Mixed Integer Programming

GUROBI
OPTIMIZATION

Mixed Integer Programming

- A mixed-integer program (MIP) is an optimization problem of the form

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \sum_{j=1}^{n} c_j x_j \\ \text{subject to} \quad & \sum_{j=1}^{n} A_{ij} x_j = b_i, \quad i = 1, \ldots, m, \\ & \ell_j \leq x_j \leq u_j, \quad j = 1, \ldots, n, \\ & \text{some or all } x_j \text{ integer} \end{aligned}$$

- Why do we care about this problem?

Applications

- Accounting
- Advertising
- Agriculture
- Airlines
- ATM provisioning
- Compilers
- Defense
- Electrical power
- Energy
- Finance
- Food service
- Forestry
- Gas distribution
- Government
- Internet applications
- Logistics/supply chain
- Medical
- Mining
- National research labs
- Online dating
- Portfolio management
- Railways
- Recycling
- Revenue management
- Semiconductor
- Shipping
- Social networking
- Sports betting
- Sports scheduling
- Statistics
- Steel Manufacturing
- Telecommunications
- Transportation
- Utilities
- Workforce scheduling
- ...

National Football League

NFL Schedule

- NFL revenues (2014) from TV contracts: ~$5 billion in 2014
- 256 games in regular season
- Each team plays 16 games (with 1 BYE) over 17 weeks
- Each team's opponents are predetermined before scheduling
- NFL begins scheduling immediately after Superbowl (once all opponents known)
- Input from all 32 clubs (stadium blocks, travel considerations, competitive factors, etc.)
- Input from all 5 television partners (key games, dates, markets, etc.)
- Entire process takes 2-3 months

2013 NFL Schedule

DAL	NYG	PHL	WAS	CHI	DET	GB	MIN	ATL	CAR	NO	TB	ARZ	STL	SF	SEA		BUF	MIA	NE	NYJ	BAL	CIN	CLE	PIT	HOU	IND	JAX	TEN	DEN	KC	OAK	SD
NYG	DAL	WAS	PHI	CIN	MIN	SF	DET	NO	SEA	ATL	NYJ	STL	ARZ	GB	CAR	1	NE	CLE	BUF	TB	DEN	CHI	MIA	TEN	SD	OAK	KC	PIT	BAL	JAC	IND	HOU
