Goal: reaching first integer-feasible solution quickly

BRANCHING TO FORCE VARIABLE VALUE PROPAGATION IN MILP

John W. Chinneck
Systems and Computer Engineering
Carleton University, Ottawa, Canada

Introduction

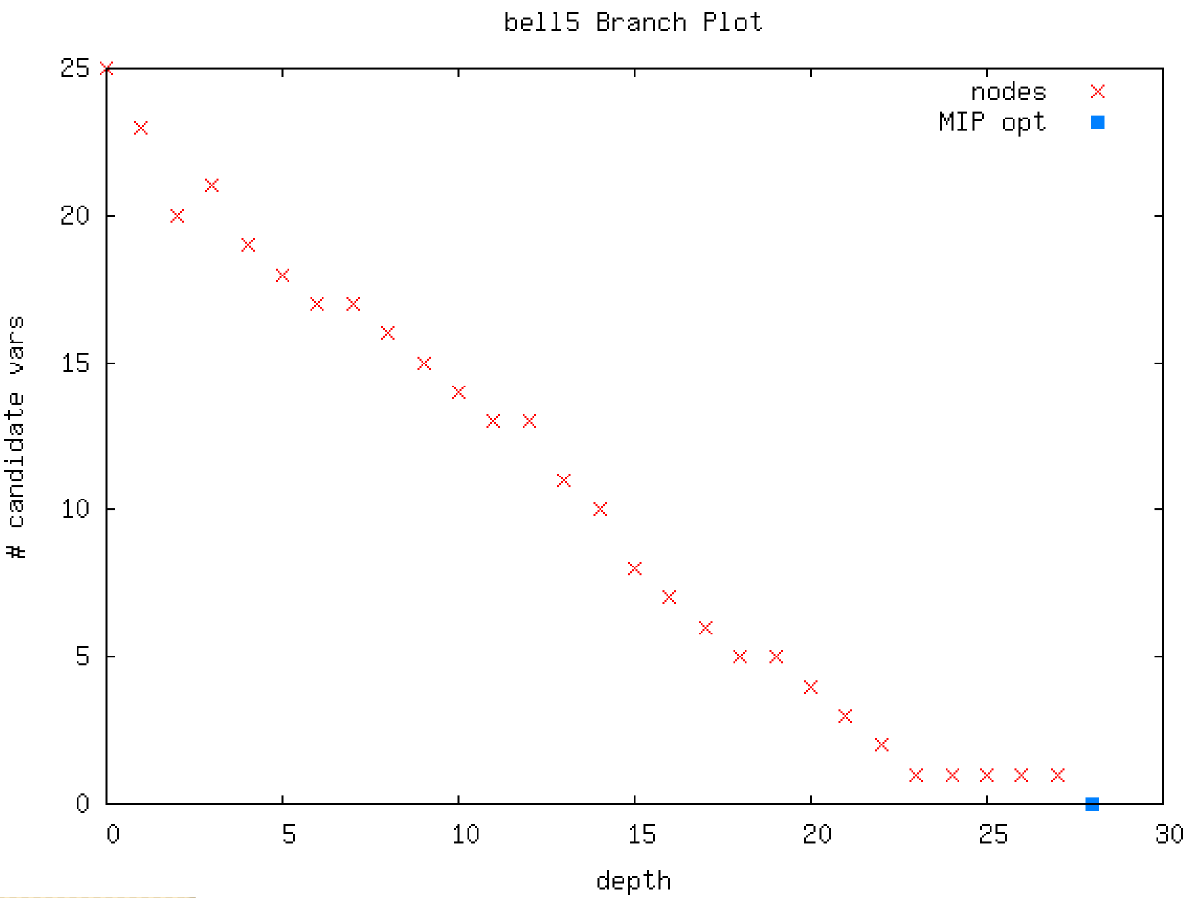
- ***Goal:*** fastest achievement of first integer-feasible solution in MILP.
- ***Question:*** What ***principle*** underlies the best branching heuristics for this goal?
- ***Intuition:*** branch towards the largest number of feasible solutions.
  - *But is this correct?*

A hint

- 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***.

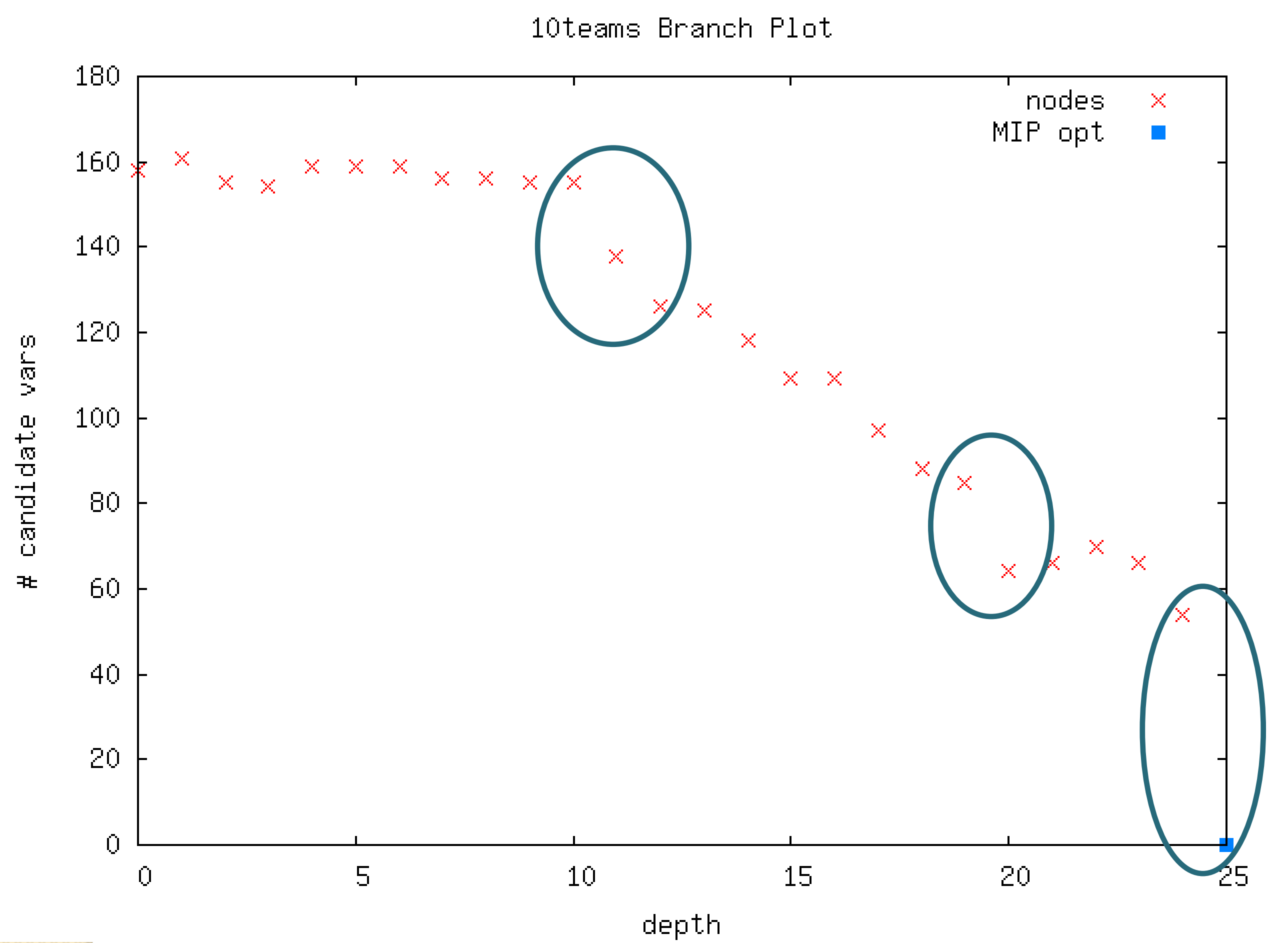
Frequent pattern

Candidate branching variable:
integer/binary, but has fractional value in current LP relaxation solution.
Zero candidate variables = integer feasibility



