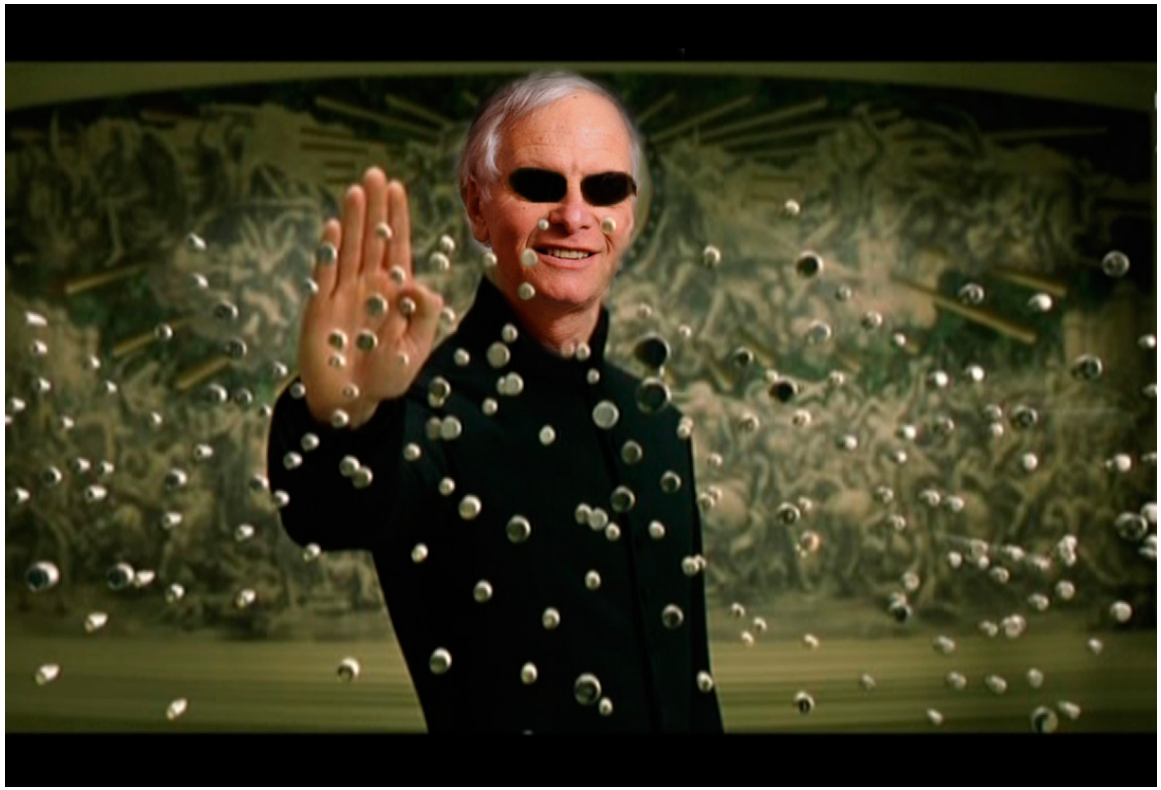
George Reloaded

M. Monaci (University of Padova, Italy)

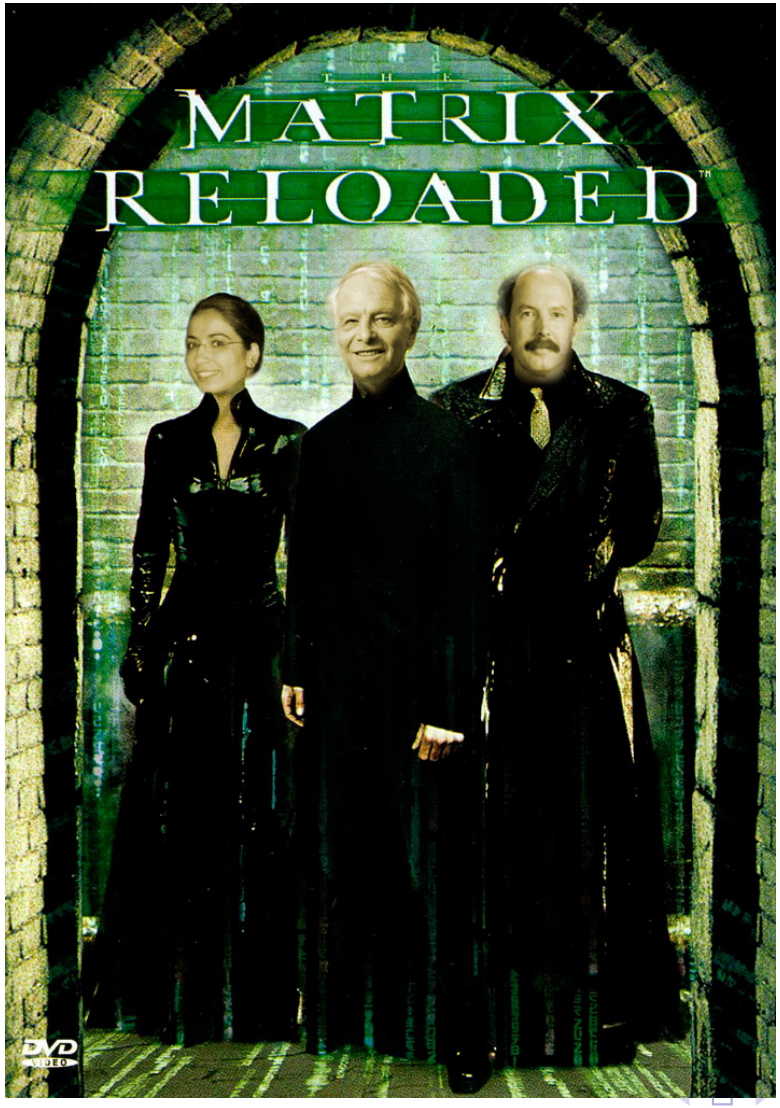
joint work with M. Fischetti

MIP Workshop, July 2010



Why George?