KC	DEN	SD	GB	MIN	ARZ	WAS	CHI	STL	BUF	TB	NO	DET	ATL	SEA	SF	2	CAR	IND	NYJ	NE	CLE	PIT	BAL	CIN	TEN	MIA	OAK	HOU	NYG	DAL	JAC	PHI
STL	CAR	KC	DET	PIT	WAS	CIN	CLE	MIA	NYG	ARZ	NE	NO	DAL	IND	JAC	3	NYJ	ATL	TB	BUF	HOU	GB	MIN	CHI	BAL	SF	SEA	SD	OAK	PHI	DEN	TEN
SD	KC	DEN	OAK	DET	CHI		PIT	NE		MIA	ARZ	TB	SF	STL	HOU	4	BAL	NO	ATL	TEN	BUF	CLE	CIN	MIN	SEA	JAC	IND	NYJ	PHI	NYG	WAS	DAL
DEN	PHI	NYG		NO	GB	DET		NYJ	ARZ	CHI		CAR	JAC	HOU	IND	5	CLE	BAL	CIN	ATL	MIA	NE	BUF		SF	SEA	STL	KC	DAL	TEN	SD	OAK
WAS	CHI	TB	DAL	NYG	CLE	BAL	CAR		MIN	NE	PHI	SF	HOU	ARZ	TEN	6	CIN		NO	PIT	GB	BUF	DET	NYJ	STL	SD	DEN	SEA	JAC	OAK	KC	IND
PHI	MIN	DAL	CHI	WAS	CIN	CLE	NYG	TB	STL		ATL	SEA	CAR	TEN	ARZ	7	MIA	BUF	NYJ	NE	PIT	DET	GB	BAL	KC	DEN	SD	SF	IND	HOU		JAC
DET	PHI	NYG	DEN		DAL	MIN	GB	ARZ	TB	BUF	CAR	ATL	SEA	JAC	STL	8	NO	NE	MIA	CIN		NYJ	KC	OAK			SF		WAS	CLE	PIT	
MIN		OAK	SD	GB		CHI	DAL	CAR	ATL	NYJ	SEA		TEN		TB	9	KC	CIN	PIT	NO	CLE	MIA	BAL	NE	IND	HOU		STL		BUF	PHI	WAS
NO	OAK	GB	MIN	DET	CHI	PHI	WAS	SEA	SF	DAL	MIA	HOU	IND	CAR	ATL	10	PIT	TB			CIN	BAL		BUF	ARZ	STL	TEN	JAC	SD		NYG	DEN
	GB	WAS	PHI	BAL	PIT	NYG	SEA	TB	NE	SF	ATL	JAC		NO	MIN	11	NYJ	SD	CAR	BUF	CHI	CLE	CIN	DET	OAK	TEN	ARZ	IND	KC	DEN	HOU	MIA
NYG	DAL		SF	STL	TB	MIN	GB	NO	MIA	ATL	DET	IND	CHI	WAS		12		CAR	DEN	BAL	NYJ		PIT	CLE	JAC	ARZ	HOU	OAK	NE	SD	TEN	KC
OAK	WAS	ARZ	NYG	MIN	GB	DET	CHI	BUF	TB	SEA	CAR	PHI	SF	STL	NO	13	ATL	NYJ	HOU	MIA	PIT	SD	JAC	BAL	NE	TEN	CLE	IND	KC	DEN	DAL	CIN
CHI	SD	DET	KC	DAL	PHI	ATL	BAL	GB	NO	CAR	BUF	STL	ARZ	SEA	SF	14	TB	PIT	CLE	OAK	MIN	IND	NE	MIA	JAC	CIN	HOU	DEN	TEN	WAS	NYJ	NYG
GB	SEA	MIN	ATL	CLE	BAL	DAL	PHI	WAS	NYJ	STL	SF	TEN	NO	TB	NYG	15	JAC	NE	MIA	CAR	DET	PIT	CHI	CIN	IND	HOU	BUF	ARZ	SD	OAK	KC	DEN
WAS	DET	CHI	DAL	PHI	NYG	PIT	CIN	SF	NO	CAR	STL	SEA	TB	ATL	ARZ	16	MIA	BUF	BAL	CLE	NE	MIN	NYJ	GB	DEN	KC	TEN	JAC	HOU	IND	SD	OAK
PHI	WAS	DAL	NYG	GB	MIN	CHI	DET	CAR	ATL	TB	NO	SF	SEA	ARZ	STL	17	NE	NYJ	BUF	MIA	CIN	BAL	PIT	CLE	TEN	JAC	IND	HOU	OAK	SD	DEN	KC

