Recent Advances in Mixed-Integer Linear Programming

at Carleton University

John W. Chinneck

Systems and Computer Engineering

Carleton University

Ottawa, Canada

Outline

1. Introduction
2. Node selection for faster optimality
   - Important common patterns
3. Active constraints branching variable selection
4. Branching to force change
5. General disjunctions
6. Conclusions

INTRODUCTION

Themes

- Relationship between MILP-*feasibility* and MILP-*optimality*
- Seeking MILP-feasibility quickly
- Focus on *candidate variables*
  - Integer/binary variables that do not have integer/binary values in LP relaxation solution
- Branching to *force change* in the candidate variables

Assumptions

- **Branch and bound method for *minimization***
  - Focus on branching
  - Branching always needed, even in conjunction with cutting, local exploration, root node heuristics, etc.
- **Simplex LP solver**
  - Usual MILP choice, for fast restart at child nodes
- **Measuring solution speed**:
  - *Time*: best
  - *Simplex iterations*: good proxy for time
    - Non-simplex time must be minimal
    - Best choice for multi-core machines where time measurements are not repeatable
  - *Node count*: often poor proxy for time

Interesting patterns…

NODE SELECTION FOR FASTER OPTIMALITY

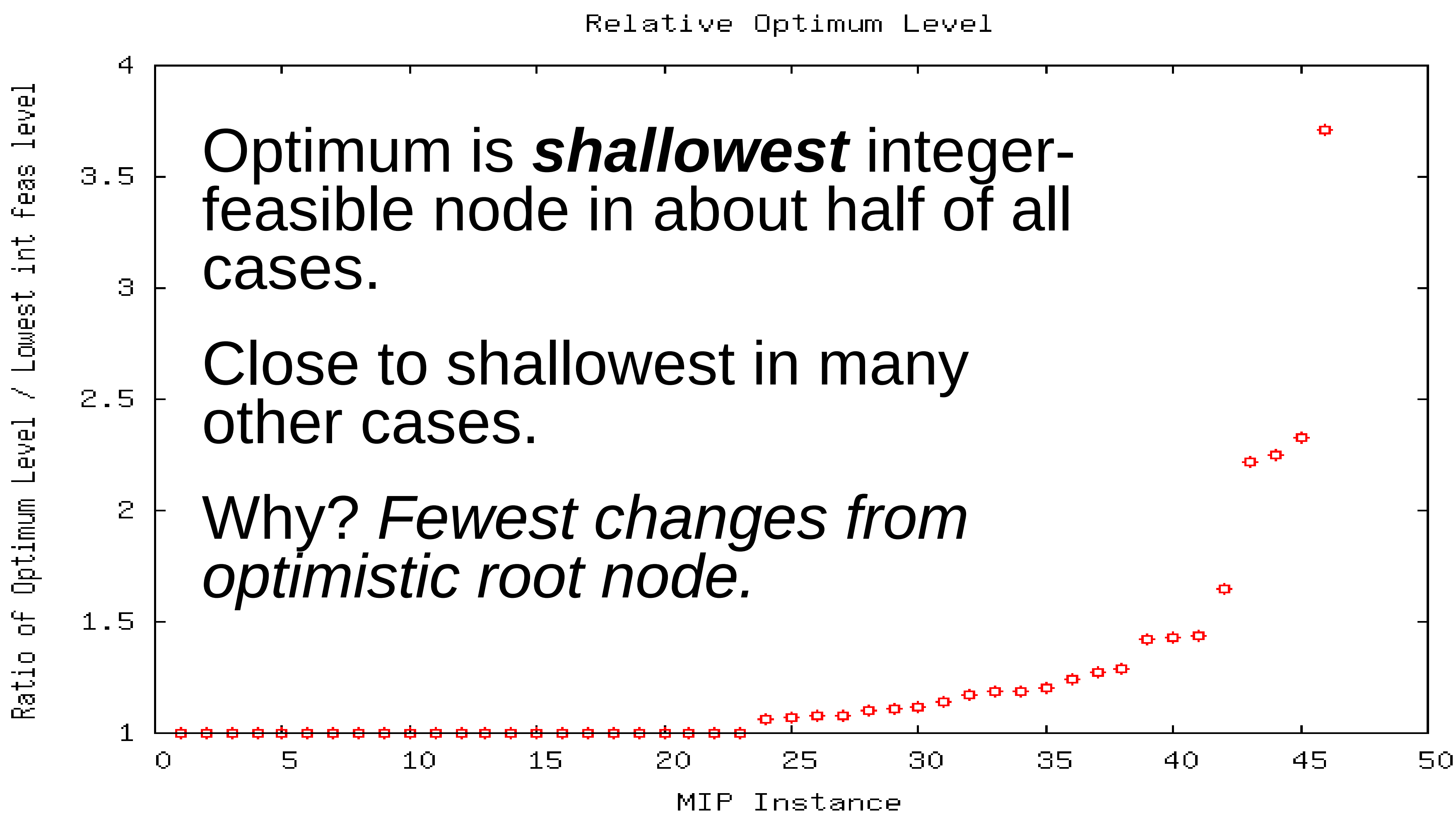
Node selection

- Depth-first
  - Choose child of last solved node
  - *Big advantage*: child node almost identical to parent. Hot start speeds LP solutions!
- Best bound
  - Choose node having best bounding function value anywhere on tree
    - Usually high in the tree
- Best estimate
  - Rate node's progress toward integer feasibility vs. degradation in objective function value
- Others…

Triggering backtrack/jumpback

- Assuming depth-first dive as default behaviour:
  - What conditions trigger backtrack?
  - Which node should be selected?
- *Aspiration level* trigger:
  - Trigger backtrack when node bound is worse than pre-selected aspiration level

Pattern:
Optimality, feasibility, and depth



Optimum is ***shallowest*** integer-feasible node in about half of all cases.

Close to shallowest in many other cases.

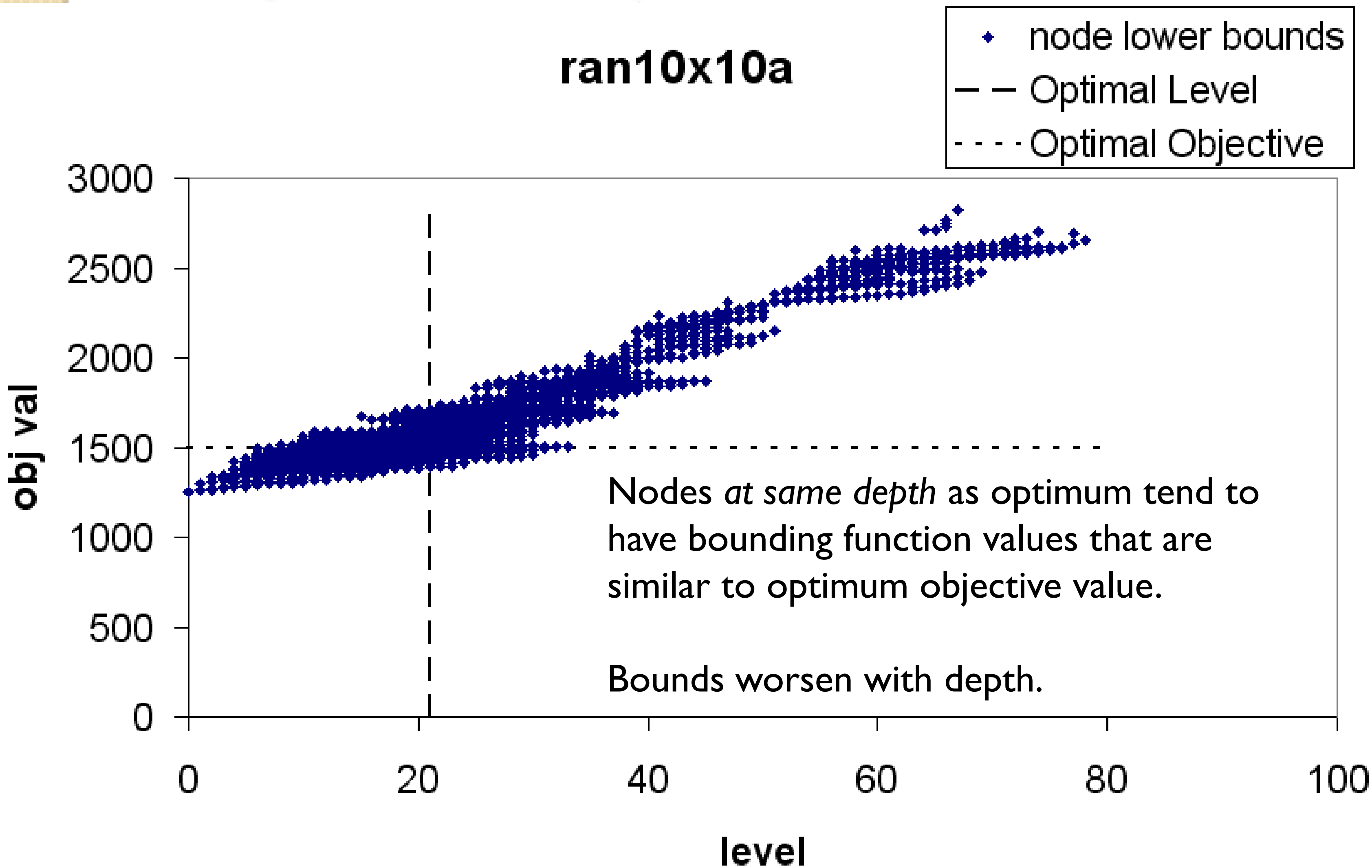
Why? *Fewest changes from optimistic root node.*

Optimality, feasibility, and depth

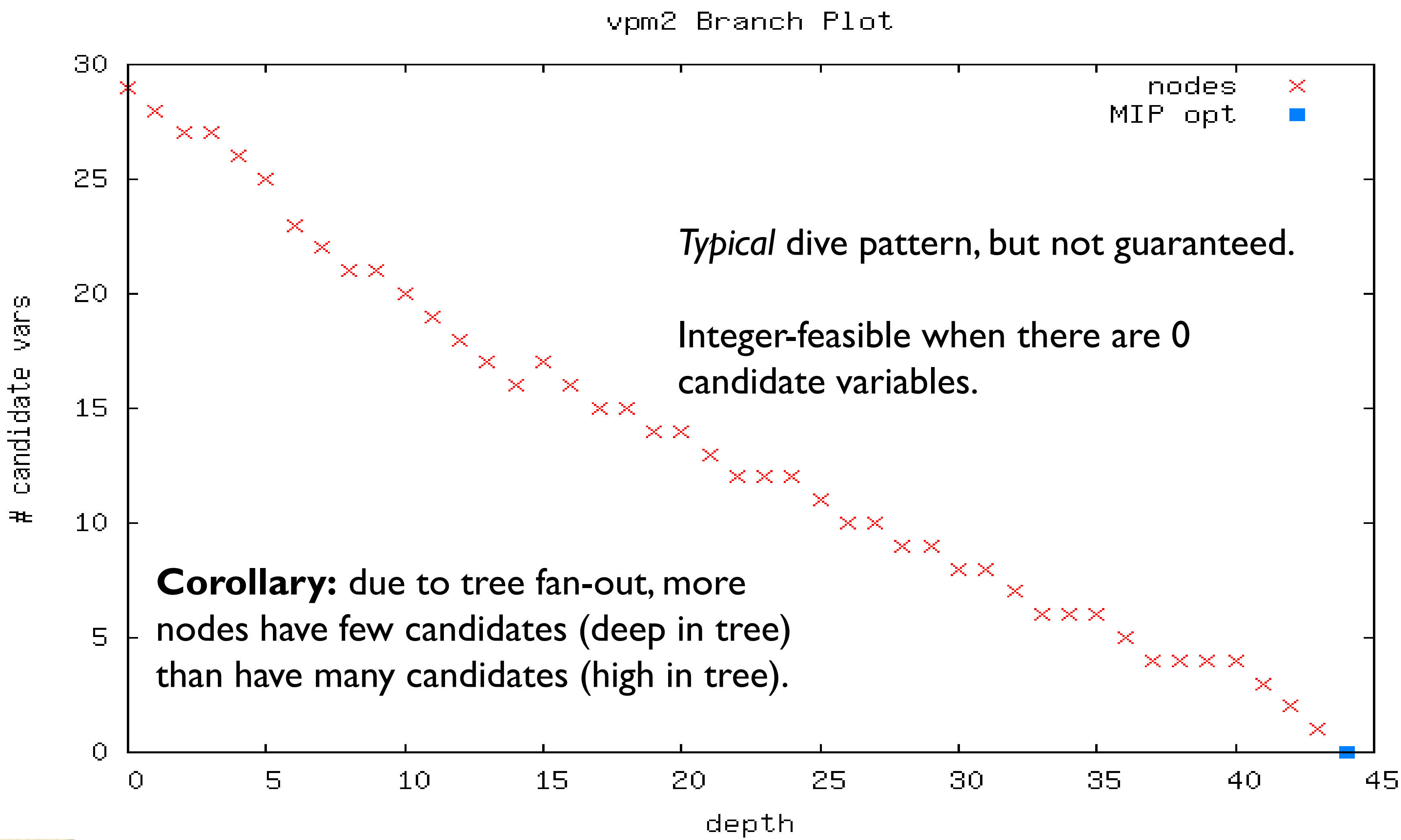
Lesson

- Node selection chooses the *most promising* node
- Shallowest *integer-feasible* descendent most likely to give best objective value
- *Ergo:* fastest integer-feasibility important for optimality

