapsed b&b time = 479.85 sec. : 432.0
- Elapsed b&b time = 2380.52 sec. : 431.0
- Elapsed b&b time = 2772.62 sec. : 430.0
- Elapsed b&b time = 3162.93 sec. : 429.0
- Elapsed b&b time = 4507.88 sec. : 428.0
- Elapsed b&b time = 7605.32 sec. : 427.0
- Final Sol. after 18000.0 sec.s : 427.0

CPLEX 7.0 & Local Branching (k=10)

- Local Branch Time = 151.8 sec. : 435.0
- Local Branch Time = 392.1 sec. : 430.0
- Local Branch Time = 404.8 sec. : 427.0
- Local Branch Time = 826.6 sec. : 426.0
- Local Branch Time = 1122.3 sec. : 425.0
- Local Branch Time = 1608.5 sec. : 424.0
- Local Branch Time = 2470.9 sec. : 423.0

MIPLIP: problem arki001.lp

CPLEX 7.0: MIP emphasis: optimality

- Elapsed b&b time = 21.12 sec. 7,594,629.2
- 7,590,295.2
- 7,590,247.2
- Elapsed b&b time = 212.32 sec. 7,585,194.4
- Elapsed b&b time = 1897.90 sec. 7,584,116.1
- Elapsed b&b time = 2088.58 sec. 7,583,895.3
- 7,583,878.4
- Elapsed b&b time = 2450.85 sec. 7,582,953.8
- Elapsed b&b time = 2613.20 sec. 7,582,840.6
- Elapsed b&b time = 4160.22 sec. 7,582,751.4
- Elapsed b&b time = 6216.88 sec. 7,582,634.8
- Elapsed b&b time = 7161.85 sec. 7,582,414.4
- Elapsed b&b time = 7161.85 sec. 7,582,302.6
- Elapsed b&b time = 14322.80 sec. 7,582,202.7
- Elapsed b&b time = 16237.02 sec. 7,582,031.3
- Elapsed b&b time = 16237.02 sec. 7,582,024.4

CPLEX 7.0 & Local Branching (k=10)

- Time = 21.6 best sol = 7,594,629.28
- Time = 471.9 best sol = 7,581,831.57
- Time = 922.0 best sol = 7,581,267.36
- Time = 2279.9 best sol = 7,580,980.56
- Time = 2730.1 best sol = 7,580,947.53
- Time = 3630.4 best sol = 7,580,937.90
- Time = 4080.5 best sol = 7,580,925.20