Because of Karzan, Nemhauser, Savelsbergh "Information-based branching schemes for binary linear mixed integer problems" (MPC, 2009)



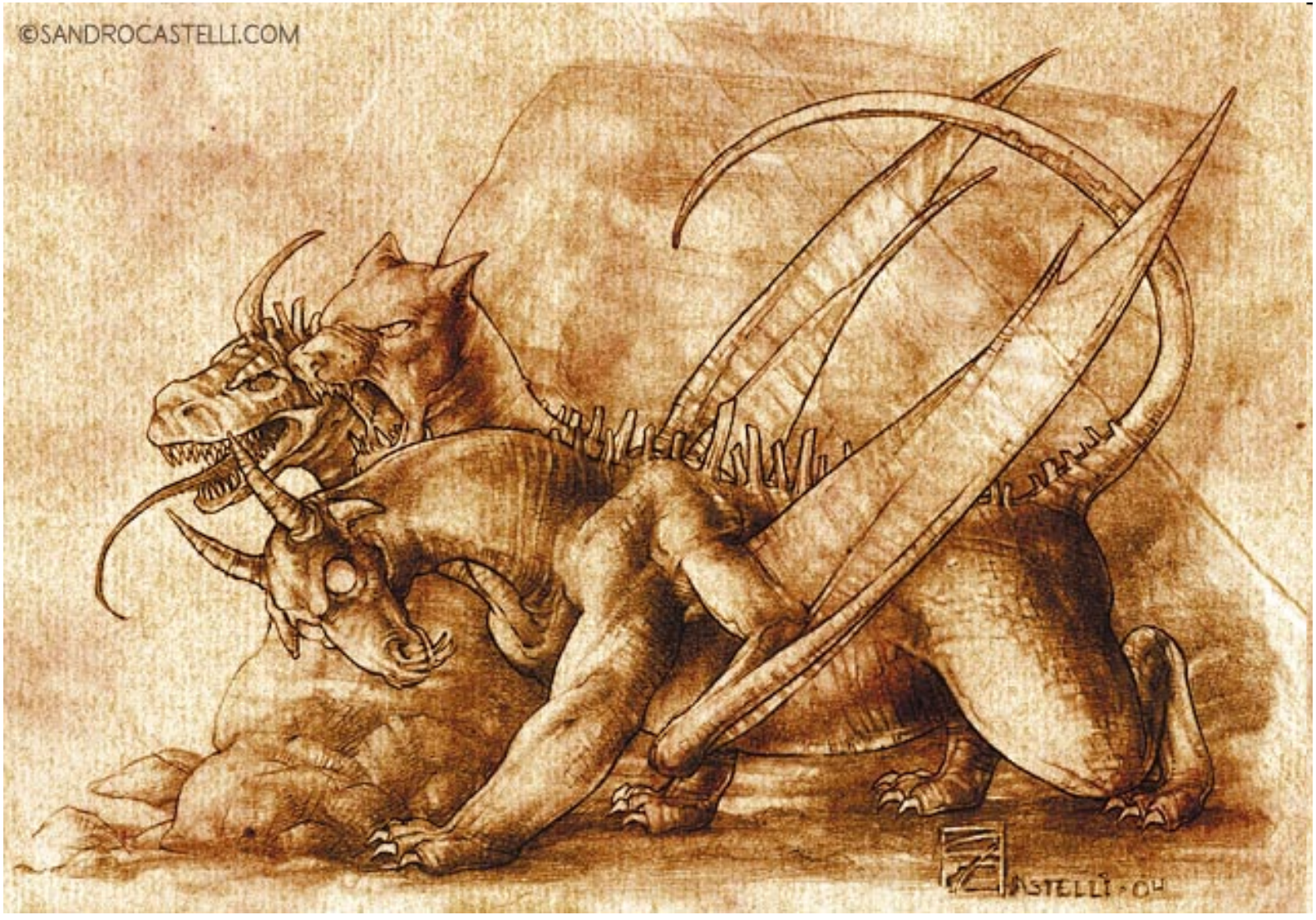
Branching on Nonchimerical Fractionalities