Pattern: Depth and objective value



Pattern:
Candidates decrease with depth

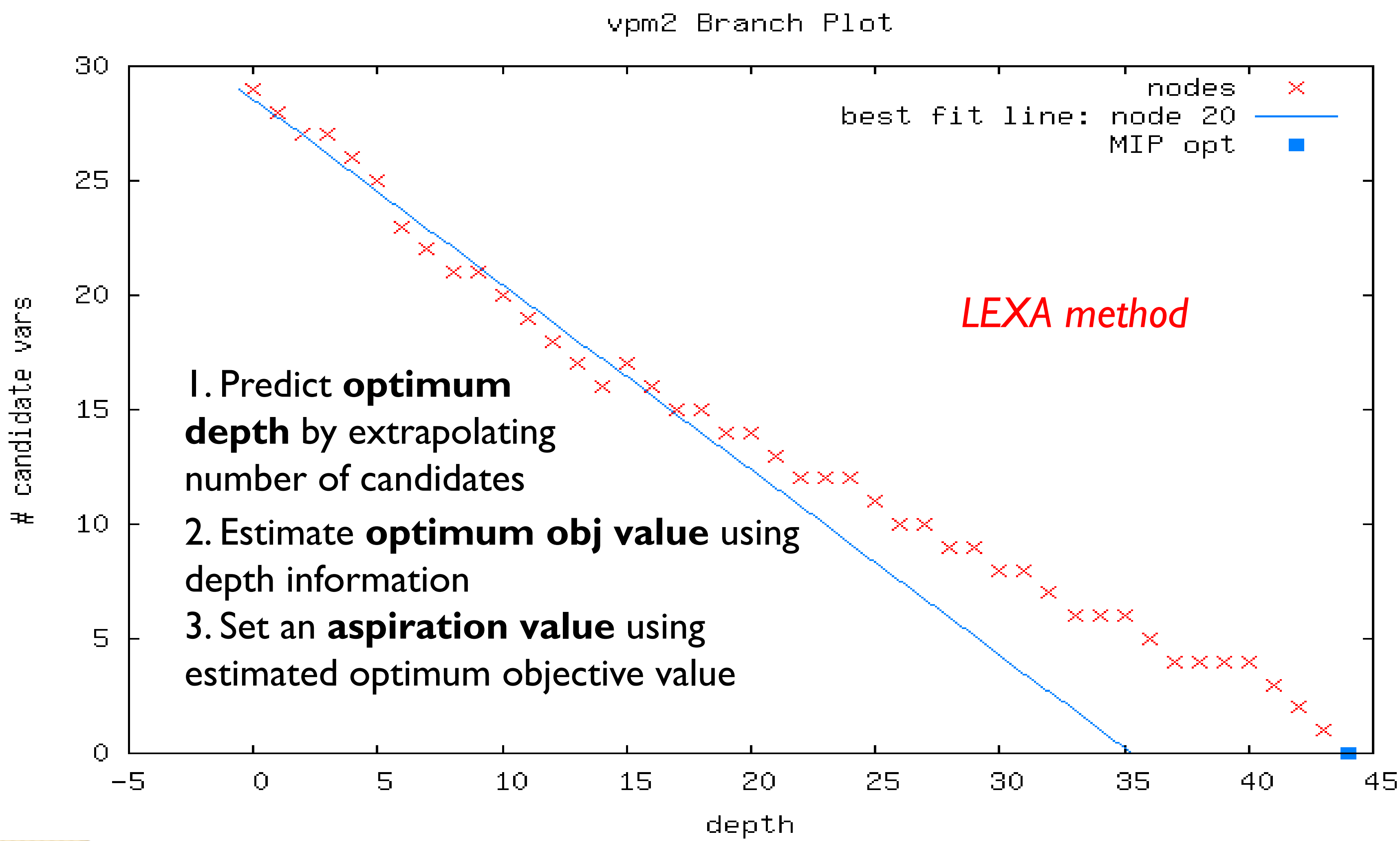


Typical dive pattern, but not guaranteed.

Integer-feasible when there are 0 candidate variables.

Corollary: due to tree fan-out, more nodes have few candidates (deep in tree) than have many candidates (high in tree).

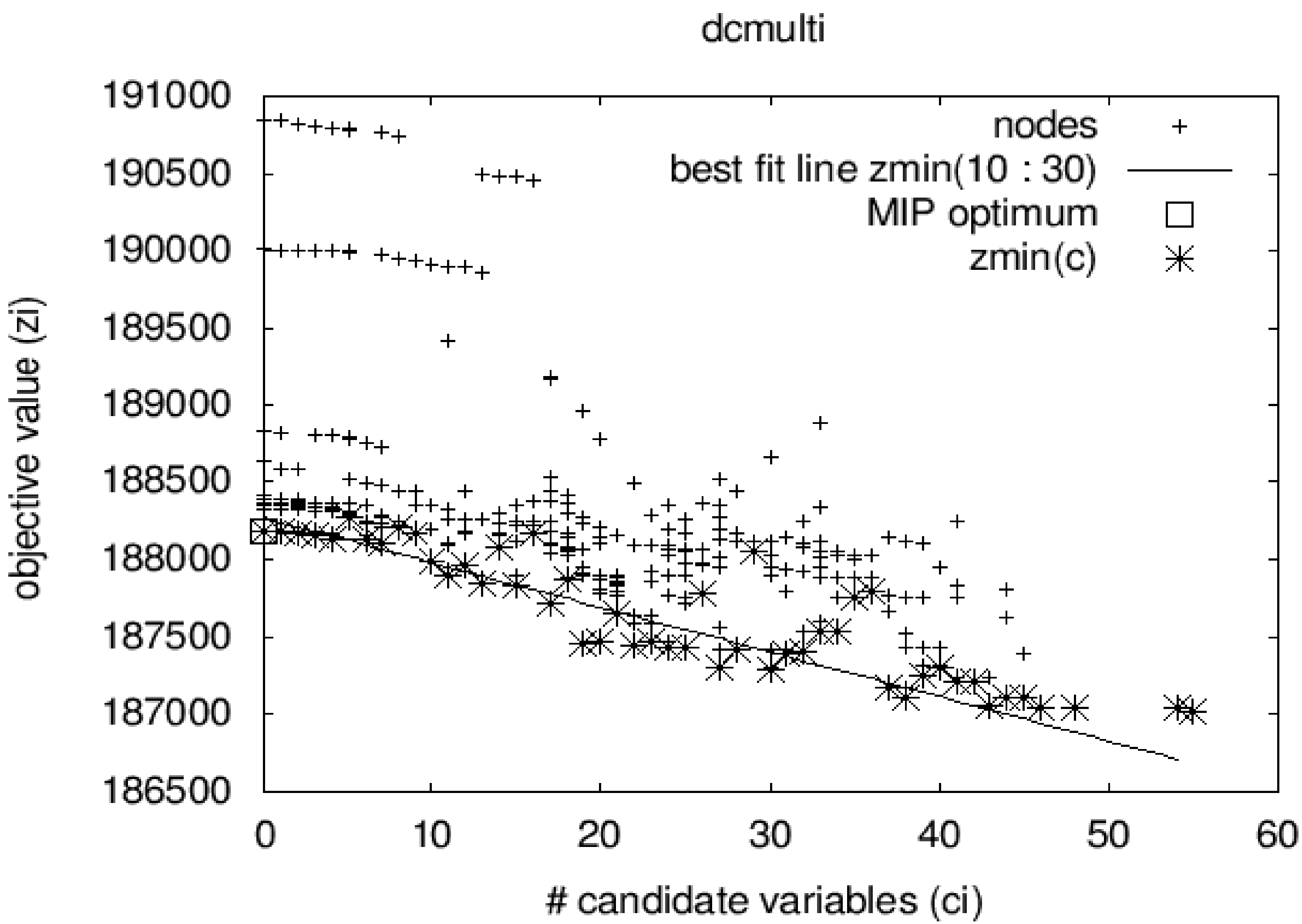
Idea:
Aspiration level by linear extrapolation



LEXA method

1. Predict **optimum depth** by extrapolating number of candidates
2. Estimate **optimum obj value** using depth information
3. Set an **aspiration value** using estimated optimum objective value

Pattern: Objective value vs. candidates



$Z^{min}(c)$: smallest bounding function value given c candidates.

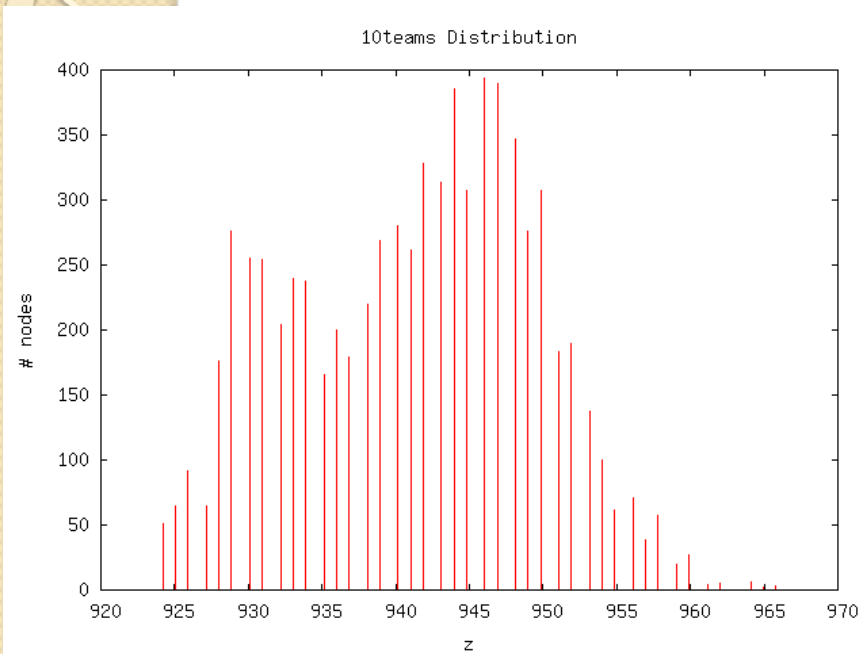
Extrapolate optimum value using $Z^{min}(c)$ over a range of c.

Set aspiration level.