- Each branch forces about 1 candidate variable to integrality
- Integer feasibility reached when number of candidates is zero
- 25 candidates to 0 candidates in 25 branches

A better pattern

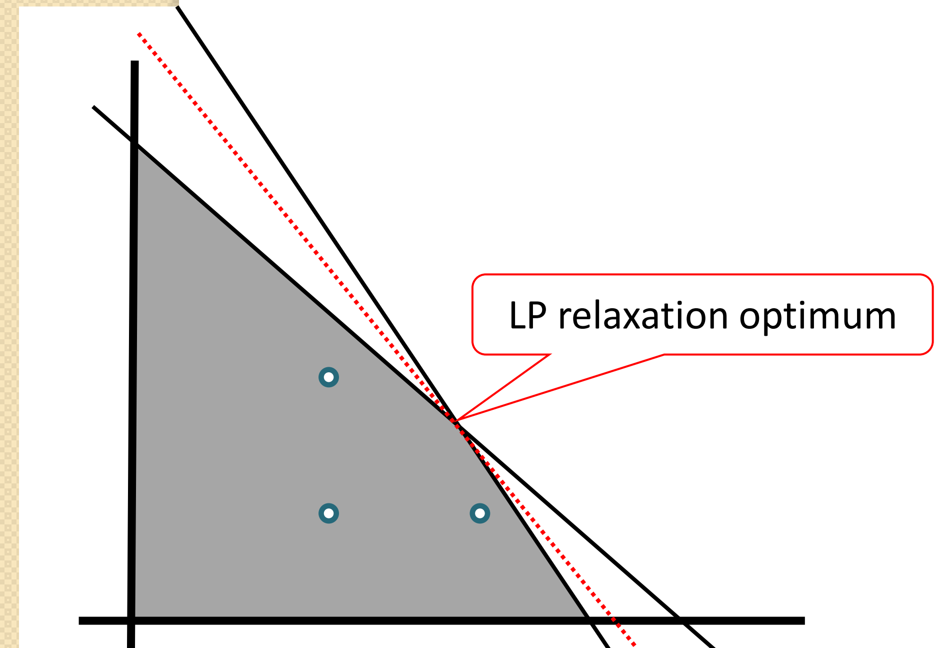


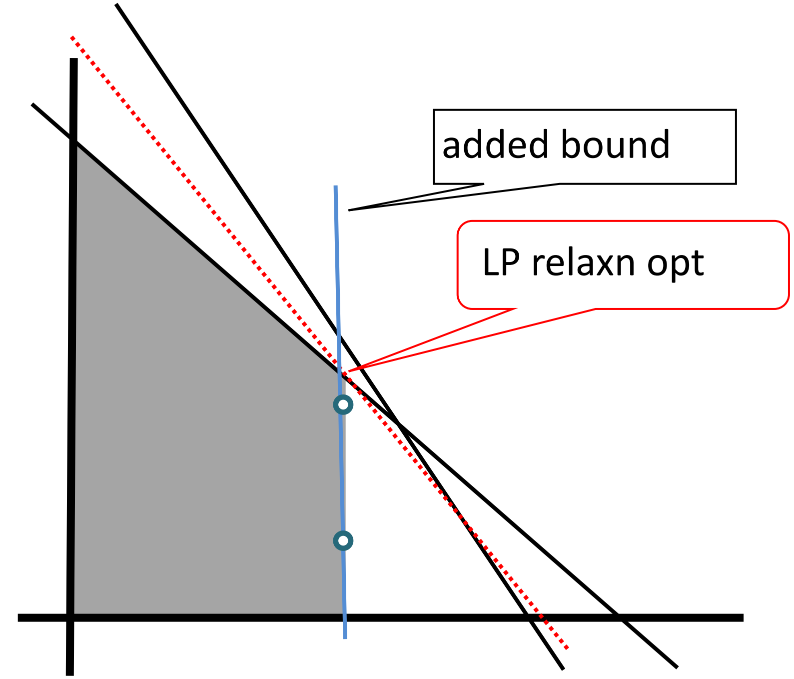
Goal: Force many candidates to integrality at each branch

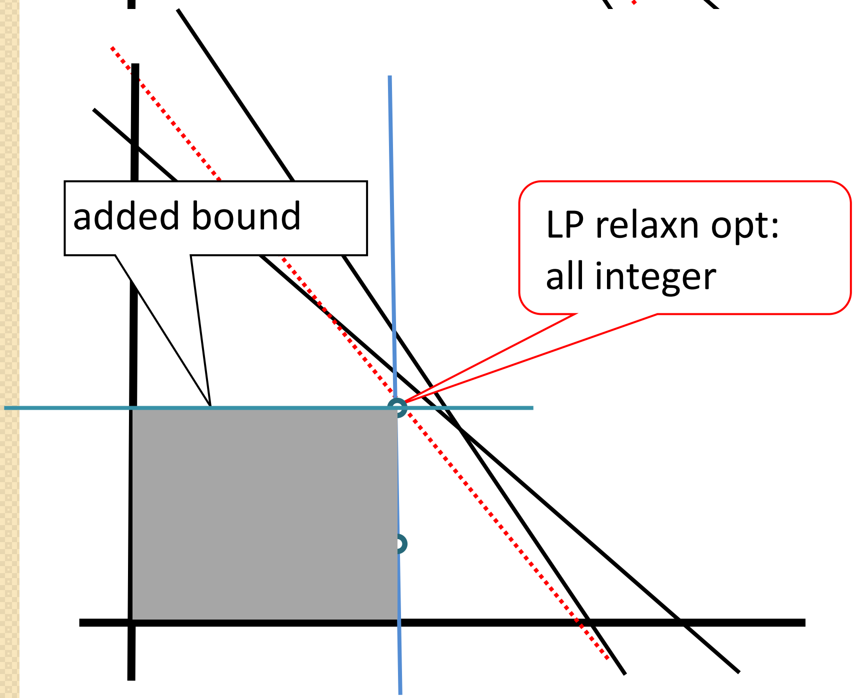
160 candidates to 0 candidates in 25 branches

A new principle

- ***Goal:*** force *many* candidates to integrality at each branch
- ***How?***
  - Branch to force *many* candidate variables to change value
  - Some are forced onto "squared-off" polytope vertices and take integer values
  - Hope that <u>*many*</u> will take integer values







Why it Works

Feasible region is "squared-off" as you dive into tree.

Propagation forces variables onto integer "corners".

Hypothesis

- Branching to maximize probability of a feasible solution does ***not*** force propagation of changed variable values
  - Branching to minimize probability ***does***
- Testing:
  - Develop branching methods where probability of satisfying an individual *active constraint* can be calculated
    - Active constraints determine solution point
  - Test branching to *max* vs. *min* probability of satisfying active constraints

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]	1	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?*
  - x_2=2 forces total integrality

New: Generalization

Assume:

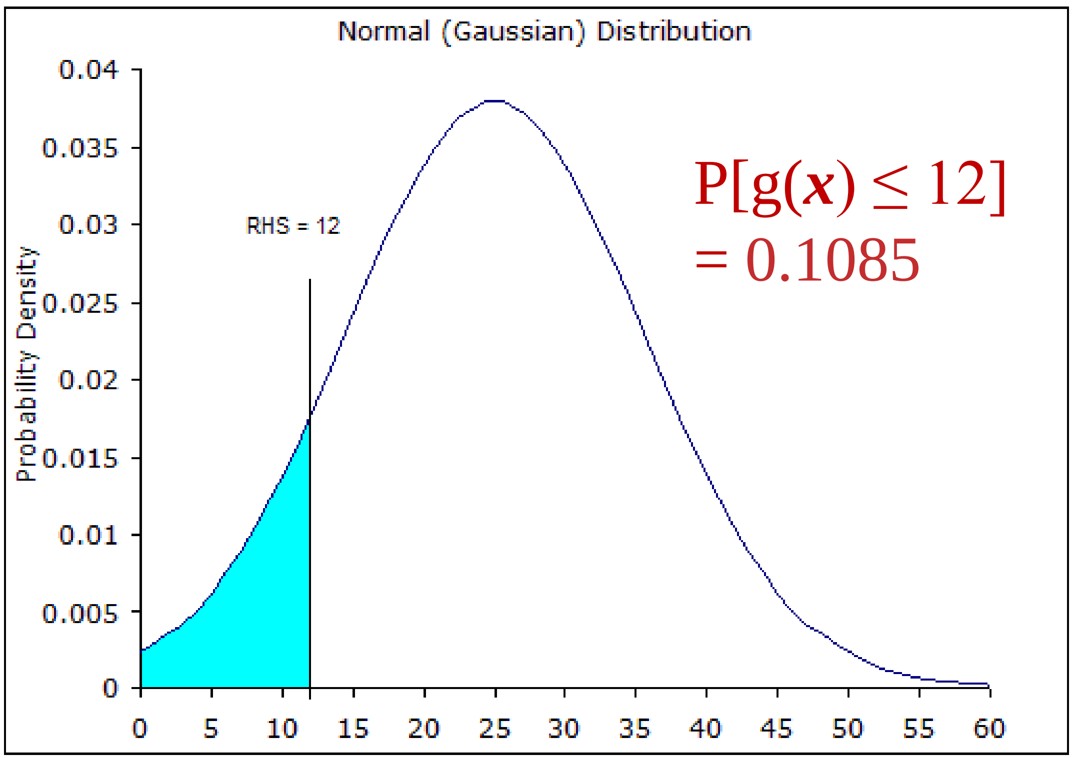
- All variables bounded, real-valued
- Uniform distribution within range

Result:

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

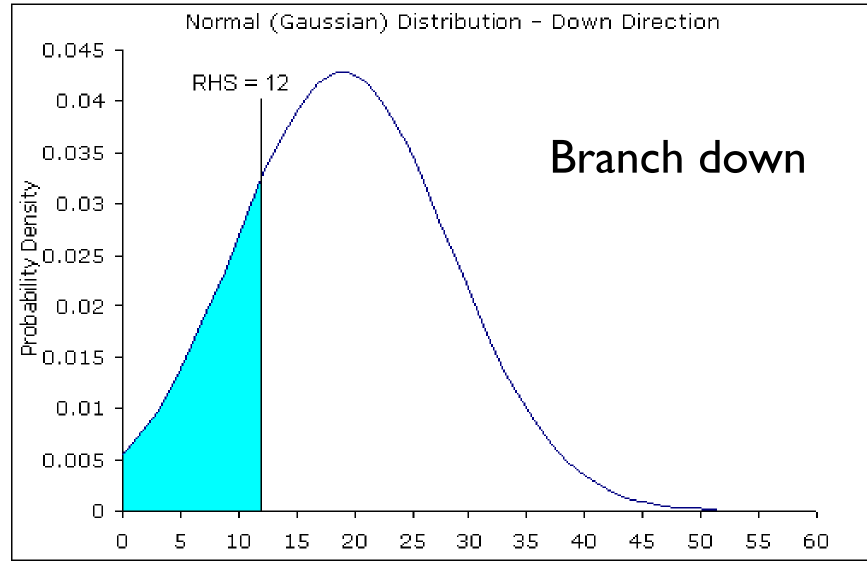
$$g(\boldsymbol{x}) = 3x_1 + 2x_2 + 5x_3 \leq 12 \text{ for } 0 \leq \boldsymbol{x} \leq 5$$

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



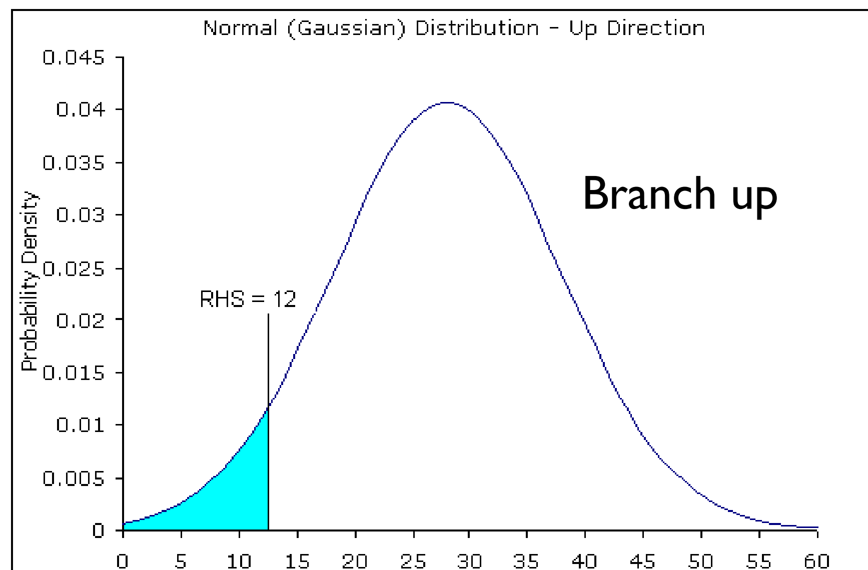
To use for branching:

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



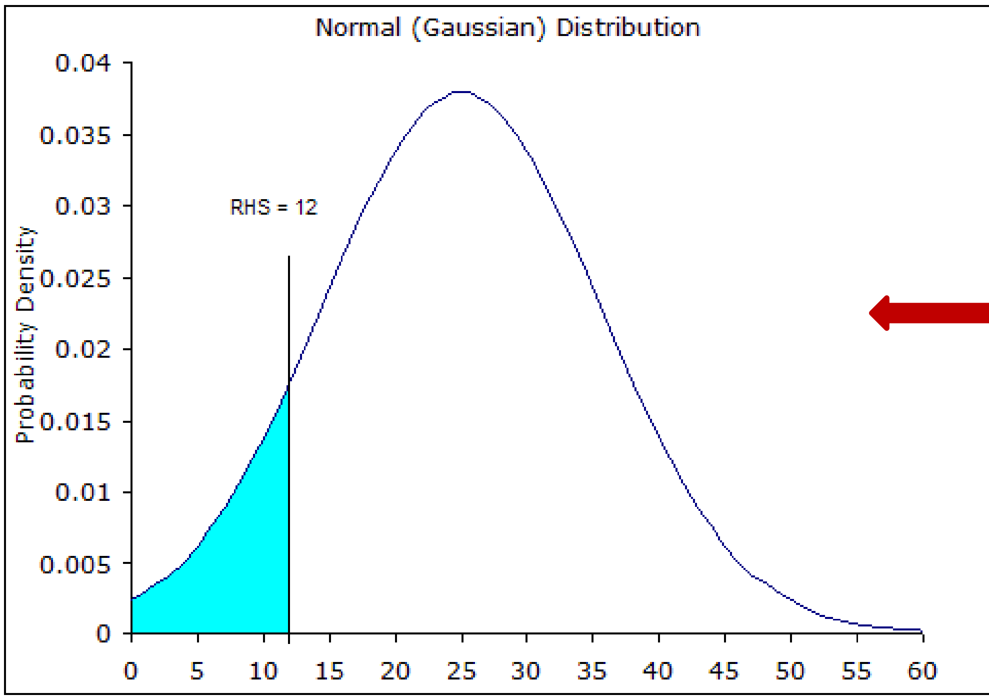
Example:

- 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

$$e.g.\;\; g(\mathbf{x}) = 3x_1 + 2x_2 + 5x_3 = 12 \text{ for } 0 \le \mathbf{x} \le 5$$



$P[g(\mathbf{x}) \le 12] = 0.1085$

$P[g(\mathbf{x}) \ge 12] = 0.8915$

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

$0.1085/0.8915 = 0.1217$

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 *any* 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 *all* 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.