M. Monaci (University of Padova, Italy)

joint work with M. Fischetti

MIP Workshop, July 2010

Why chimerical?



Why chimerical?

Chimerical

- Created by or as if by a wildly fanciful imagination; highly improbable.
- Given to unrealistic fantasies; fanciful.
- Of, related to, or being a chimera.

Outline

Branching

- Computationally speaking, branching is one of the most crucial steps in Branch&Cut for general MIPs.
- We will concentrate on binary branching on a single variable (still the most used policy)
- Key problem: how to select the (fractional) variable to branch on?
- Recent works on this subject:
  - Linderoth and Savelsbergh (IJOC, 1999)
  - Achterberg, Koch, Martin (ORL, 2005)
  - Patel and Chinneck (MPA, 2007)
  - Karzan, Nemhauser, Savelsbergh (MPC, 2009)

Discussion

- Branching on a most-fractional variable is not a good choice.
- It is alike random branching.

Why?

- Think of a knapsack problem with side constraints:
  - Large vs small items;
  - Because of side constraints, many fractionalities might arise (LP wants to squeeze 100% optimality, taking full advantage of any possible option);
  - E.g., we could have a single large item with small fractionality, and a lot of small items with any kind of fractionality;
  - Branching should detect the large fractional item ...
  - ... but the presence of a lot of other fractionalities may favor small items
- Even worse: MIPs with big-M coefficients to model conditional constraints
  - An "important" binary variable activating a critical constraint may be very close to zero $\rightarrow$ never preferred for branching.

Full Strong Branching

- Proposed by Applegate, Bixby, Chvàtal and Cook in 1995.
- Simulate branching on all fractional var.s, and choose one among those that give "the best progress in the dual bound".
- ... "finding the locally best variable to branch on".
- Computationally quite demanding: in practice, cheaper methods:
  - strong branching on a (small) subset of variables;
  - hybrid schemes based on pseudocosts;
  - ...