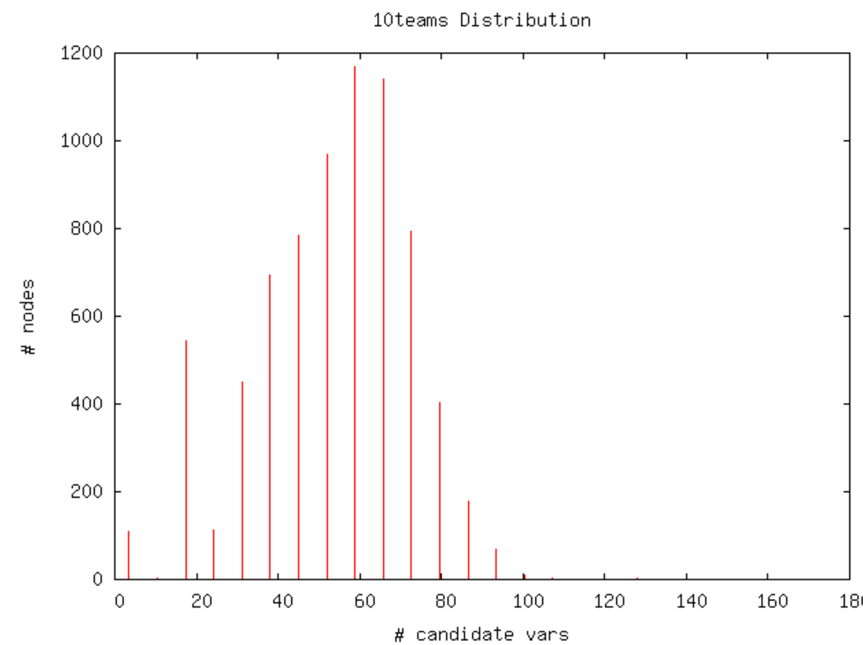
Modified best projection

- Two "anchors":
  - root and node with fewest candidates
- For node selection (MOBP):
  - $Z^{bp(i)} = Z^i + C^i[Z^{min}(C^{min})\text{-}Z^0]/(C^0\text{-}C^{min})$
    - $Z^{bp(i)}$: best projection of Z at node i
    - Z^0, C^0: bounding value, candidates at root node
    - Z^i, C^i: bounding value, candidates at node i
    - C^{min}: minimum candidate variables at any node
- For setting aspiration level (MPAS):
  - Find min($Z^{bp(i)}$) over all active nodes
  - Backtrack if Z^i > min($Z^{bp(i)}$)
- *Does not need incumbent like original does*

Pattern: Common distributions



Distribution of Z^i



Distribution of number of candidate variables

Both distributions often Normal-like

Distribution node selection (DIST)

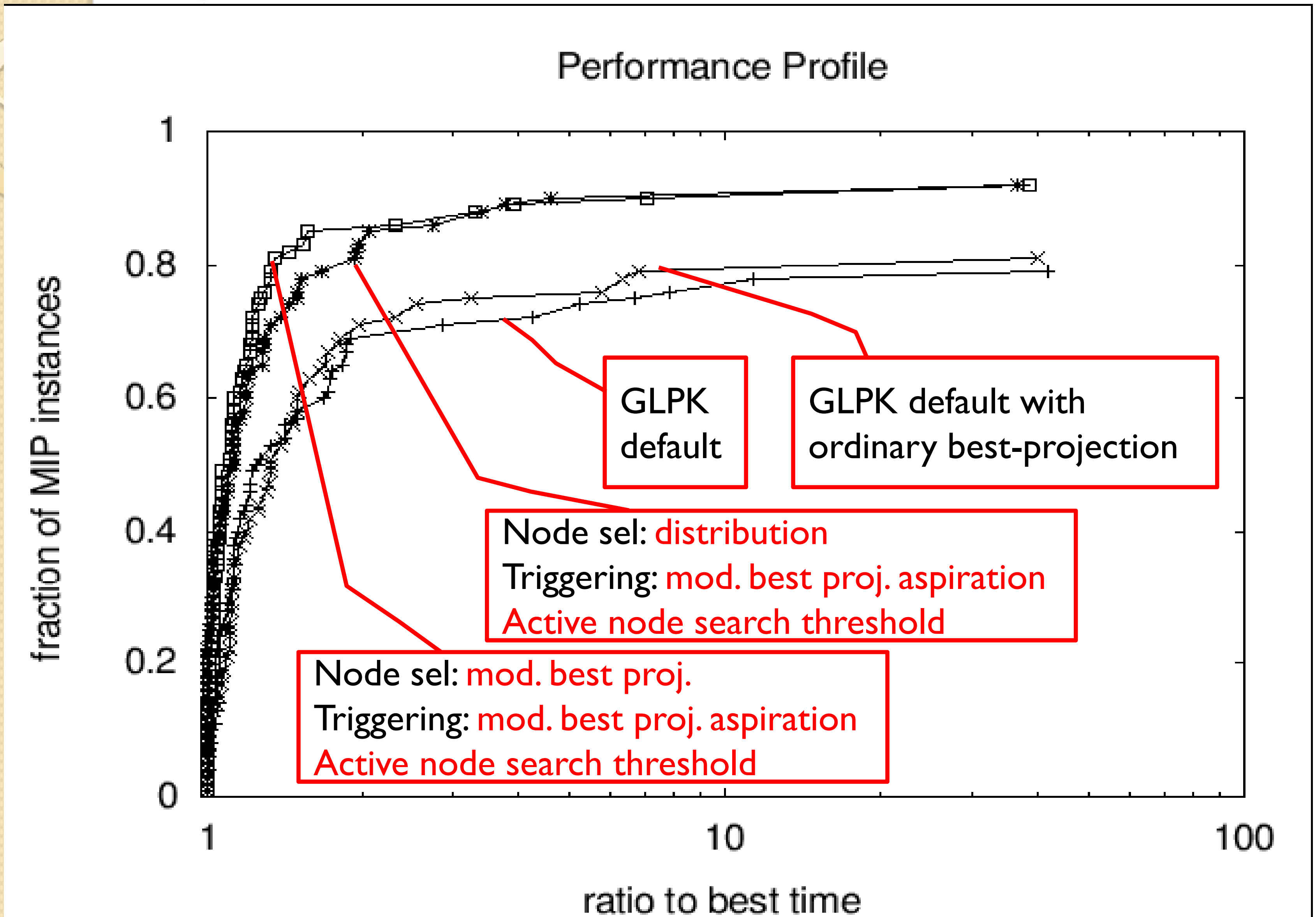
- Balance pursuit of feasibility and optimality
  - Smaller Z^i and C^i both desirable
  - Z^i larger where C^i is smaller, and vice versa

- Ranges quite different: how to balance?
  - Normalize ranges of Z^i and C^i <u>assuming</u> independent normal probability distributions
  - Choose node $n = \arg\min_i P(Z \leq Z^i) \times P(C \leq C^i)$

Idea:
Active node search threshold (ANST)

- Advanced node selection can take a lot of time
- Switch to simple depth-first node selection if current node selection method is taking too much time

Experiments with GLPK 4.9



Lessons learned

- MILP-feasibility (candidates) and optimality are linked
- Patterns relating them can be exploited
- Reaching first integer-feasible solution quickly helps to reach optimality quickly

Goal: reaching first integer-feasible solution quickly

ACTIVE CONSTRAINTS BRANCHING VARIABLE SELECTION

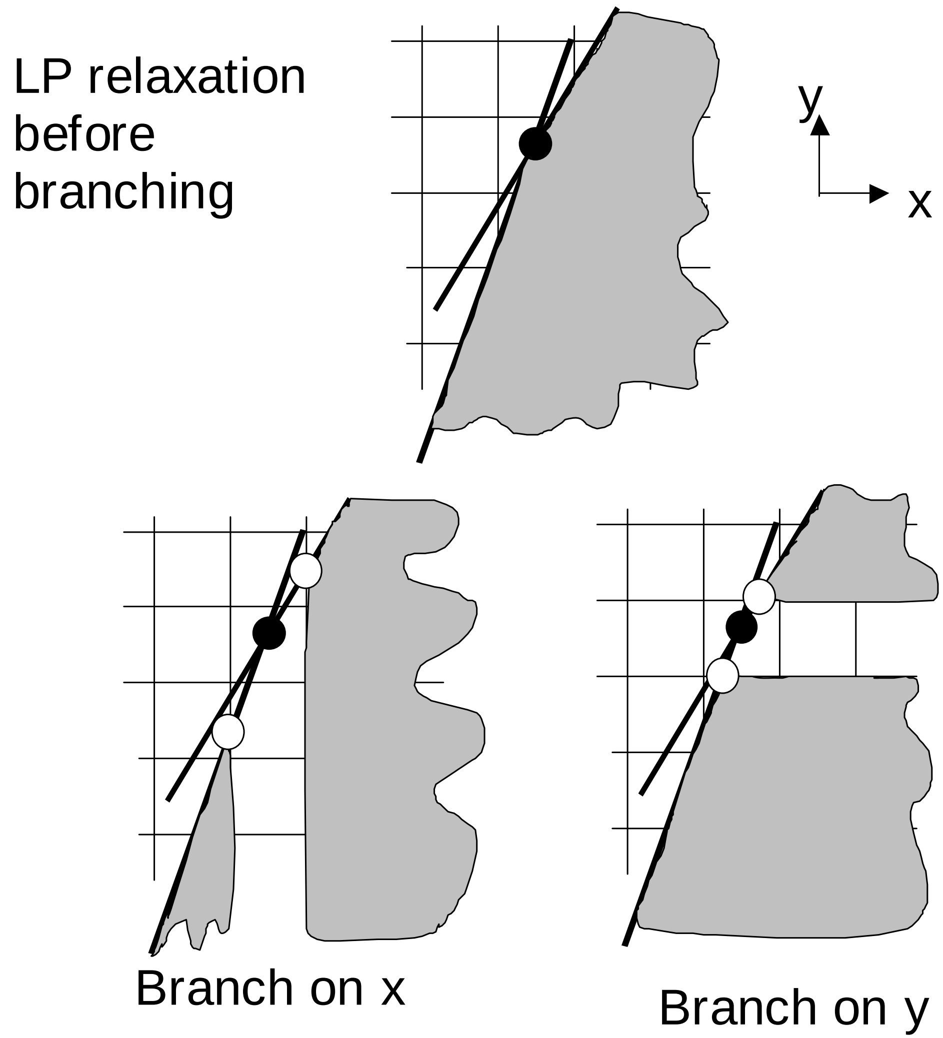
Is branching variable selection important?

	B&B nodes to First Feasible Soln	
model	**Cplex 9.0**	**Active-Constraints Method**
aflow30a	23,481	22 (A, H_M, H_O, O, P)
aflow40b	100,000+ (limit)	33 (H_O, O, P)
fast0507	14,753	26 (A)
glass4	7,940	62 (A, H_M, H_O, O, P)
nsrand-ipx	3,301	18 (H_M)
timtab2	14,059	100,000+ (limit)

Traditional: branch to impact objective function value

New: branch to impact active constraints in current LP-relaxation

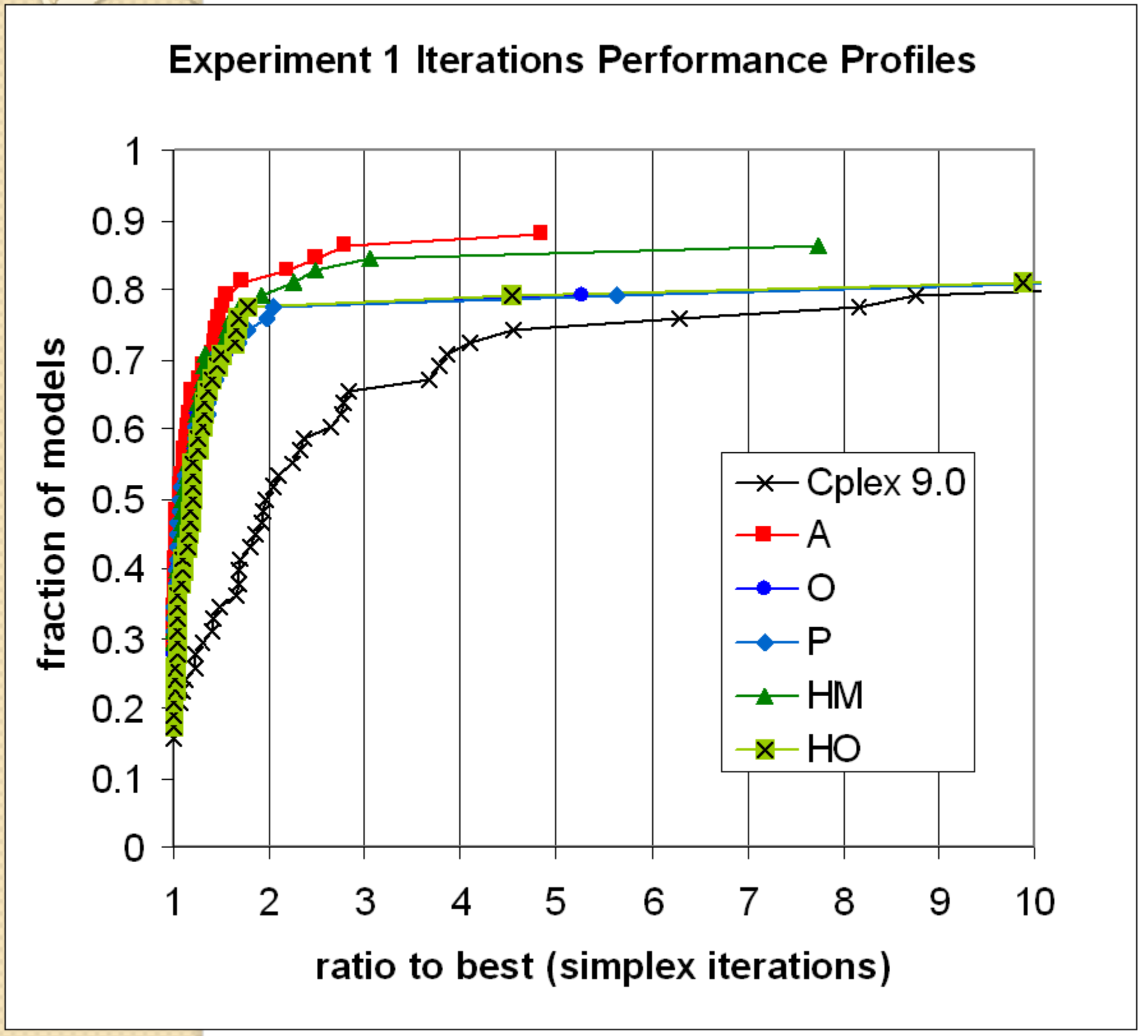
Try to make the child LP-relaxations as different as possible