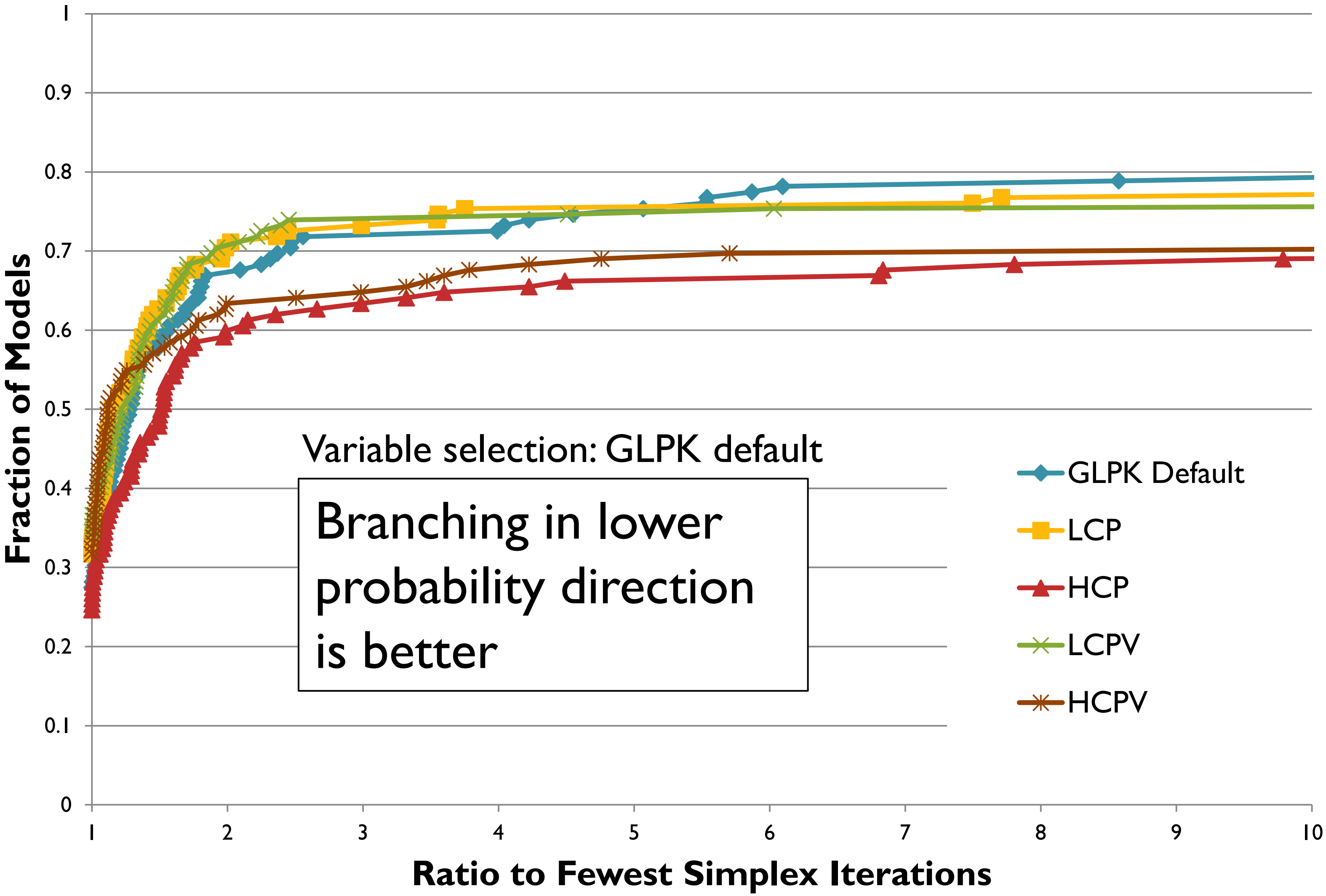
New simultaneous variable and direction selection methods

- 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

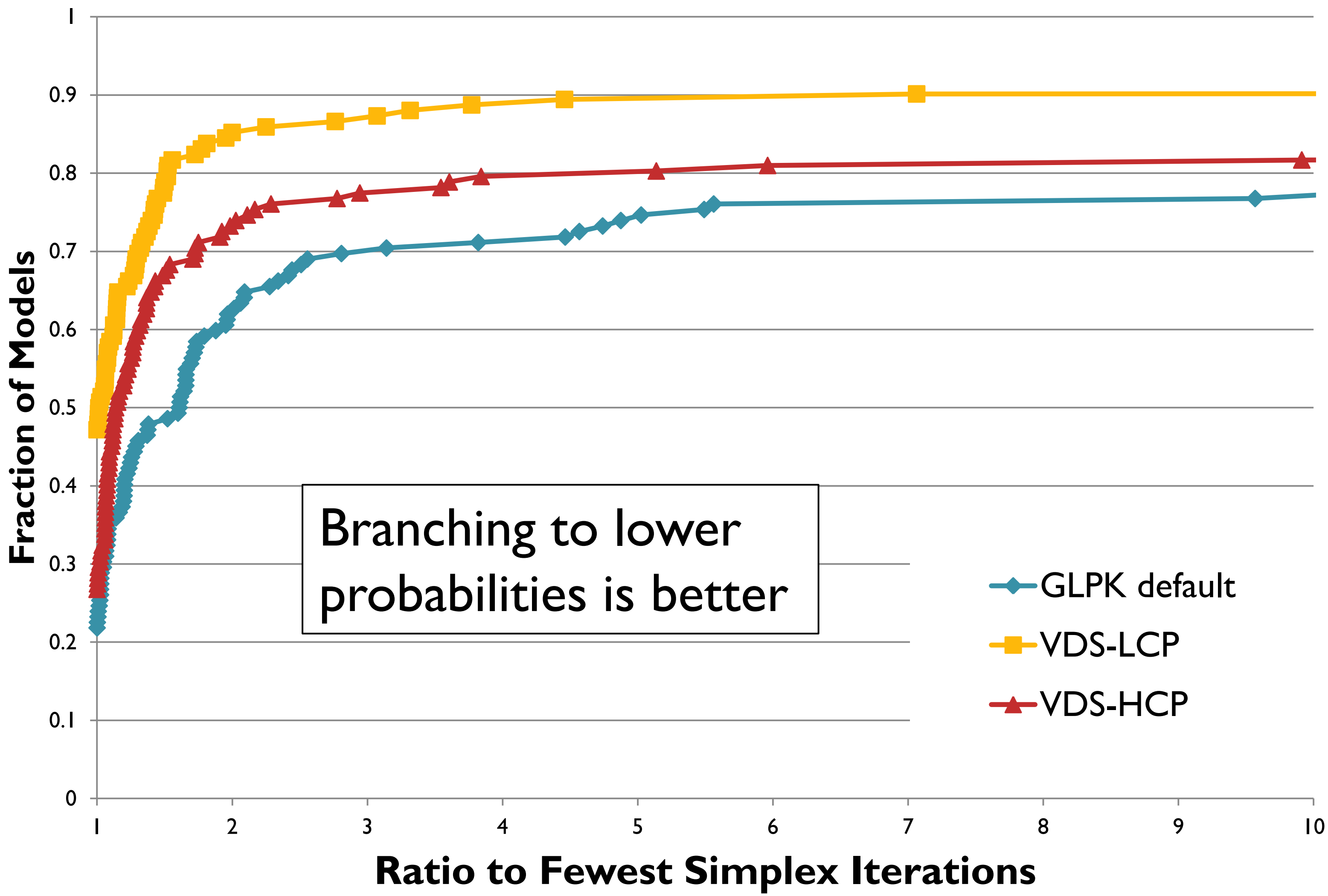
Experimental Setup

- Modified GLPK 4.28
- Stopping: first feasible solution, or two hours
- Node selection:
  - Driebeek and Tomlin (GLPK default), or
  - Depth first (best for first feasibility)
- Variable selection:
  - GLPK default (unless otherwise noted)
- Test models
  - 142 total, 47 equality-containing, 95 equality-free
  - 56 from MIPLIB2003
  - 11 from MIPLIB 3.0
  - 7 from MIPLIB 2.0
  - 68 from COR@L
- Speed metric: number of simplex iterations
  - Due to variety of machines