Chimeras

- We need to distinguish between:
  - "structural" fractionalities, that have a large impact in the current MIP;
  - "chimerical" fractionalities, that can be fixed by only a small deterioration of the objective function.
- Strong branching can be viewed as a computationally-expensive way to detect **chimerical** fractionalities.
  - Try and fix every fractionality by simulating branching;
  - 2 LPs for each candidate
- (Studying the implication of branching through, e.g., constraint propagation is also another interesting option, see Patel and Chinneck)

Our starting point . . .

	test set	random	most inf	least inf	pseudocost	full strong	strong	hybr
time	MIPLIB	+139	+139	+266	+16	+92	+38	
	CORAL	+332	+314	+575	+40	+97	+59	
	MILP	+81	+86	+109	+23	+107	+44	
	ENLIGHT	+115	−40	+149	−27	+45	+11	
	ALU	+1271	+1991	+1891	+619	+180	+13	
	FCTP	+288	+267	+379	+35	+36	+25	
	ACC	+52	+85	+138	−41	+174	+82	
	FC	+912	+1152	+837	+98	+14	+18	
	ARCSET	+1276	+1114	+1296	+106	+112	+72	
	MIK	+10606	+10606	+9009	+102	+59	+8	
	total	**+226**	**+219**	**+341**	**+33**	**+95**	**+44**	
nodes	MIPLIB	+475	+341	+1096	+87	−65	−62	
	CORAL	+694	+517	+1380	+79	−79	−68	
	MILP	+194	+187	+306	+71	−72	−59	
	ENLIGHT	+163	−29	+219	+3	−83	−85	
	ALU	+6987	+5127	+9084	+1659	−60	−78	
	FCTP	+511	+443	+931	+103	−73	−68	
	ACC	+393	+513	+1422	+88	−95	−52	
	FC	+5542	+5060	+6039	+603	−81	−73	
	ARCSET	+3219	+2434	+3573	+248	−60	−51	
	MIK	+8994	+7397	+9195	+123	−90	−86	
	total	**+543**	**+428**	**+976**	**+98**	**−75**	**−65**	

Table 5.1. Performance effect of different branching rules for solvi
denote the percental changes in the shifted geometric mean of th
of branching nodes (bottom) compared to the default *hybrid reliab*
Positive values represent a deterioration, negative values an improv

Computation (Actherberg's thesis)

	test set	random	most inf	least inf	pseudocost	full strong	strong	hybr strong	psc strinit	reliability	inference
time	MIPLIB	+139	+139	+266	+16	+92	+38	+20	+5	−1	+101
	CORAL	+332	+314	+575	+40	+97	+59	+27	+2	+7	+177
	MILP	+81	+86	+109	+23	+107	+44	+43	+9	+6	+20
	ENLIGHT	+115	−40	+149	−27	+45	+11	+9	+27	+5	−70
	ALU	+1271	+1991	+1891	+619	+180	+13	+11	+55	+36	−35
	FCTP	+288	+267	+379	+35	+36	+25	+4	+14	+2	+187
	ACC	+52	+85	+138	−41	+174	+82	+153	+11	+84	−24
	FC	+912	+1152	+837	+98	+14	+18	+14	−5	−2	+188
	ARCSET	+1276	+1114	+1296	+106	+112	+72	+38	+18	−1	+317
	MIK	+10606	+10606	+9009	+102	+59	+8	+11	+35	+2	+5841
	total	**+226**	**+219**	**+341**	**+33**	**+95**	**+44**	**+30**	**+8**	**+6**	**+95**
nodes	MIPLIB	+475	+341	+1096	+87	−65	−62	−18	+13	−7	+269
	CORAL	+694	+517	+1380	+79	−79	−68	−12	+18	+16	+329
	MILP	+194	+187	+306	+71	−72	−59	+41	+40	+7	+76
	ENLIGHT	+163	−29	+219	+3	−83	−85	+1	+23	−8	−49
	ALU	+6987	+5127	+9084	+1659	−60	−78	−31	+120	+17	+6
	FCTP	+511	+443	+931	+103	−73	−68	+6	+39	0	+364
	ACC	+393	+513	+1422	+88	−95	−52	+392	+31	+404	+33
	FC	+5542	+5060	+6039	+603	−81	−73	−28	+54	0	+1137
	ARCSET	+3219	+2434	+3573	+248	−60	−51	−6	+37	0	+742
	MIK	+8994	+7397	+9195	+123	−90	−86	+1	+32	−1	+4652
	total	**+543**	**+428**	**+976**	**+98**	**−75**	**−65**	**+3**	**+27**	**+9**	**+217**

Table 5.1. Performance effect of different branching rules for solving MIP instances. The values denote the percental changes in the shifted geometric mean of the runtime (top) and number of branching nodes (bottom) compared to the default *hybrid reliability/inference branching* rule. Positive values represent a deterioration, negative values an improvement.