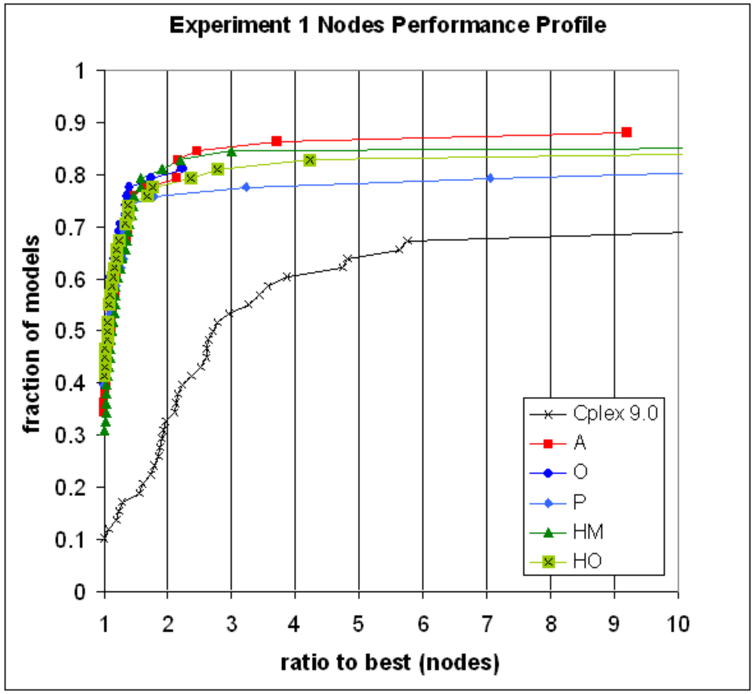


Selecting the branching variable

- Calculate a weight W_{ik} for each candidate i in active constraint k:
- **A:** $W_{ik}=1$.
  - *Is candidate variable present in the active constraint?*
- **M:** $W_{ik} = 1/$(no. *candidate variables*)
  - *Like A, but relative impact of a constraint normalized by number of candidate variables it contains*
- **O:** $W_{ik} = |\text{coeff}_i|/$(no. of *integer variables*)
  - *size of coefficient affects weight of varb in constraint*
- A, O, M: choose k with largest $\sum_i W_{ik}$
- H_M, H_O, etc.: choose k with largest W_{ik}
- Many other methods….

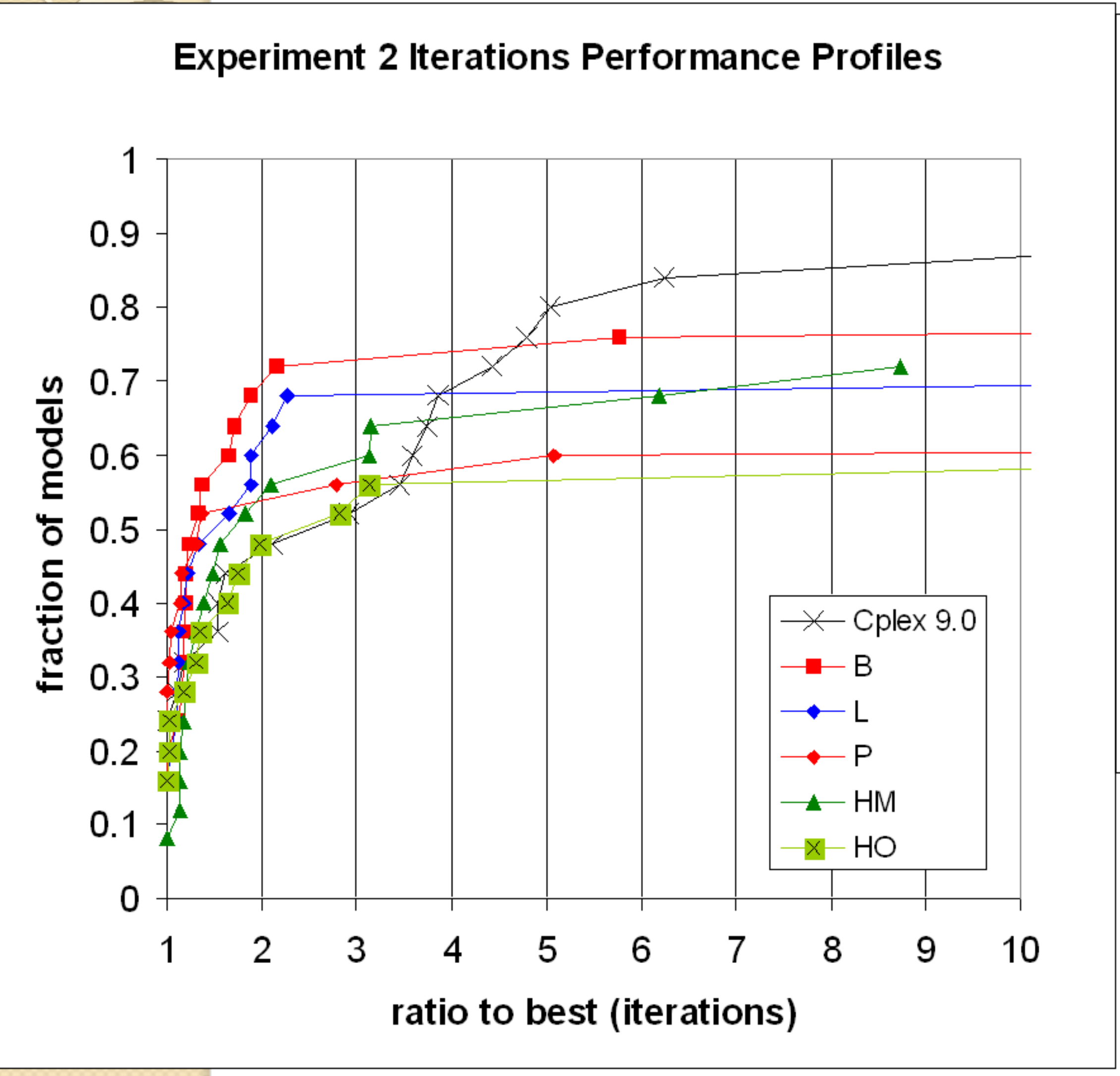
Experiment 1: Cplex heuristics off

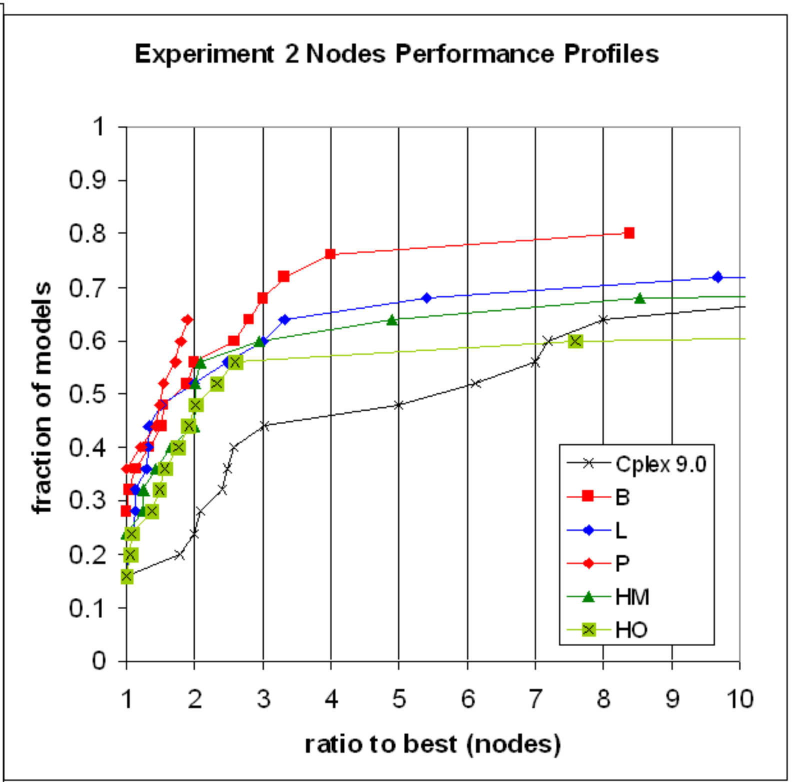




Usually a better optimality gap at first integer-feasible solution (53-78% of models).

Experiment 2: Cplex heuristics on





Only for models not solved at root node.

Usually a smaller optimality gap at first integer-feasible solution

Lessons learned

- It's important to impact the active constraints
  - *Forces many candidates to change values simultaneously*
  - Forces child node solutions to be quite different from each other and from parent

Goal: reaching first integer-feasible solution quickly

BRANCHING TO FORCE CHANGE

Question

- Should you branch so child node has ***largest*** or ***smallest*** probability of a feasible solution?
- Insight from *multiple choice* constraints
  - $x_1 + x_2 + x_3 + \ldots x_n \{\leq,=\}\ 1$, where x_i are binary
    - *Branch down*: x_i can take real values
    - *Branch up*: all x_i forced to integer values
  - E.g.: $x_1 + x_2 + x_3 + x_4 = 1$ at (0.25, 0.25, 0.25, 0.25)
  - Branching on x_1:
    - *Branch down*: (0, 0.333, 0.333, 0.333) or many others
    - *Branch up*: (1, 0, 0, 0) is only solution, and ***all integer***.