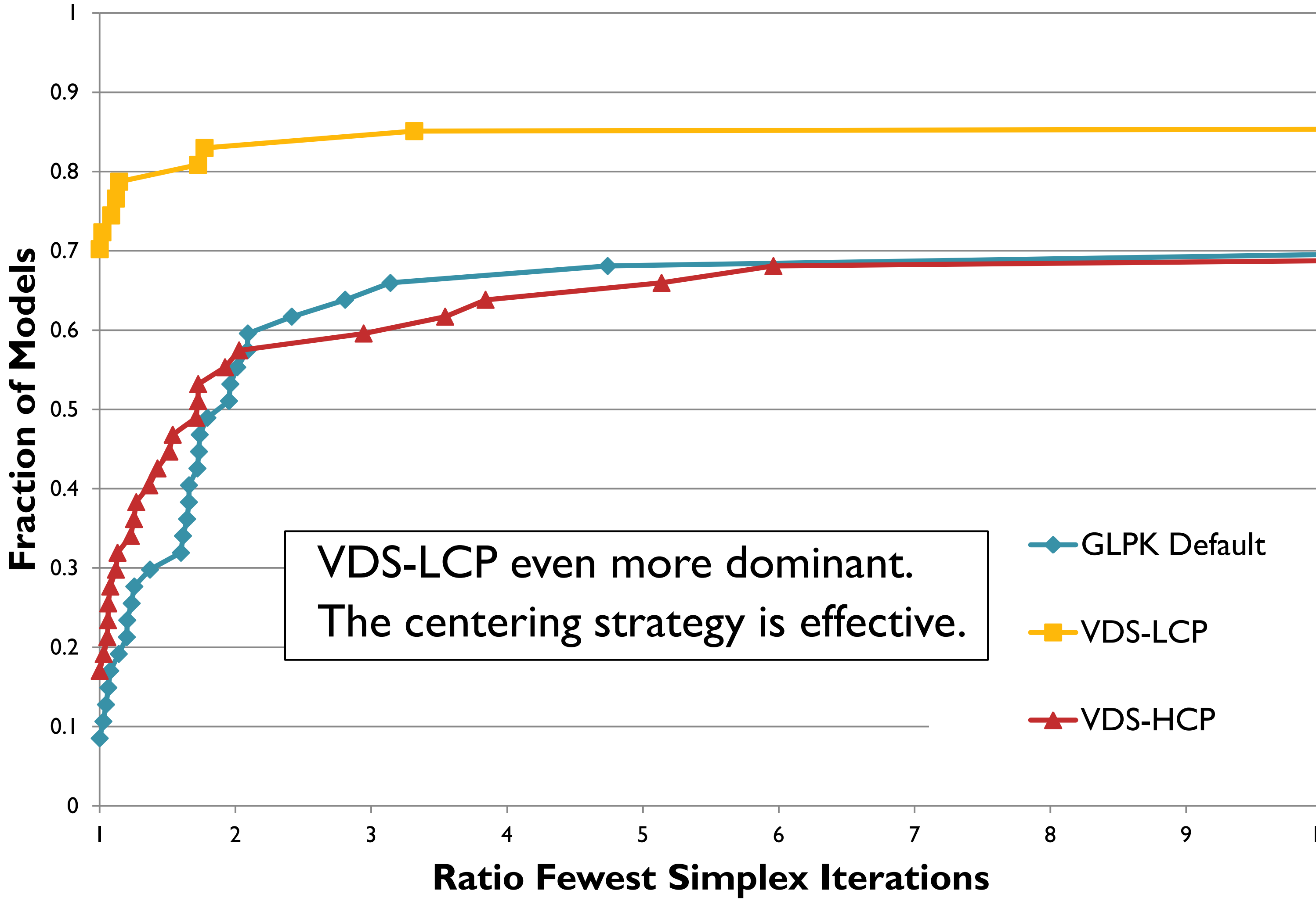
LCP vs. HCP; LCPV vs. HCPV (all models)



VDS-LCP vs. VDS-HCP (all models)



VDS-LCP vs. VDS-HCP (at least one equality constraint)



Lessons learned thus far

- *Low probability* branching directions and *low probability* variables are more effective
  - ***These force change in the candidate variable values***
  - ... causing propagation of the variable values
- It's better to choose *both* variable *and* direction based on low probability
  - Using a different criterion to choose the variable first is not as effective