Hard nesting problem (broken glass) glass4.lp

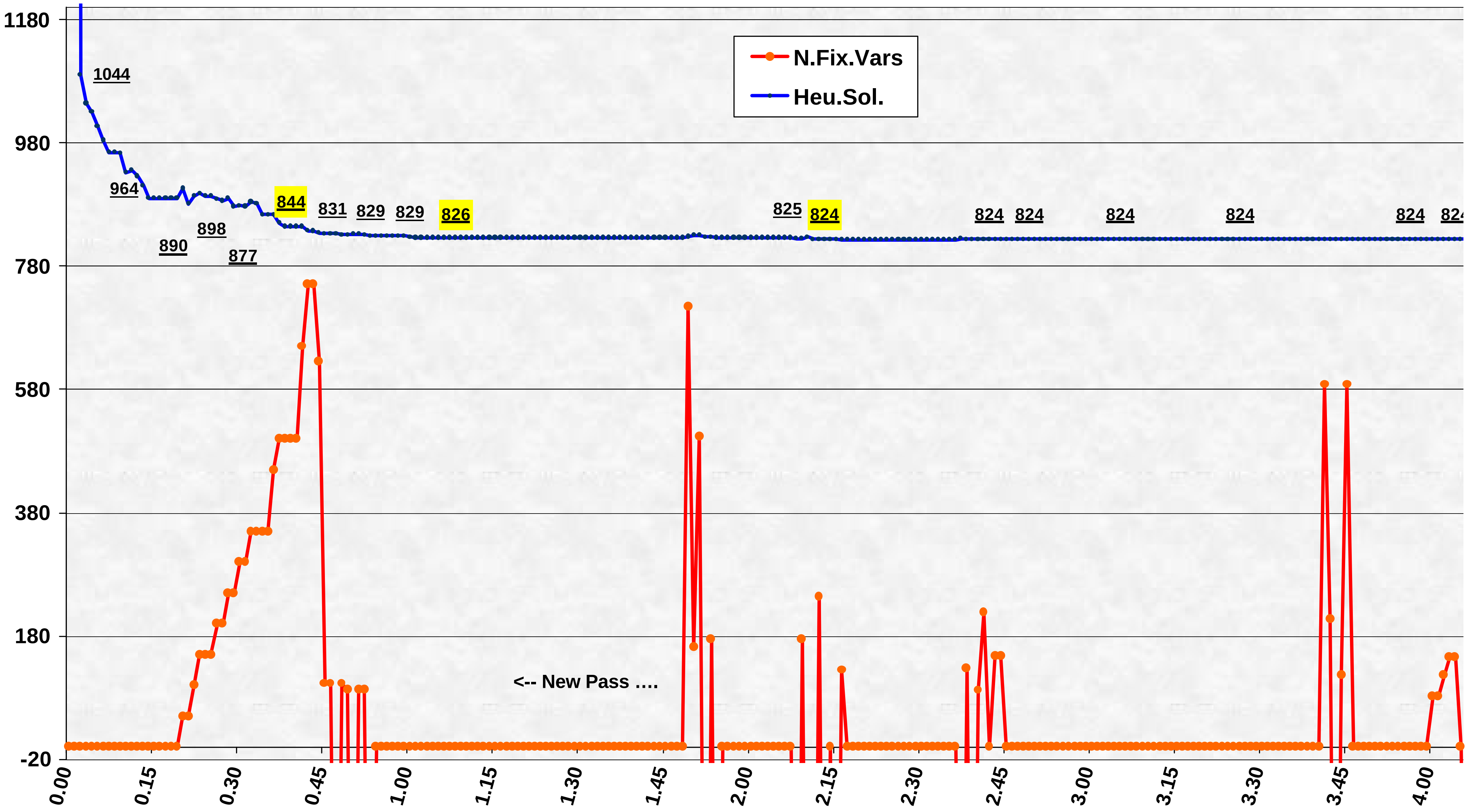
CPLEX 7.0: MIP emphasis: optimality

- Elapsed b&b time = 4.42 sec. 2.9334e+09
- Elapsed b&b time = 14.83 sec. 2.8334e+09
- Elapsed b&b time = 68.20 sec. 2.3000e+09
- Elapsed b&b time = 100.58 sec 2.2800e+09
- Elapsed b&b time = 354.20 sec. 2.1000e+09
- Elapsed b&b time = 1062.98 sec. 2.0867e+09
- Elapsed b&b time = 1257.15 sec. 2.0500e+09
- Elapsed b&b time = 3436.65 sec. 2.0125e+09
- 1.9500e+09
- Elapsed b&b time = 3922.70 sec. 1.9334e+09
- Elapsed b&b time = 5711.65 sec. 1.8625e+09
- Elapsed b&b time = 6119.68 sec. 1.8500e+09
- Elapsed b&b time = 10770.95 sec. 1.8133e+09
- Elapsed b&b time = 14460.53 sec. 1.8000e+09