A new principle

- Goal: zero candidates (integer feasibility)
- Observations:
  - *Often*: each branching forces roughly 1 candidate variable to integrality
  - *Desirable:* force as many candidates as possible to integrality at each branch
- *Branch to force change in as many candidate variables as possible*
  - Hope that *many* will take integer values

Probability-based branching

Counting solutions (Pesant and Quimper 2008)

- $l \leq \mathbf{cx} \leq u$: l, $\mathbf{c}$, u are integer values, $\mathbf{x}$ integer
- Example: $x_1 + 5x_2 \leq 10$ where $x_1, x_2 \geq 0$

Value of x_2	***Range for x_1***	***Soln count***	***Soln density***
x_2=0	[0,10]	11	11/18 = 0.61
x_2=1	[0,5]	6	6/18 = 0.33
x_2=2	[0]	<u>1</u>	1/18 = 0.06
Total solutions		18	

- Choose x_2 =0 for max prob of satisfaction
- Choose x_2=2 for min prob of satisfaction
- *Which is best?*

New: Generalization

Assume:

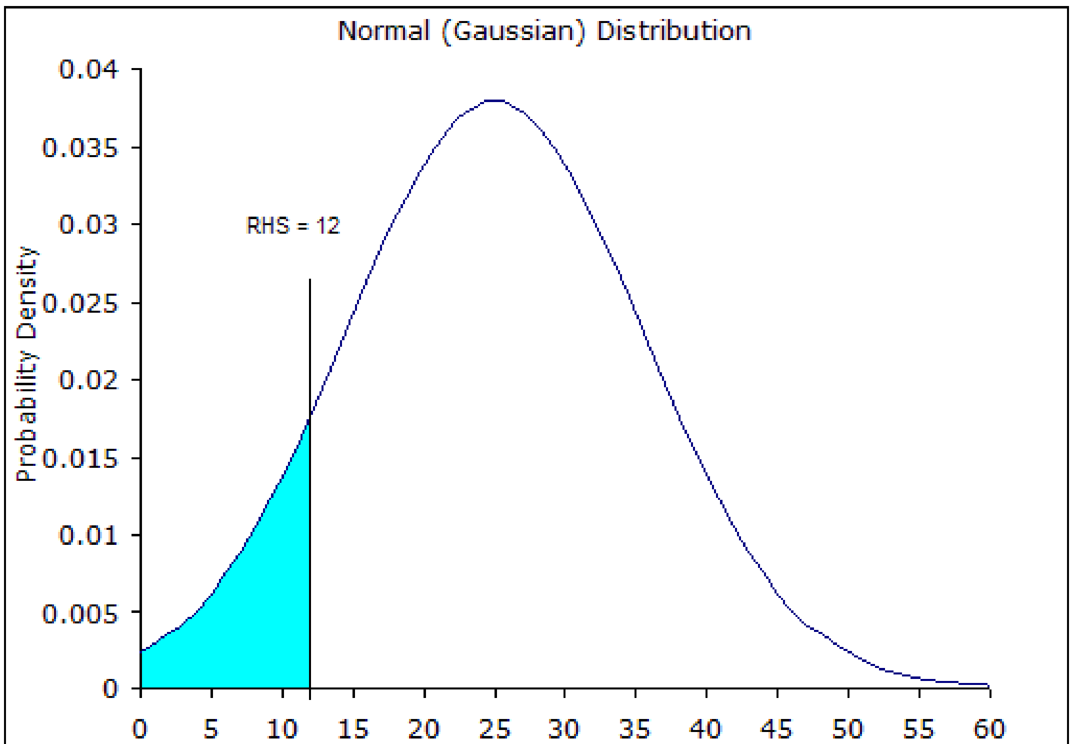
- All variables bounded, real-valued
- Uniform distribution within range

Result:

- linear combination of variables yields *normal distribution* for function value
- Example: $g(\boldsymbol{x}) = 3x_1 + 2x_2 + 5x_3, 0 \leq \boldsymbol{x} \leq 5$ has mean 25, variance 110.83
- *Plot*.... Look at $g(\boldsymbol{x}) \leq 12$

$g(\boldsymbol{x}) = 3x_1 + 2x_2 + 5x_3 \leq 12$ for $0 \leq \boldsymbol{x} \leq 5$

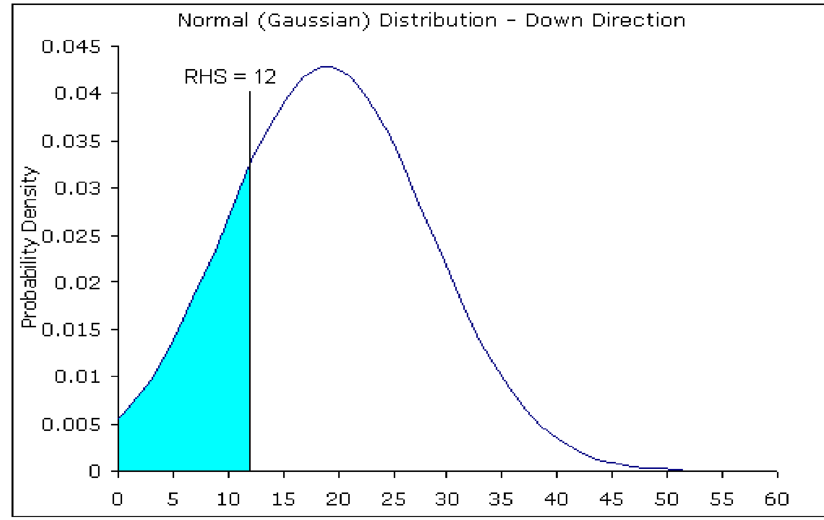
- Probability density plot
  - Cumulative prob of satisfying function in blue

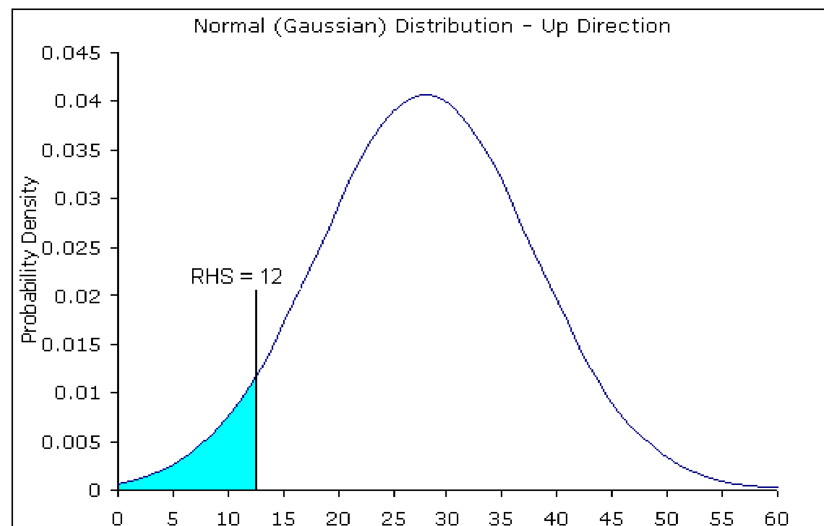


$P[g(\boldsymbol{x}) \leq 12] = 0.1085$

To use for branching:

- Separate distributions for DOWN and UP branches due to changed variable ranges
- Calculate cumulative probability of satisfying constraint in each direction



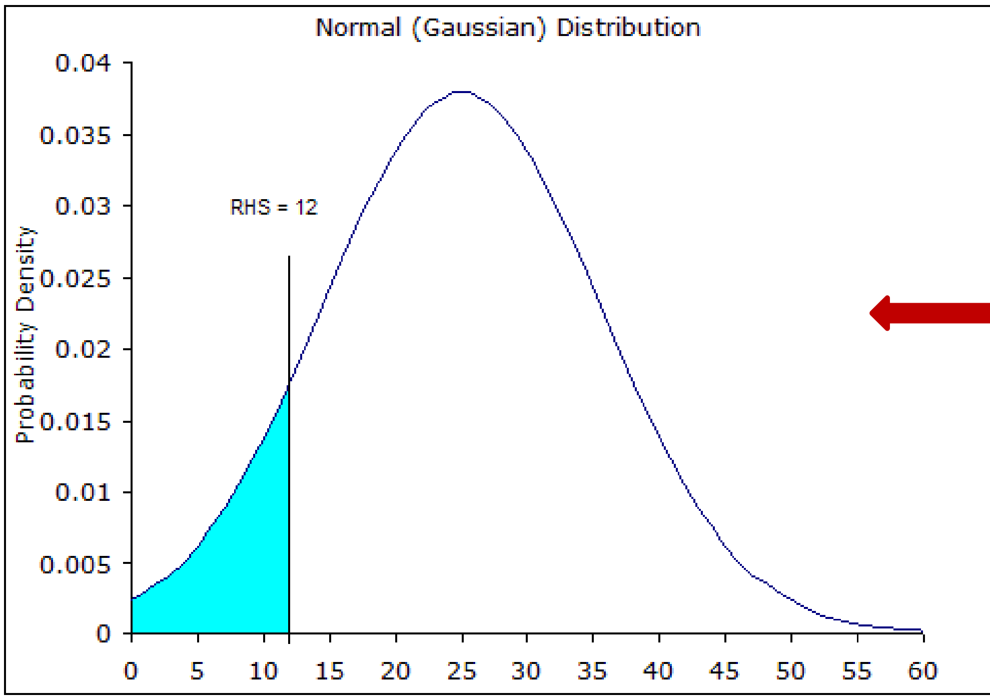


Example:

- Branch on $x_1=1.5$
- *Down*: x_1 range [0,1], p=0.23
- *Up*: x_1 range [2,5], p=0.05

New: handling equality constraints

$g(\boldsymbol{x}) = 3x_1 + 2x_2 + 5x_3 = 12$ for $0 \le \boldsymbol{x} \le 5$



P[g($\boldsymbol{x}$)≤12]=0.1085

P[g($\boldsymbol{x}$)≥12]=0.8915

Equality "probability" = $\frac{\text{(smaller cum. prob)}}{\text{(larger cum. prob)}}$

0.1085/0.8915 = 0.1217

Gives value between 0 and 1.

Larger value means more centred in the distribution, hence larger chance of satisfying the equality

New branching direction methods

Given the branching variable:

- Choose direction based on cum. prob. in *<u>any</u>* active constraint branching variable is in:
  - LCP: Lowest cum. prob. in any active constraint
  - HCP: Highest cum. prob. in any active constraint
- Choose direction based on *votes* using cum. prob. in *<u>all</u>* active constraints branching variable is in:
  - LCPV: direction most often selected based on lowest cum. prob.
  - HCPV: direction most often selected based on highest cum. prob.

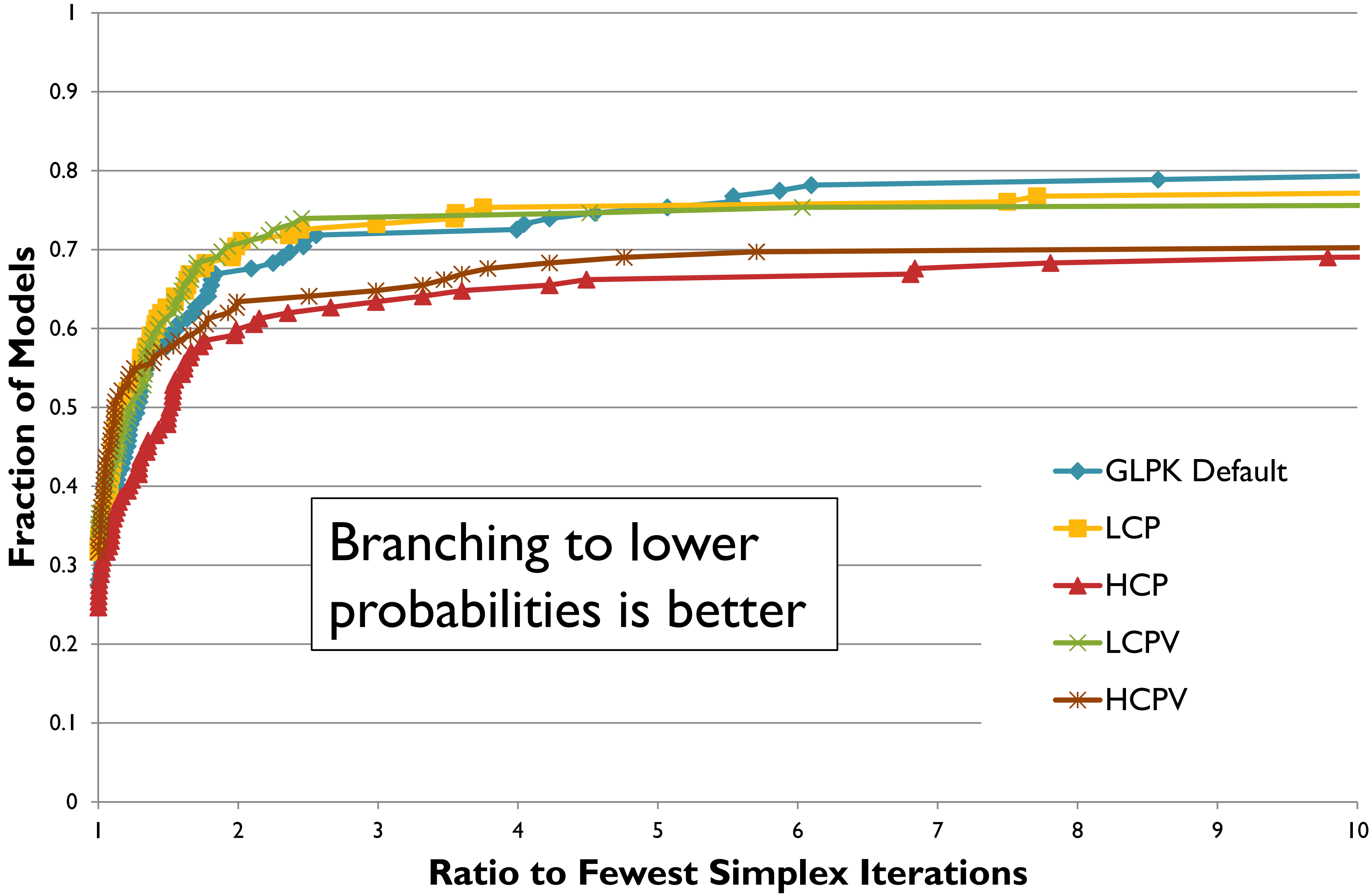
New simultaneous variable and direction selection methods