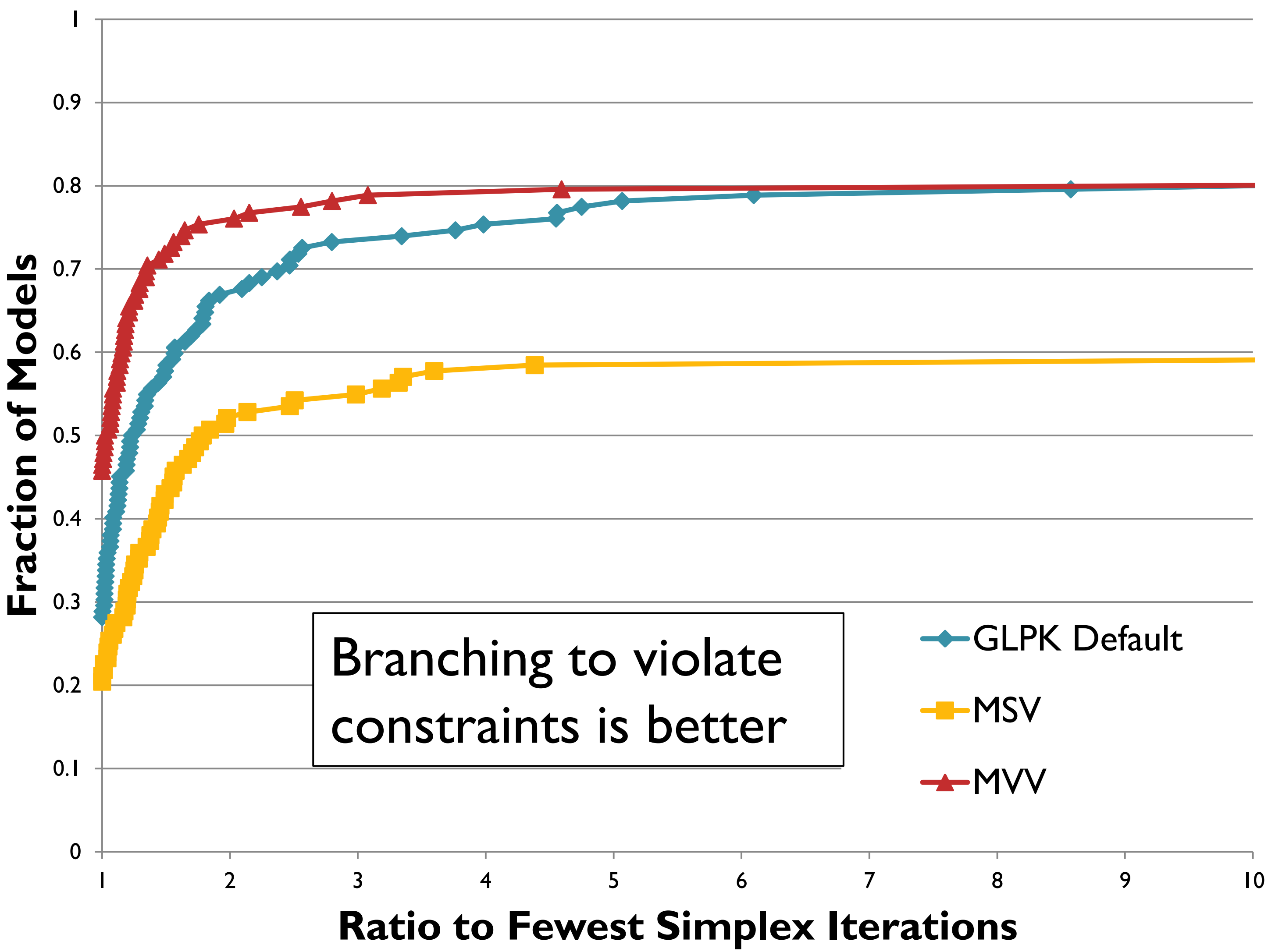
MORE EVIDENCE

1. Violation-based branching
2. Branching up
3. Active constraint based branching
4. More on multiple-choice constraints

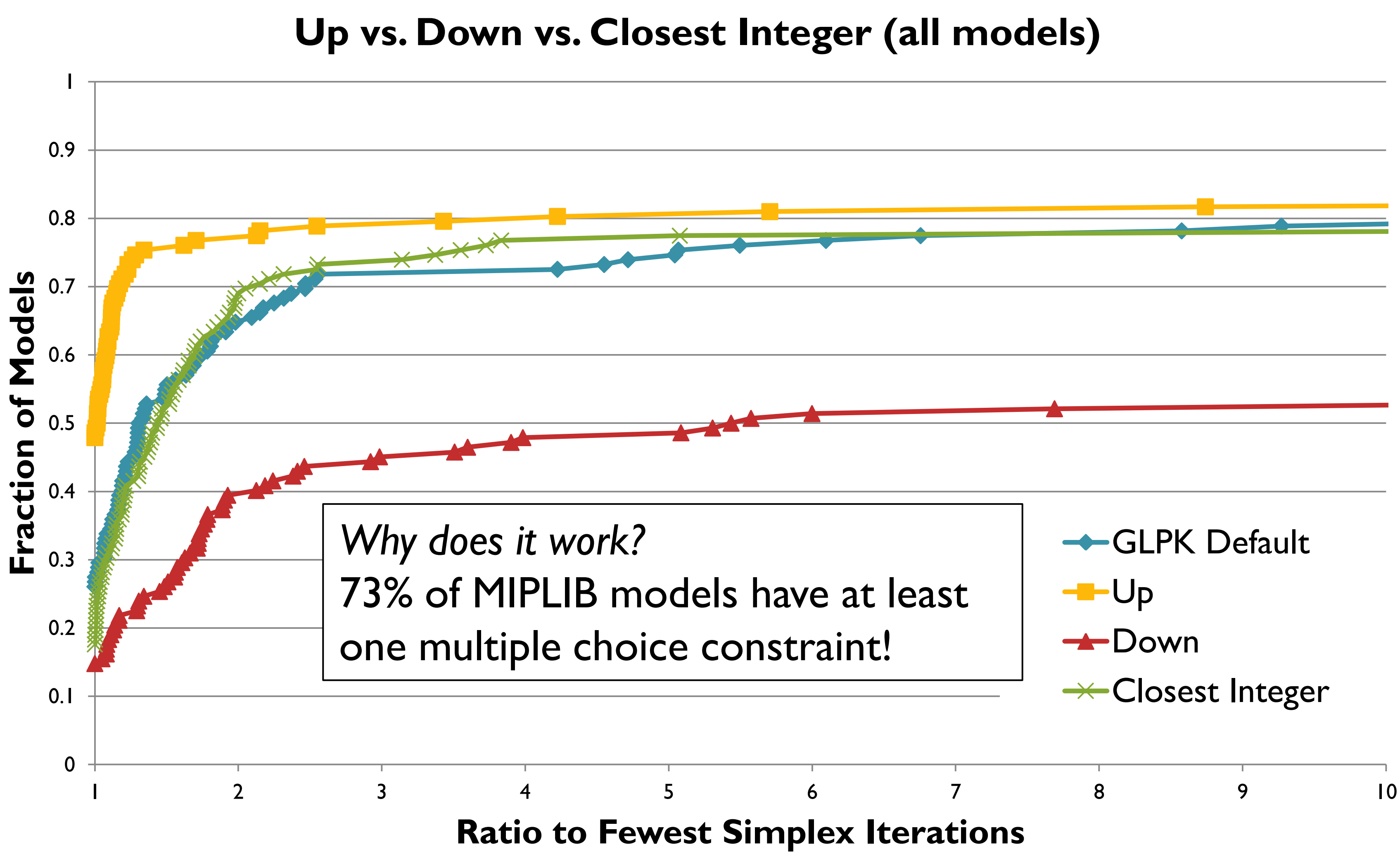
1. ***New*** violation-based methods

- Fix all variables except branching variable. What happens when branching UP vs. DOWN?
  - *Inequality:* active 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

MVV vs. MSV (all models)