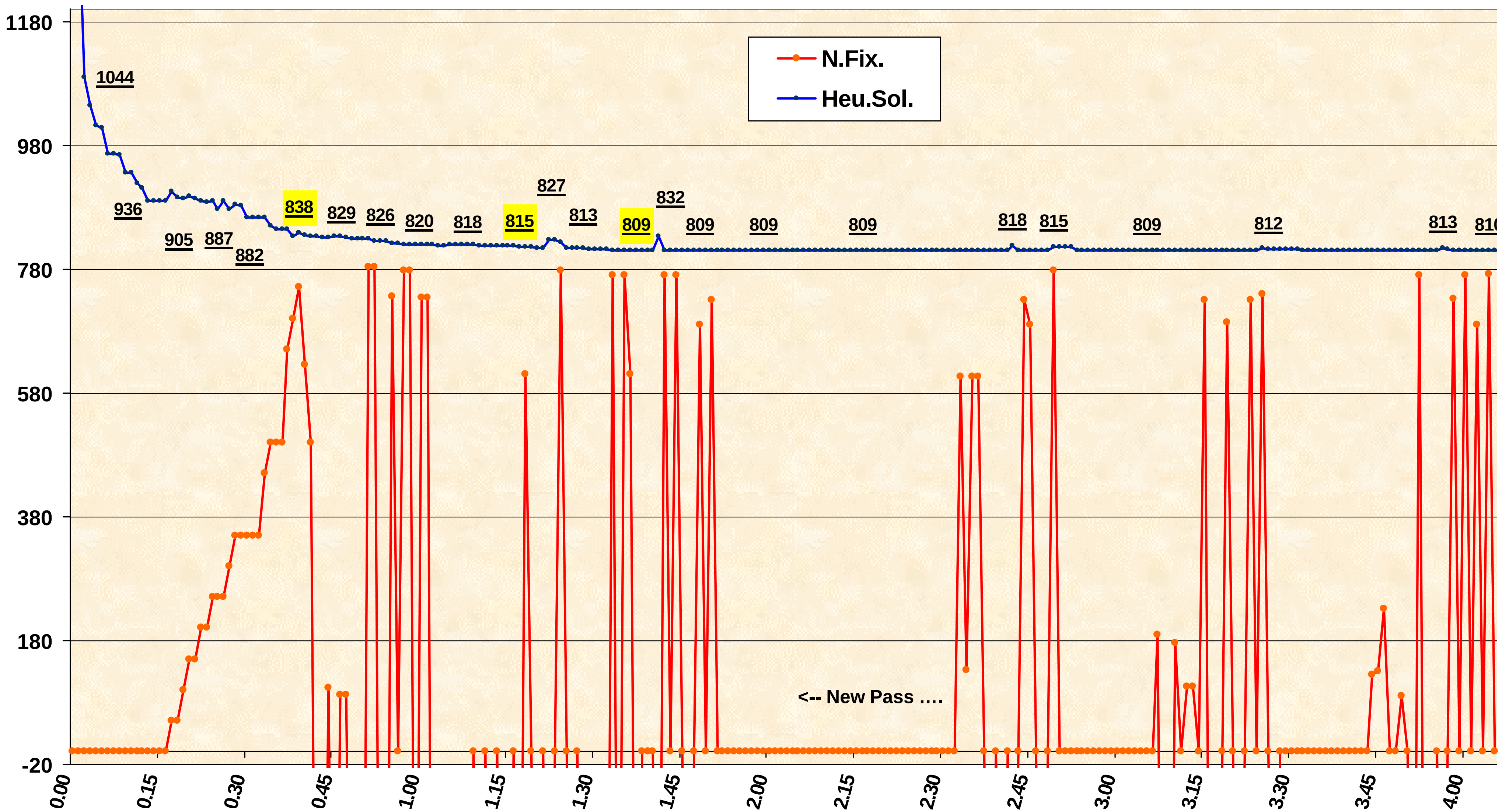
CPLEX 7.0 & Local Branching (k=10)

- Time = 1.9 best sol = 2,933,355,933.33
- Time = 302.1 best sol = 1,850,015,600.00
- Time = 602.3 best sol = 1,800,015,033.33
- Time = 902.5 best sol = 1,750,015,100.00
- Time = 1202.7 best sol = 1,744,458,422.22
- Time = 1803.0 best sol = 1,640,013,670.00
- Time = 7137.6 best sol = 1,600,015,950.00
- Time = 17513.9 best sol = 1,600,013,400.00

<u>Example</u>: 800+ driver duties at NSR (the Dutch railways) computed by TURNI <u>without local branch</u>



Example: 800+ driver duties at NSR (the Dutch railways) computed by TURNI with local branch



Hard crew scheduling (NSR): nsr8k.lp

Just the TURNI core problem of the previous real-world crew scheduling instance

CPLEX 7.0: MIP emphasis: optimality

Root relaxation solution time = 6509.12 sec.

	Node	Nodes Left	Objective	IInf	Best Integer	Cuts/ Best Node	Gap
	0	0	1.7501e+07	3653		1.7501e+07	
			1.7501e+07	3673		Fractcuts: 41	
*	0+	0	4.1831e+08	0	4.1831e+08	1.7501e+07	95.82%
	10	10	1.7507e+07	3680	4.1831e+08	1.7501e+07	95.82%
*	10+	10	3.2858e+08	0	3.2858e+08	1.7501e+07	94.67%
	20	20	1.7523e+07	3658	3.2858e+08	1.7501e+07	94.67%
	30	30	1.7531e+07	3618	3.2858e+08	1.7501e+07	94.67%

Final Sol. after 36,000.0 sec.s **328,581,877.40**

CPLEX 7.0 & Local Branching (k=10)

- Time = 12,820.0 best sol = 418,308,979.25
- Time = 13,956.1 best sol = 123,196,426.17
- Time = 16,356.4 best sol = 51,215,212.14
- Time = 18,759.4 best sol = 42,812,004.17
- Time = 21,160.0 best sol = 24,064,072.04
- Time = 23,561.2 best sol = 23,189,634.03
- Time = 28,362.8 best sol = 22,812,458.03
- Time = 33,167.6 best sol = **22,757,539.03**

TURNI: 809-duty sol. in about 5,400 sec.s **18,257,835.61**