- VDS-LCP: choose varb *and* direction having lowest cum. prob. among all candidate varbs and all active constraints containing them
- VDS-HCP: choose varb *and* direction having highest cum. prob. among all candidate varbs and all active constraints containing them

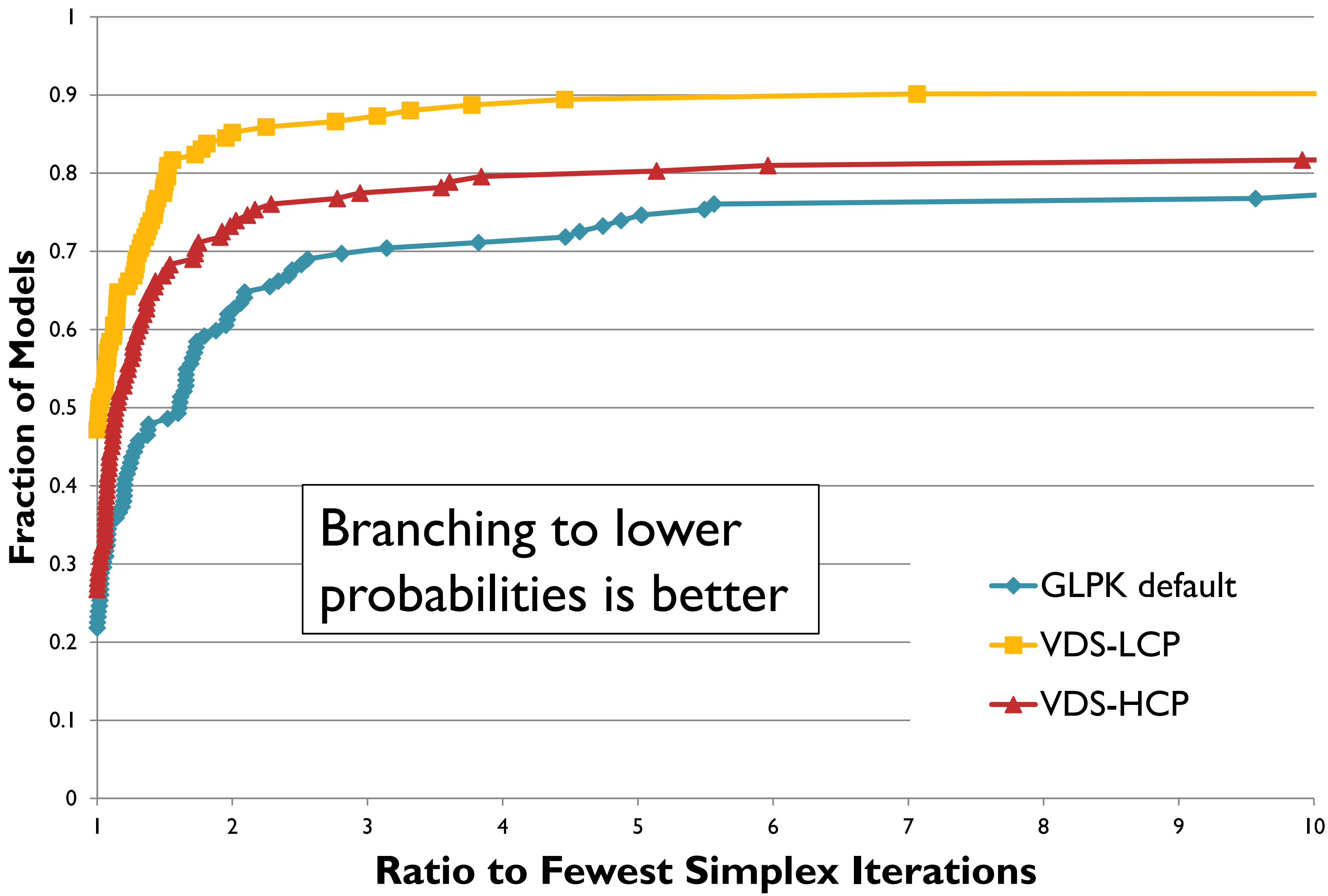
New violation-based methods

- Fix all variables except branching variable. What happens when branching UP vs. DOWN?
  - *Inequality:* is act. constraint violated or still satisfied?
  - *Equality:*
    - "violated": less centred direction
    - "satisfied": more centred direction
- MVV: Most Violated Votes method
  - Choose direction that violates largest number of active constraints containing branching varb.
- MSV: Most Satisfied Votes method