2. Simple *branch-up* rule is effective

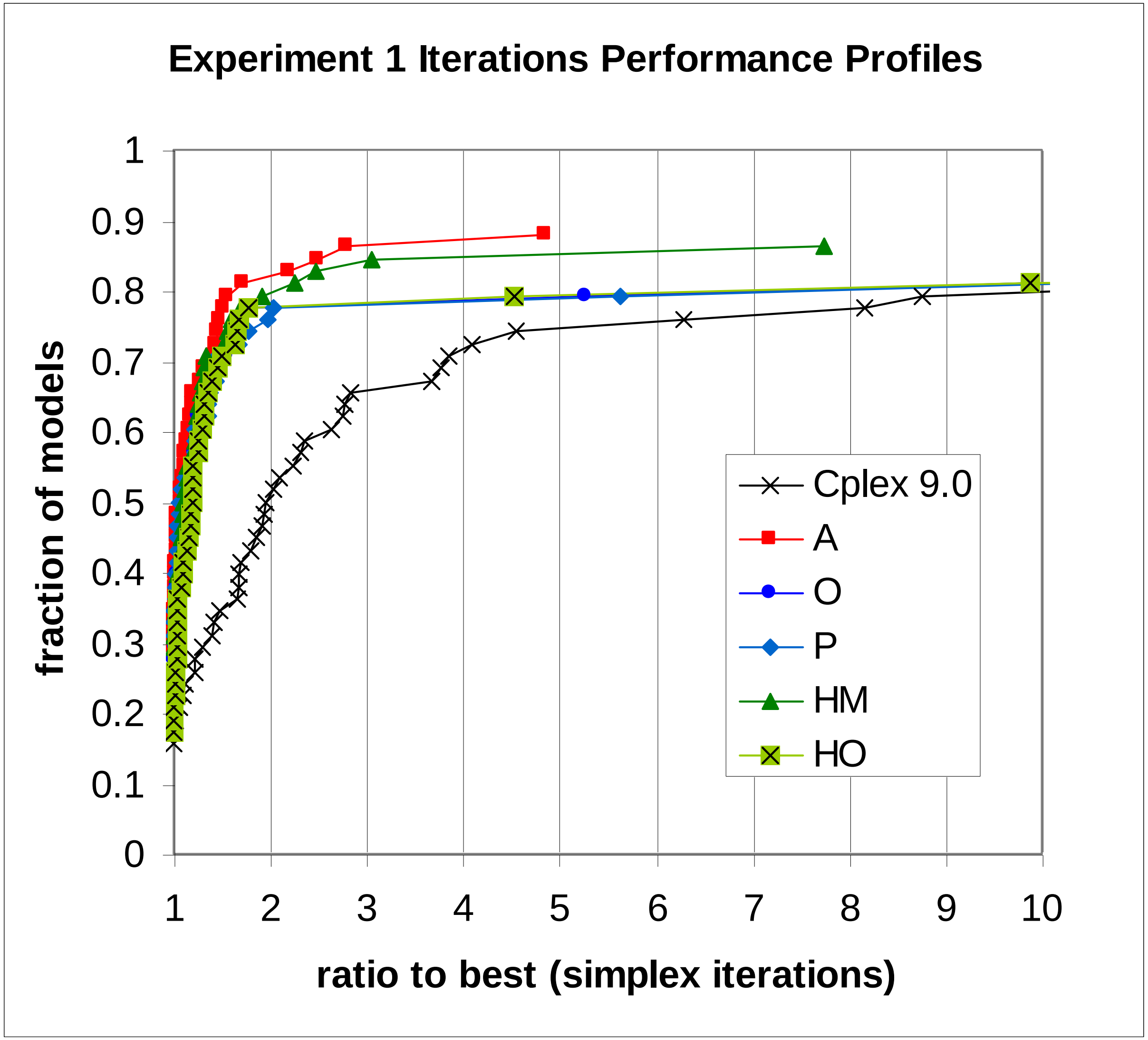


Why does it work?
73% of MIPLIB models have at least one multiple choice constraint!

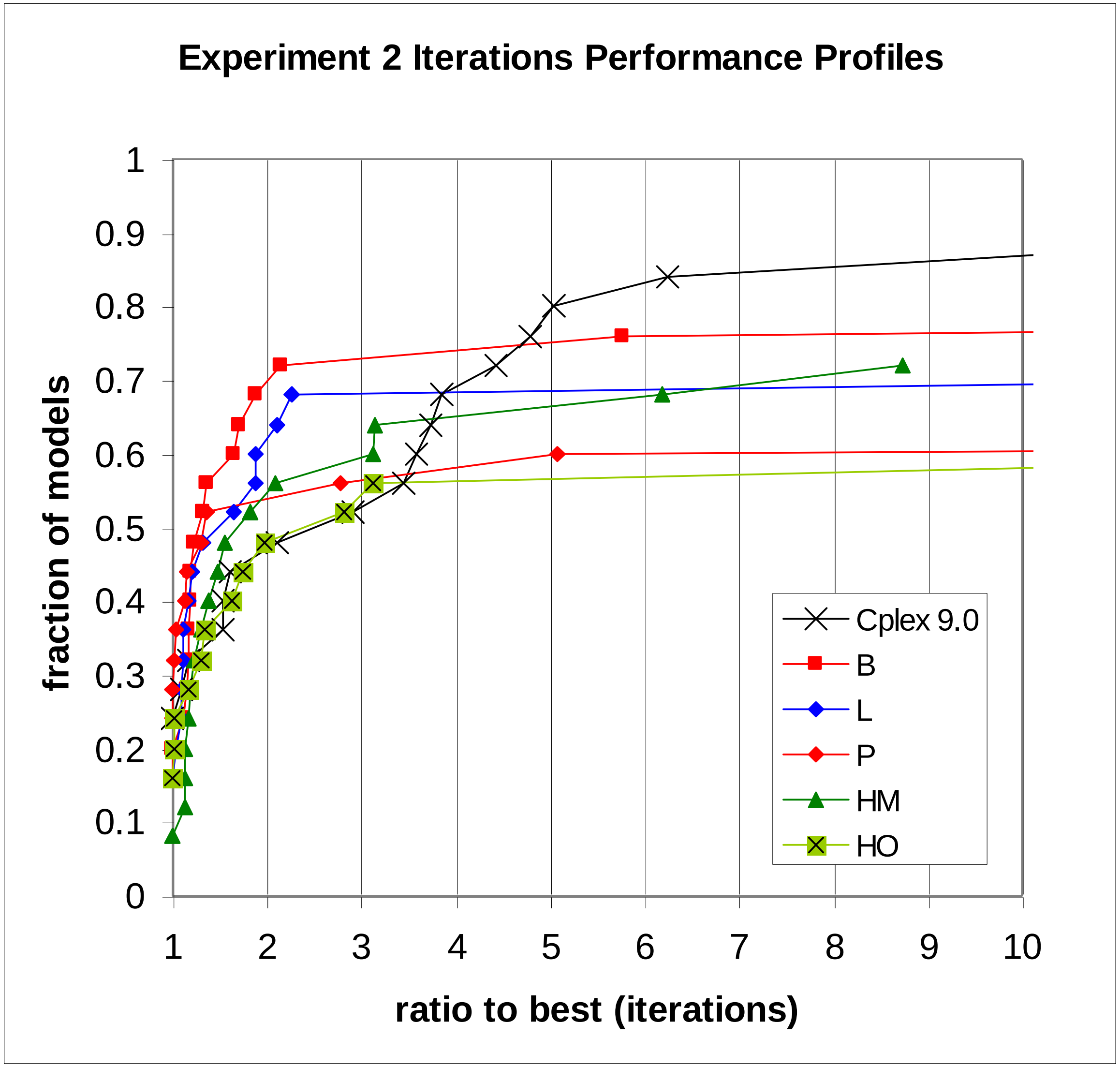
3. Active Constraints Branching
[Patel and Chinneck, 2007]

- ***Insight:*** choose candidate variable having *most impact* on active constraints in current LP relaxation
  - *i.e. force change*
  - All other methods look at impact on *objective fcn*
- Several variants indicated by letters
- ***Method A***: choose candidate variable appearing in largest number of active constraints, branch **up**