Our goal

- Full strong branching (FSB) vs the SCIP basic benchmark version
  - FSB doubles computing time, but
  - reduces the number of nodes of 75%,
  - though other effective options are available
- A speedup of just 2x would suffice to make FSB the fastest method!
- OUR GOAL:
  Find a computationally cheaper way to gather the same information as FSB (or something similar) by solving fewer LPs, so as to (at least) halve computing times.

Full Strong Branching (FSB)

- Let $\texttt{score}(LB0, LB1)$ be the score function to be maximized e.g., $\texttt{score} = \min(LB0, LB1)$, $\texttt{score} = LB0 * LB1$, or alike
- Assume `score` is monotone: decreasing $LB0$ and/or $LB1$ cannot improve the score;
- FSB: for each fractional variable x_j
  - (i) compute $LB0_j$ (resp. $LB1_j$) as the LP worsening when branching down (resp. up) on x_j
  - (ii) compute $\text{score}_j = \texttt{score}(LB0_j, LB1_j)$

  and branch on a variable x_j with maximum score.
- We aim at saving the solution of some LPs during the FSB computation.

First Step: parametrized FSB

- let x^* be the node frac. sol., and F its fractional support
- Simple data structure: for each $j \in F$
  - $LB0_j$ and $LB1_j$ (initially $LB0_j := LB1_j := \infty$)
  - $f0_j$ indicating whether $LB0_j$ has been computed exactly or it is just an upper bound (initially $f0_j :=$ `FALSE`); same for $f1_j$
- Updating algorithm
  - Look for the candidate variable x_k that has maximum score,
  - if ($f0_k = f1_k =$ `TRUE`) DONE
  - exactly compute $LB0_k$ (or $LB1_k$), update $f0_k$ (or $f1_k$), and repeat
  - KEY STEP: whenever a new LP solution $\widetilde{x}$ is available:
    - possibly update $LB0_j$ if $\widetilde{x}_j \leq \lfloor x_j^* \rfloor$ for some $j \in F$
    - possibly update $LB1_j$ if $\widetilde{x}_j \geq \lceil x_j^* \rceil$ for some $j \in F$

Parametrized FSB

1. let x^* be the node frac. sol., and F its fractional support
2. set $LB0_j := LB1_j := \infty$ and $f0_j := f1_j :=$ `FALSE` $\forall j \in F$
3. compute $\text{score}_j = \texttt{score}(LB0_j, LB1_j) \quad \forall j \in F$
4. let $k = \arg\max\{\text{score}_j : j \in F\}$
5. if $(f0_k = f1_k =$ `TRUE`$)$ return(k)
6. if $(f0_k =$ `FALSE`$)$
   - solve LP with additional constraint $x_k \leq \lfloor x_k^* \rfloor$ $(\rightarrow$ solution $\widetilde{x})$
   - set $\delta = c^T\widetilde{x} - c^T x^*$, $LB0_k = \delta$, and $f0_k =$ `TRUE`
   - $\forall j \in F$ s.t. $\widetilde{x}_j \leq \lfloor x_j^* \rfloor$ (resp. $\widetilde{x}_j \geq \lceil x_j^* \rceil$)
     - set $LB0_j = \min\{LB0_j, \delta\}$ (resp. $LB1_j = \min\{LB1_j, \delta\}$)
     - if $(\delta = 0)$, set $f0_j =$ `TRUE` (resp. $f1_j =$ `TRUE`)
   - goto 3
7. else . . . the same for $LB1_k$ and $f1_k$