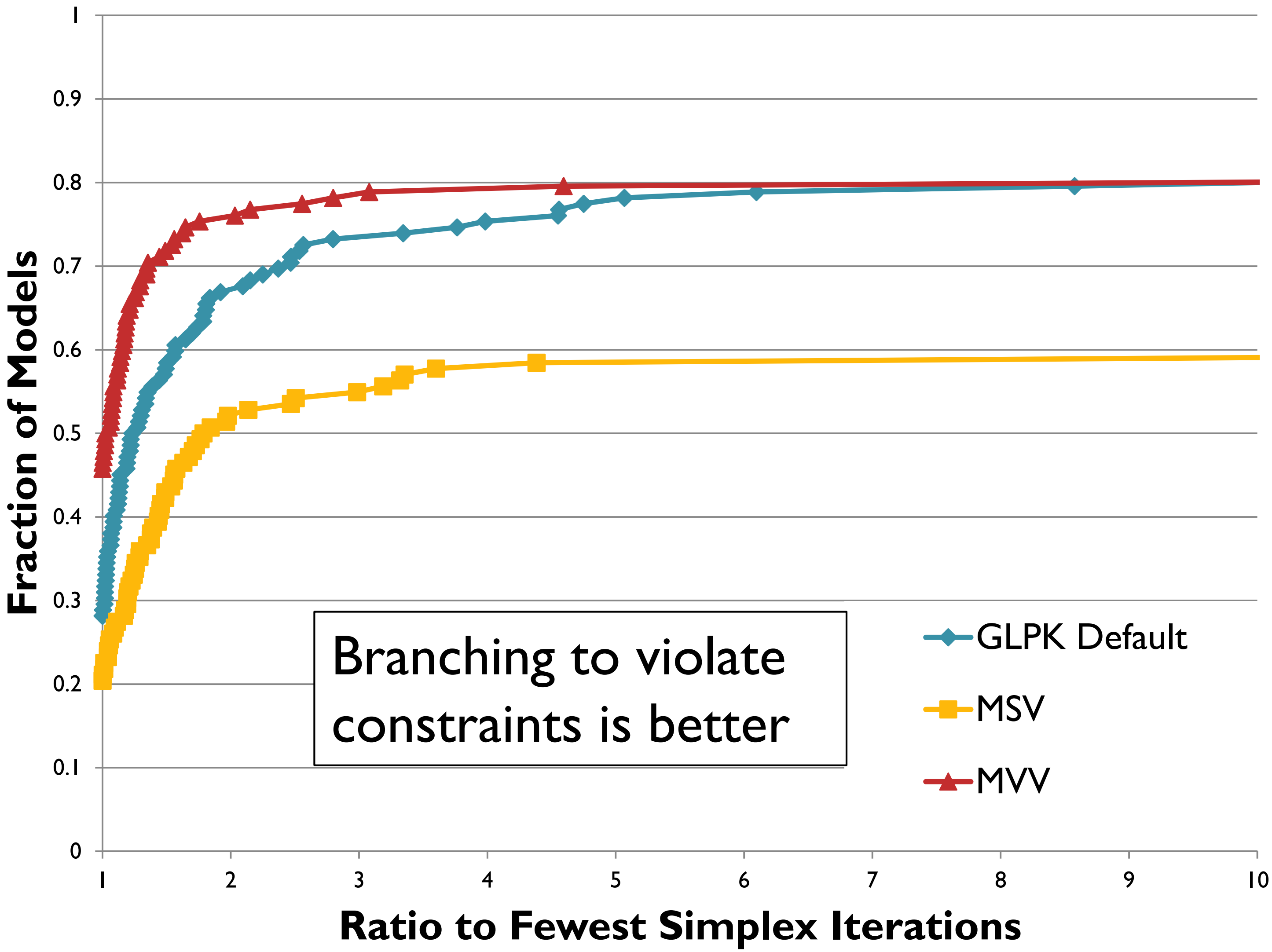
LCP/LCPV vs. HCP/HCPV



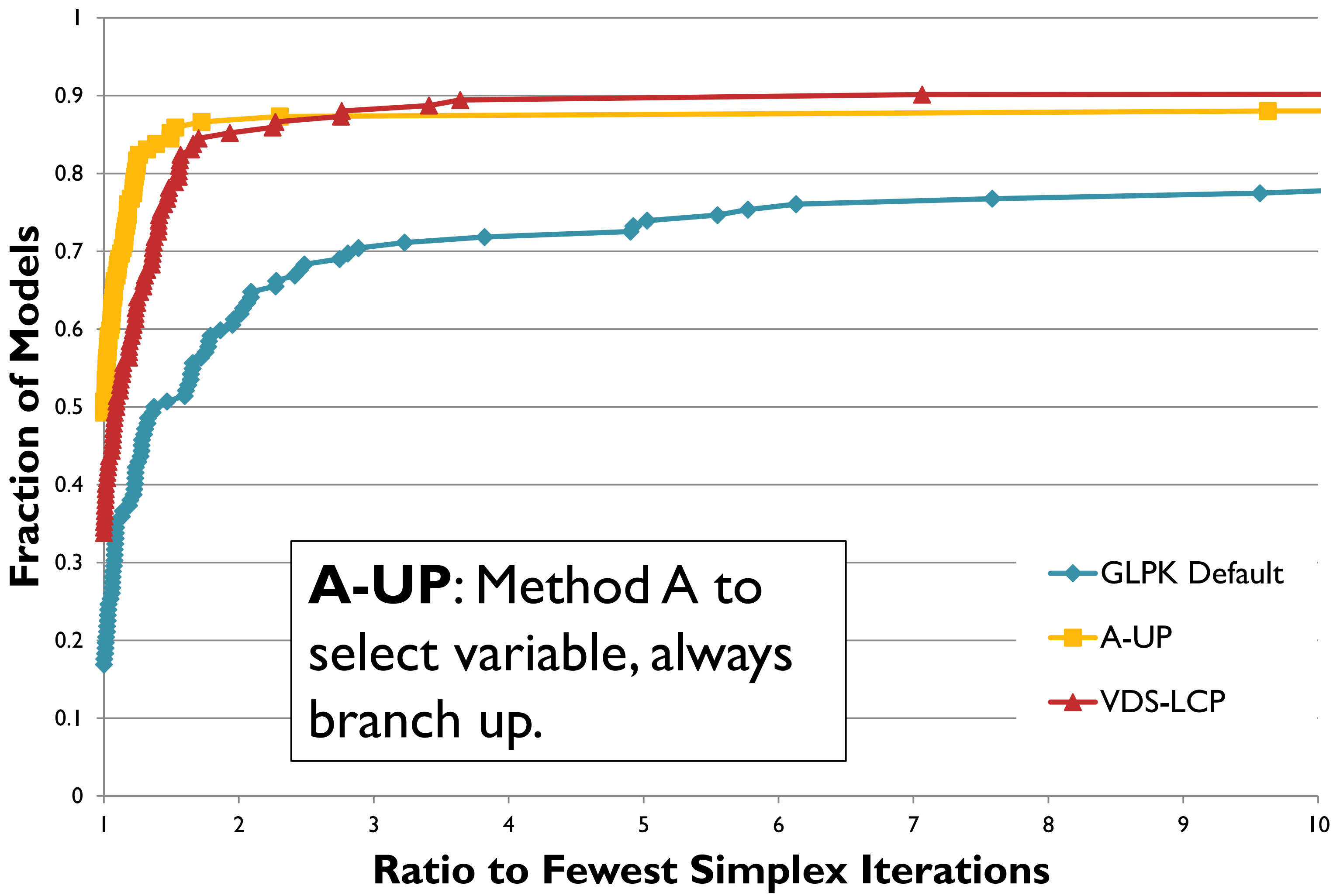
VDS-LCP vs. VDS-HCP



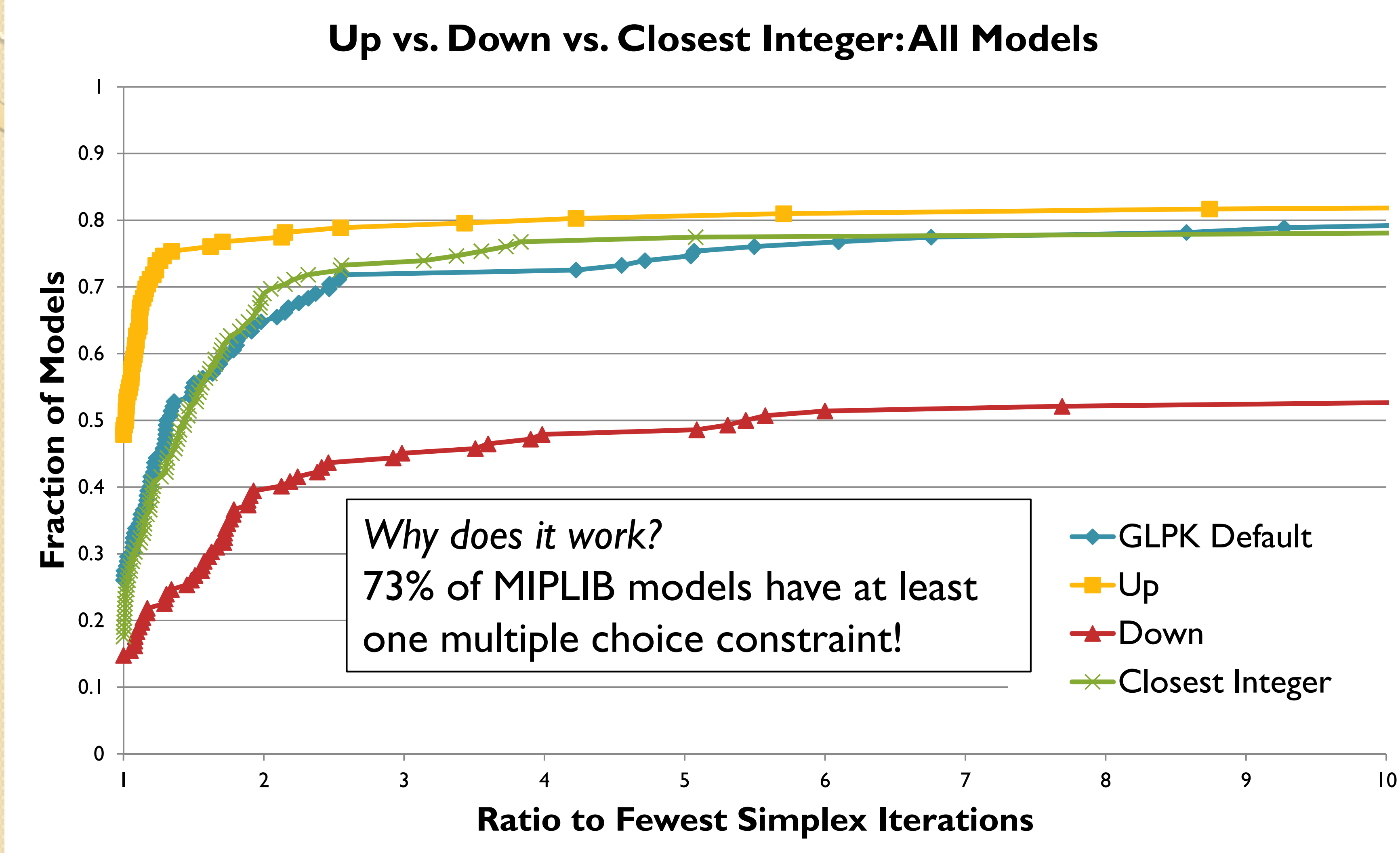
MVV vs. MSV


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Fraction of Models
1
2
3
4
5
6
7
8
9
10
Ratio to Fewest Simplex Iterations
Branching to violate constraints is better
GLPK Default
MSV
MVV

A-UP vs. VDS-LCP



Simple *branch-up* rule is effective



Lessons learned

- Most effective:
  - Branch to low probability variables and directions
  - Branch to violate constraints
  - ***Branch to force change in the candidates***

- Compare:
  - MILP:
    - Constraints always satisfied, varbs not integer
  - Constraint programming:
    - Constraints not satisfied, varbs always integer

Goal: reaching first integer-feasible solution quickly

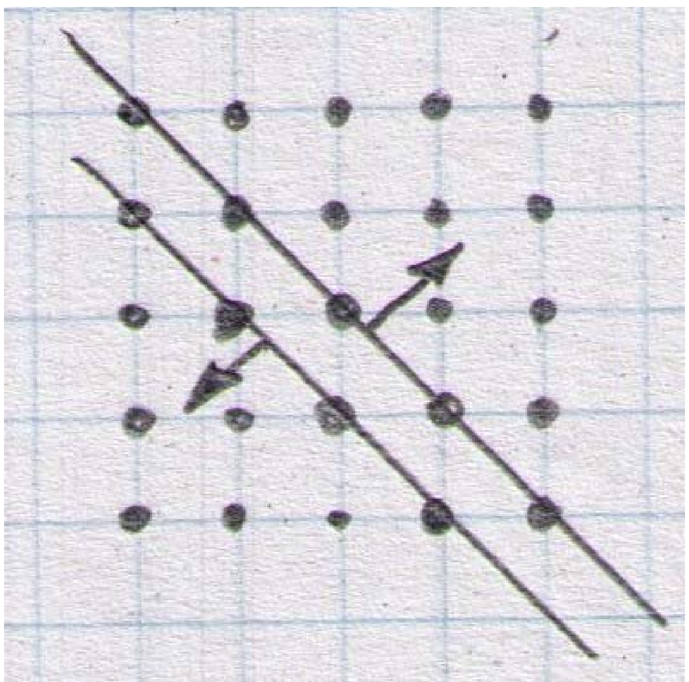
GENERAL DISJUNCTIONS

Beyond branching on variables

- Why not branch on a general linear equation? E.g.:
  - $a_1x_1 + a_2x_2 + a_3x_3 + \ldots + a_nx_n \leq k$
  - $a_1x_1 + a_2x_2 + a_3x_3 + \ldots + a_nx_n \geq k + 1$
  - a_i and k are integers
- Literature:
  - Very hard to find a good general disjunction
    - NP-hard to find *best* general disjunction
  - Usually fewer nodes, but much more time

“45-degree” general disjunctions

- Coefficients are +1, -1, 0
- Run through many lattice points
- Leave an empty interior



Still NP-hard to find best disjunction

New methods: principles

- When to use a general disjunction:
  - *Infrequently*, only when it is beneficial

- Constructing the general disjunction:
  - 45 degree type, based on the active constraint having the most impact on candidate variables
    - <u>Reverse</u> of active constraint variable branching
  - *Branch to force change!*

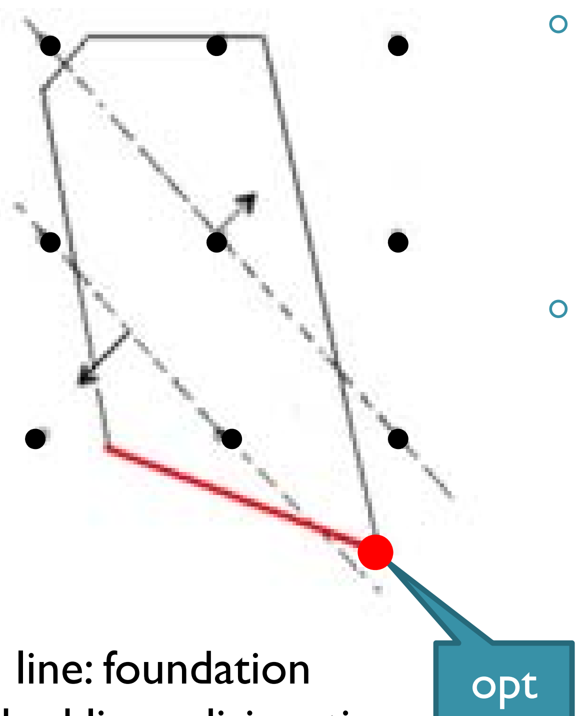
When to insert a general disjunction?

- Only when there are many candidate variables (60+)
  - i.e. large models, high in tree
- Only when axis-parallel branching is stalling:
  - Monitor:
    - number of candidates
    - infeasibility sum
  - ***2A***: both increase 3 times in a row
  - ***2B***: either increase 10 times in a row

Constructing a general disjunction

1. Select active constraint as *foundation*:
   - ***3A***: Choose active constraint having most <u>integer variables</u>. Break ties using highest sum of <u>integer-variable</u> absolute coefficients.
   - ***3B***: Choose active constraint having most <u>candidate variables</u>. Break ties using highest sum of <u>candidate-variable</u> absolute coefficients.

Constructing a general disjunction



Red line: foundation
Dashed lines: disjunction

2. If foundation is an **inequality**:
   - Branching disjunction is *approximately parallel* to foundation: match signs
   - E.g.: $2x_1 - 7x_2 + 15x_3 \leq 30$, where $x_i \geq 0$ and integer. LP-relaxation soln (4.6, 3.2, 2.88)
     - Down branch: $x_1 - x_2 + x_3 \leq \lfloor 4.6\text{-}3.2 + 2.88 \rfloor = 4$
     - Up branch: $x_1 - x_2 + x_3 \geq \lfloor 4.6\text{-}3.2 + 2.88 \rfloor + 1 = 5$

Constructing a general disjunction

3. If foundation is an **equality**:
   - Branching disjunction: *approximately perpendicular* to foundation(*exactly perpendicular* to approx. parallel)
     - No point to approximately parallel: usually no intersection
   - Many ways to construct!
     - E.g.: approx parallel $x_1 - x_2 + x_3$
     - Approx perpend: (1,1,0), (-1,0,1), (0,1,1), etc.
   - Method:
     - Odd no. coeffs: set *least impact* coeff to zero
     - *Least impact*: cont < non-cand int < cand with larger int infeas < larger abs coeff in foundation < varb in more active constraints
     - Switch signs in remaining varbs on even counter