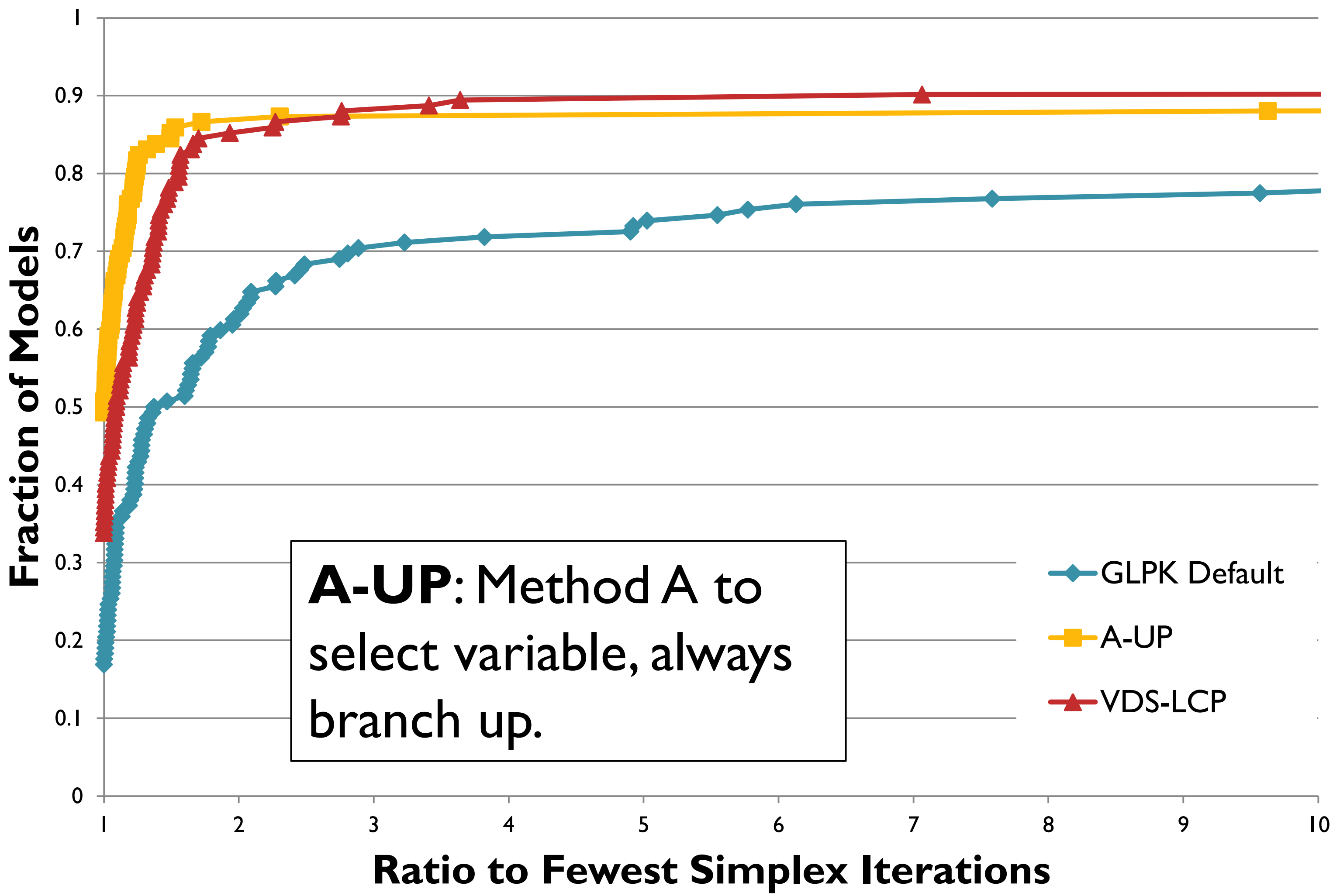
Cplex heuristics all turned off



Cplex heuristics all turned on



A-UP vs. VDS-LCP (all models)



4. More on multiple choice constraints

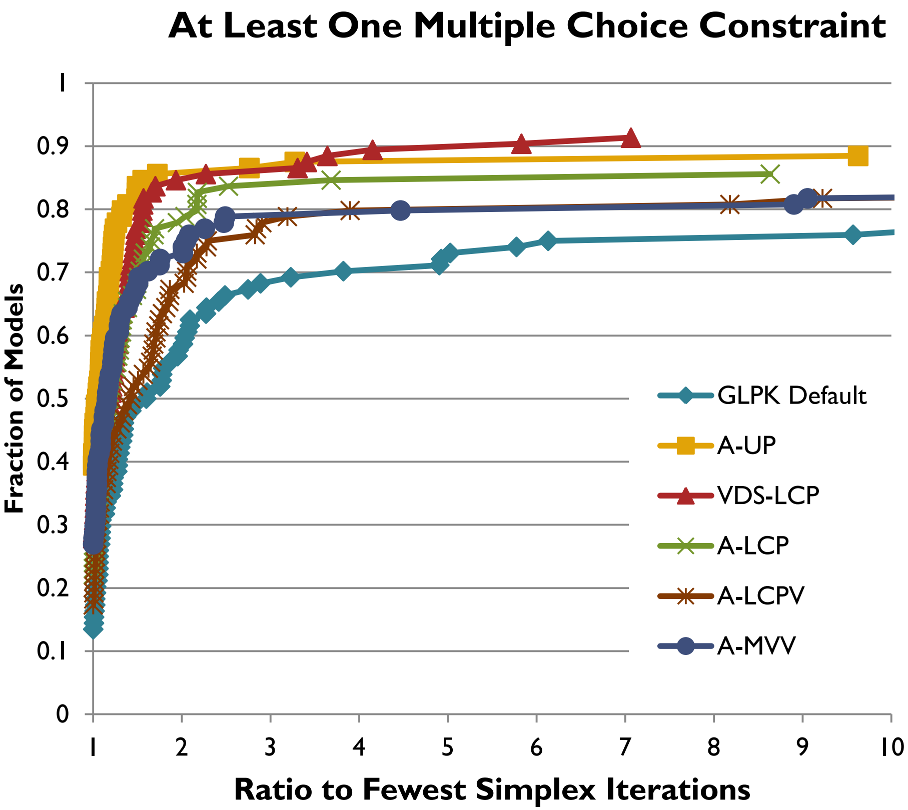
$$x_1 + x_2 \leq 1$$

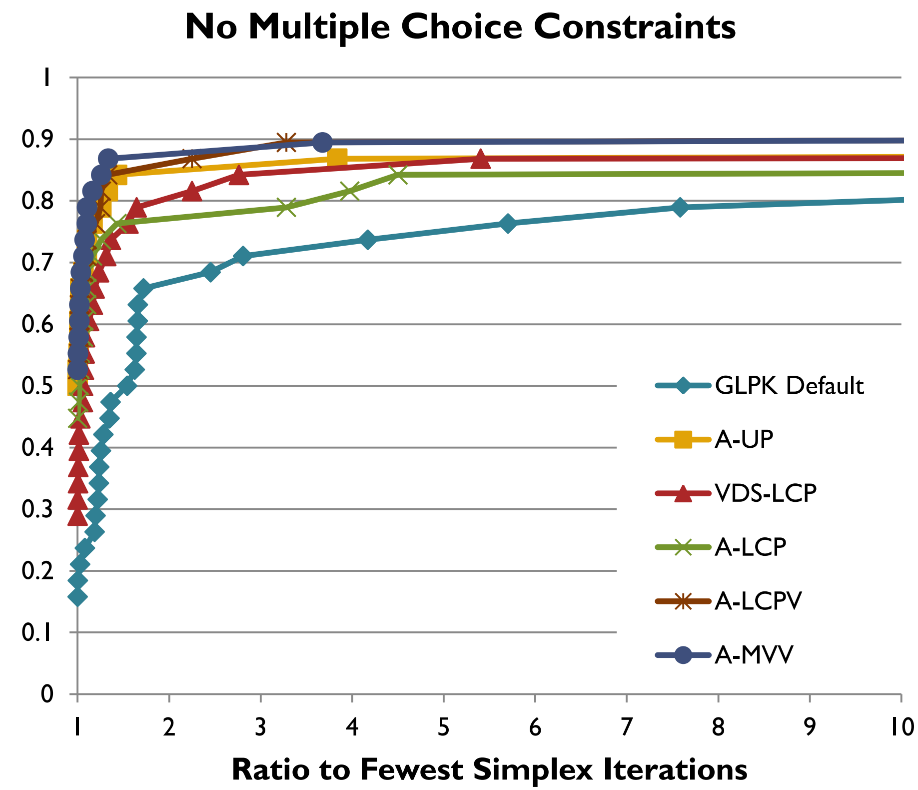
# Variables	Cum. Prob. Up	Cum. Prob. Down
2	**0.158655254**	0.841344746
3	**0.078649604**	0.5
4	**0.041632258**	0.281851431
5	**0.022750132**	0.158655254
6	**0.012673659**	0.089856247

$$x_1 + x_2 = 1$$

# Variables	Equality Ratio Up	Equality Ratio Down
2	**0.188573417**	**0.188573417**
3	**0.085363401**	1
4	**0.043440797**	0.392469529
5	**0.023279749**	0.188573417
6	**0.012836343**	0.098727533

Comparing the better methods on subsets with/without multiple choice constraints





Without multiple choice constraints, other methods outperform A-UP. *Note A-UP vs. A-MVV.*

Conclusions

1. Branching to force variable value propagation is best for MILP:
   - Variables/directions that most affect the active constraints
   - Variables/directions that have low probability of satisfying active constraints
   - Direction that violates the most active constraints
2. Multiple choice constraints are important
   - Equality constraints also have an impact
3. Compare:
   - **MILP:** constraints always satisfied, varbs not integer. *Try to force variables to integrality.*
   - **CP:** varbs always integer, constraints not satisfied. *Try to satisfy constraints.*

References

- 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. Preprint version.
- 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, no. 8, pp. 1143-1152. Preprint version.