Testbed

- Set of 24 instances considered by Achterberg, Koch, Martin.
- Branching rule imposed within Cplex branch-and-bound.
- All heuristics and cut generation procedures have been disabled
  - optimal solution value used as an upper cutoff;
  - instances obtained after preprocessing and root node using Cplex default parameters.
- Test on a PC Intel Core i5 750 @ 2.67GHz
- 4 instances have been removed (unsolved with standard FSB in 18,000 secs).
- Different `score` functions
  - $\texttt{score}(a, b) = \min(a, b)$
  - $\texttt{score}(a, b) = a * b$
  - $\texttt{score}(a, b) = \mu * \min(a, b) + (1 - \mu) * \max(a, b)$, with $\mu = 1/6$

Parametrized FSB: results

- Lexicographic implementation$\rightarrow$ same number of nodes.
- For each method: arithmetic (and geometric) mean of the computing times (CPU seconds)

	min	prod	μ
FSB	700.68 (163.69)	551.71 (97.53)	450.88 (85.46)
PFSB	394.17 (87.47)	381.19 (69.00)	338.45 (66.04)
% impr.	43.74 (46.56)	30.90 (29.25)	24.93 (22.72)

Second Step: perseverant branching

- **Idea:** if we branched already on a variable, then that is likely to be nonchimerical.
  - Reasonable if the high-level choices have been performed with a "robust" criterion.
- **Implementation:** use strong branching on a restricted list containing the already-branched variables (if empty, use FSB).
  - Related to the backdoor idea of having compact trees (see Dilkina and Gomez, 2009);
  - Similar to strong branching on a (restricted) candidate list (defined, e.g., by means of pseudocosts).

Perseverant branching: results

	min	prod	μ
FSB	700.68 (163.69)	551.71 (97.53)	450.88 (85.46)
PFSB	394.17 (87.47)	381.19 (69.00)	338.45 (66.04)
PPFSB	276.35 (60.45)	174.23 (46.02)	249.41 (52.15)
% impr.	60.55 (63.07)	68.42 (52.81)	44.68 (38.97)

- Small reduction in the number of nodes.

Third Step: asymmetric branching

- **Idea:** for most instances, the DOWN branching is the most critical one;
  - fixing a variable UP is likely to improve the bound anyway ("relevant" choice);
  - not too many UP branchings occur.
- **Implementation:** when evaluating the score, forget about the UP branching:
  - guess that only $LB0$ is useful in computing `score`;
  - at most 1 LP for candidate.

Asymmetric branching: results

	min	prod	μ
FSB	700.68 (163.69)	551.71 (97.53)	450.88 (85.46)
PFSB	394.17 (87.47)	381.19 (69.00)	338.45 (66.04)
PPFSB	276.35 (60.45)	174.23 (46.02)	249.41 (52.15)
APPFSB	197.01 (43.73)	189.40 (40.55)	188.56 (40.31)
% impr.	71.88 (73.28)	65.67 (58.42)	58.17 (52.83)

- Large increase in the number of nodes.

Some further results

- Set of 13 hard instances considered by Karzan, Nemhauser, Savelsbergh.
- Same preprocessing and tuning as for the previous instances.
- 2 instances have been removed (unsolved with standard FSB).

	min	prod	μ
FSB	2638.98 (1264.86)	1995.64 (834.79)	1826.38 (770.40)
PFSB	1344.79 (700.50)	1372.97 (608.88)	1301.54 (590.07)
PPFSB	914.29 (430.06)	782.51 (322.58)	875.45 (410.39)
APPFSB	876.77 (343.39)	759.46 (298.42)	728.20 (293.76)
% impr.	66.77 (72.85)	61.94 (64.25)	60.12 (61.86)

Conclusions and future work

We mainly focused on (pure) full strong branching,
that does not require any tuning,
Future research topics:

1. integration with different state-of-the-art strategies for branching
   - strong branching on a (restricted) candidate list;
   - hybrid: use FSB at the first (high) nodes, then pseudocosts;
   - reliability branching;
   - ...
2. consider a computationally cheaper way of defining nonchimerical fractionalities:
   - Threshold Branching (TB): put a threshold on the LB worsening, and solve a sequence of LPs to drive to integrality as many (chimerical) fractional variables as possible, by using a feasibility pump scheme