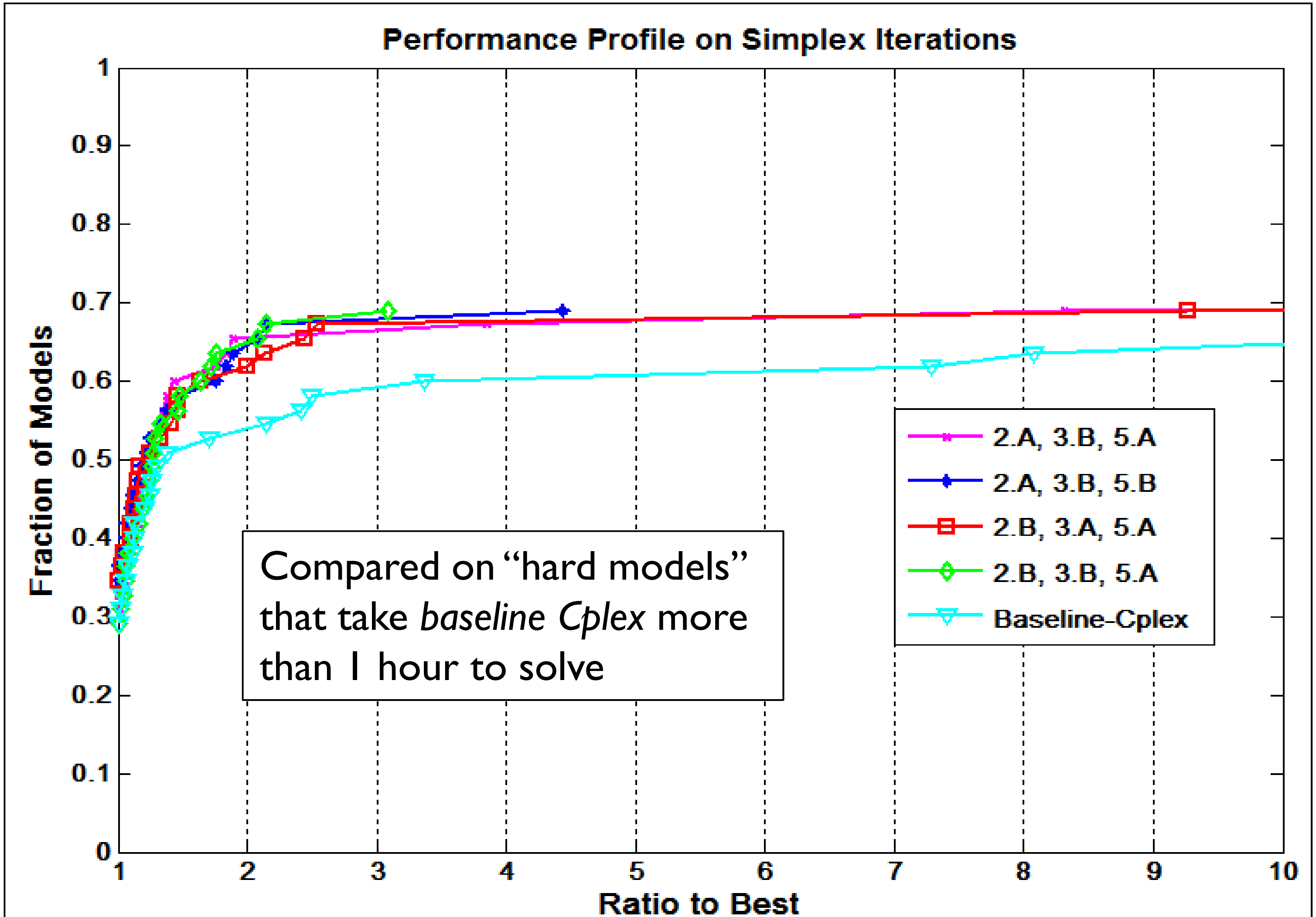
Branching direction

- *Branch to force change*
- Approx parallel disjunctions:
  - **5A**: satisfying direction of inequality, offset by 1
    - Offset in case disjunction lies <u>*on*</u> foundation (e.g. multiple choice foundation)
    - Pushes into feasible region
  - **5B**: farther from LP-relaxation optimum pt
- Approx perpendicular disjunctions:
  - Farther from LP-relaxation optimum

Experimental setup

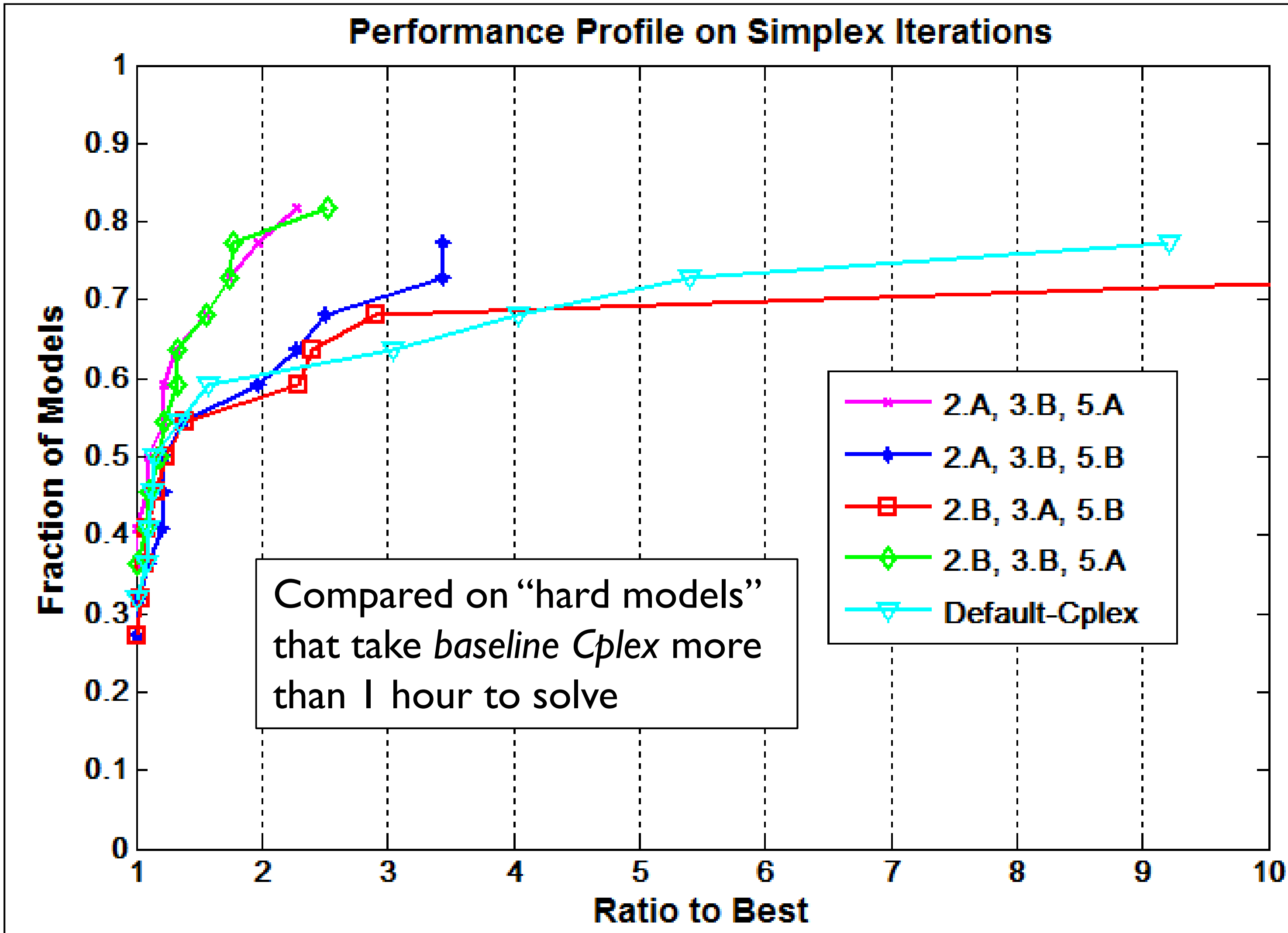
- Built into Cplex 12.1 via callbacks
- **Default Cplex:** all default except:
  - Stop at first integer-feasible solution
  - Emphasize integer-feasibility
  - Depth-first search
  - Time limit 8 hours
  - Single thread
- **Baseline Cplex:** same as default but also:
  - Pre-solve off
  - Aggregation off
  - Internal node heuristics off
  - Cut generation off

Compare to *baseline Cplex* 12.1

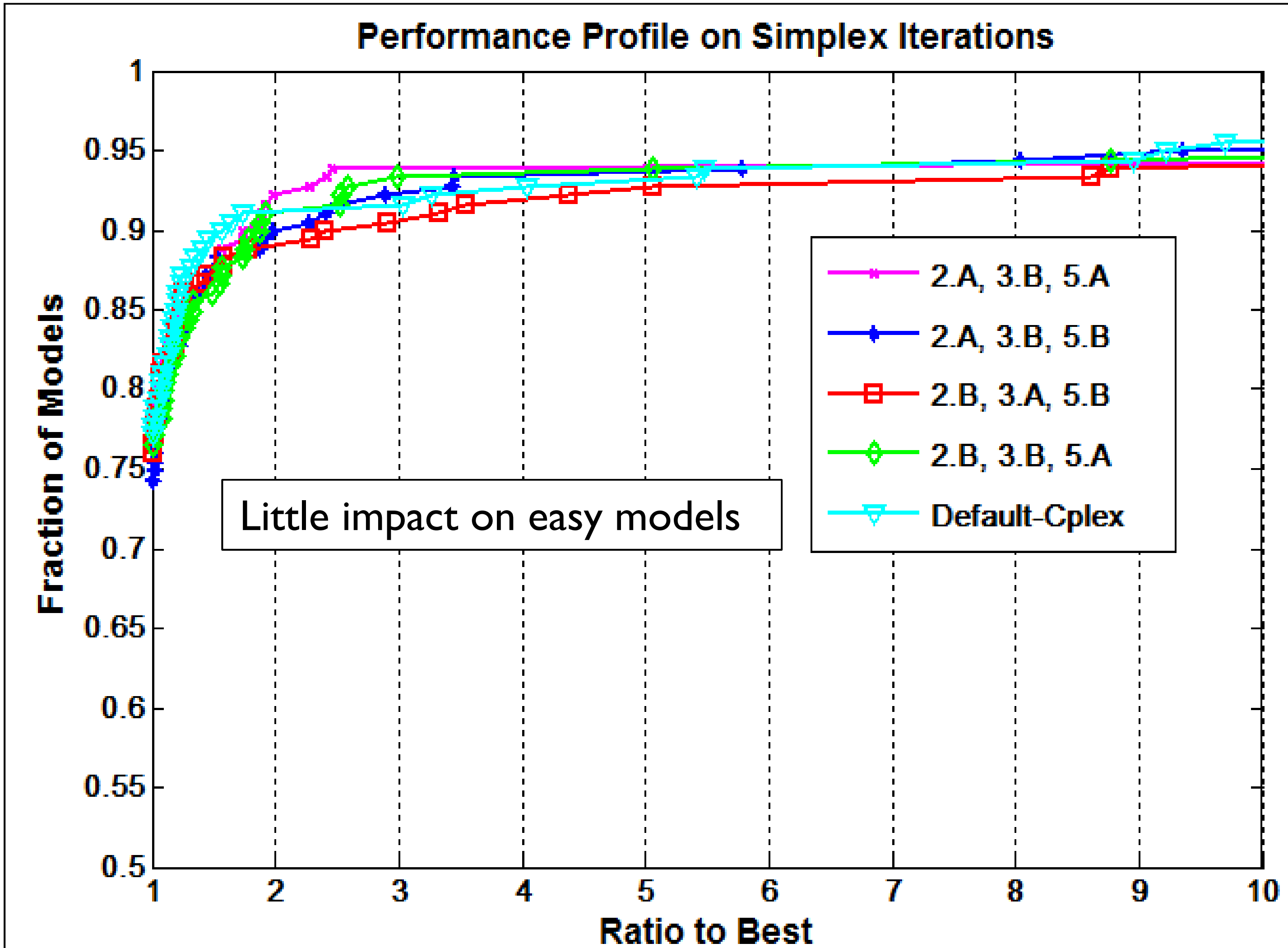


Compared on "hard models" that take *baseline Cplex* more than 1 hour to solve

Compare to *default Cplex* 12.1



Vs. *default Cplex* over **all** models



CONCLUSIONS

Lessons learned

- There are *patterns* in MILP solutions that can be exploited
- Seeking integer-feasibility and seeking optimality are *related*
- Branching should *force change* in the candidate variables
- *General disjunctions* can be helpful

References

- H. Mahmoud and J.W. Chinneck (2012), *Achieving MILP Feasibility Quickly Using General Disjunctions*, in preparation.
- J. Pryor and J.W. Chinneck (2011), *Faster Integer-Feasibility in Mixed-Integer Linear Programs by Branching to Force Change*, Computers and Operations Research, vol. 38, pp.1143–1152.
- D.T. Wojtaszek and J.W. Chinneck (2010), *Faster MIP Solutions via New Node Selection Rules*, Computers and Operations Research, vol. 37, no. 9, pp. 1544-1556.
- J. Patel and J.W. Chinneck (2007), *Active-Constraint Variable Ordering for Faster Feasibility of Mixed Integer Linear Programs*, Mathematical Programming Series A, vol. 110, pp. 445-474.