- NBC, CBS, FOX, ESPN, NFLN

Home Away

- Left NFC (E, N, S, W), Right AFC (E, N, S, W)

Partial list of rules and goals

- Teams playing in London play home week prior, or have BYE week after
- Bills game in Toronto Week 13 against ATL, BAL, CAR, KC, MIA or NYJ
- No team has earliest BYE in consecutive seasons
- No team plays more than two road games against team coming off their BYE
- All teams playing road Thursday games are home the previous week
- All teams playing home Thursday games have limited travel previous week
- Minimize non-division games during hurricane season (in order to "swap" sites)
- Minimize 3-game road trips (and 3-game home stands for teams with ticket issues)
- Minimize number of division series that end in first half of season
- Minimize number of games that would conflict with MLB postseason
- Maximize the number of late-season division games
- Minimize early-season games 1PM games for teams with weather concerns
- Minimize number of Pacific time zone teams that play at 1PM ET
- CBS/FOX have at least 3 1PM games every week, preferably geographically diverse
- CBS/FOX have at least 5 total games
- Team can play no more than 6 prime time games, and only 3 teams per year can play 6. All other teams can play no more than 5 prime time games (max 4 NBC)

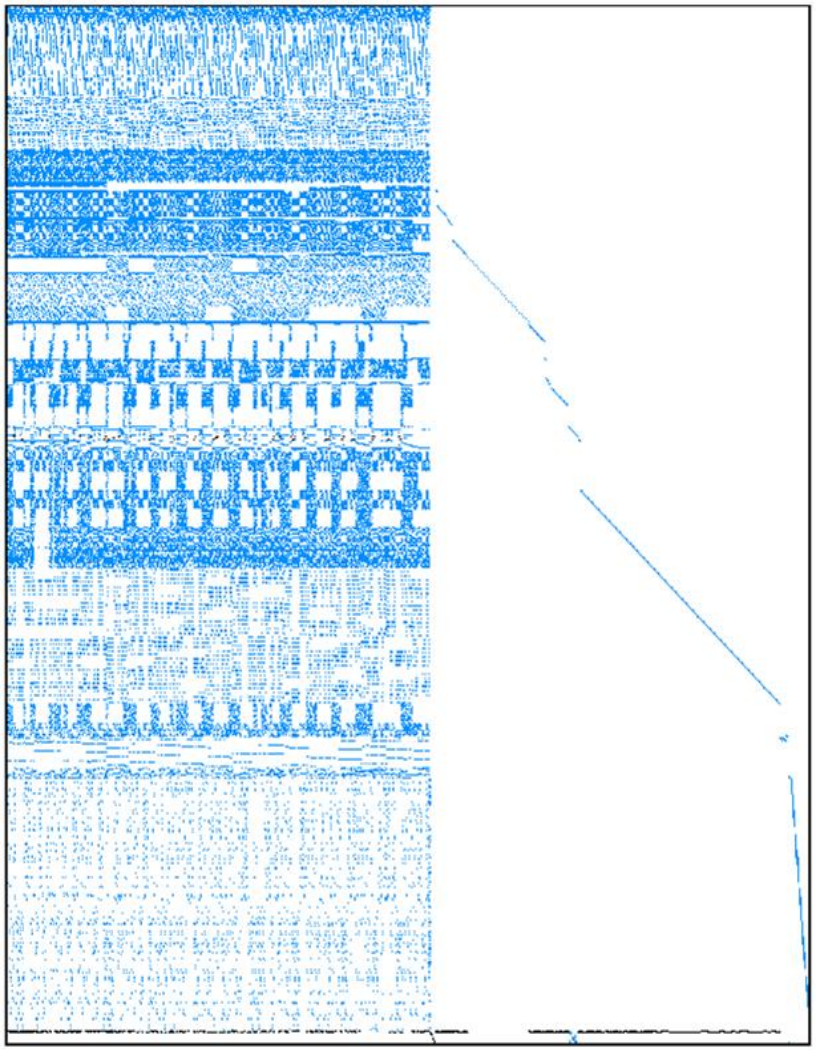
Stadium blocks

DAL	NYG	PHL	WAS	CHI	DET	GB	MIN	ATL	CAR	NO	TB	ARZ	STL	SF	SEA		BUF	MIA	NE	NYJ	BAL	CIN	CLE	PIT	HOU	IND	JAX	TEN	DEN	KC	OAK	SD
BB MON		BB SUN													BB SU-MO	1					BB SU-MO	BB SU-MO							BB SUN	BB SUN	BB SUN
BB SUN		BB MON		COLL. FB	BB SU-MO								LEASE			2					BB THU			BB TH-MO	LEASE					BB MON	BB MON	
BB MON		BB SUN			BB TH-SU		HOME	PGA TOUR					LEASE		BB MON	3						BB MON	COLL. FB	HOME SNF		NO THU				BB SUN	BB TH-SU	BB SU-MO
BB TH-SU	BON JOVI						LONDON								BB SUN	4				BON JOVI	BB TH-SU	BB SUN		LONDON								BB THU*
DIV		DIV			DIV		BYE			NO THU						5					DIV	DIV		BYE	LEASE					NASCAR	DIV	DIV
LCS		LCS		Marathon					NASCAR				LEASE		AUTO	6		CARNIVAL				LCS		LCS							LCS	
LCS		LCS											NO THU	AWAY		7						LCS		LCS							LCS	
WS		WS											Marathon	LONDON	MLS	8						WS		WS			LONDON				WS	
										NO THU				BYE		9											BYE					
													LEASE		MLS?	10									LEASE							
			SOCCER							NO THU			LEASE			11		NO MNF								NO PRIME						
															MLS?	12										LEASE						NO MNF
TGIV					TGIV					NO SUN 1p						13								HIGH SCHOOL								
KU HOOPS								SEC	ACC	NO SUN 1p			LEASE		MLS CUP	14	Toronto?								LEASE	BIG 10						
		ARMY NAVY								NO SUN			LEASE?			15																
																16																
									BELK							17																

- Of the 544 potential sites, almost 20% unavailable or had some sort of conflict
- 13 teams had 4 or more potential conflicts, affecting 25% or more of their potential home dates

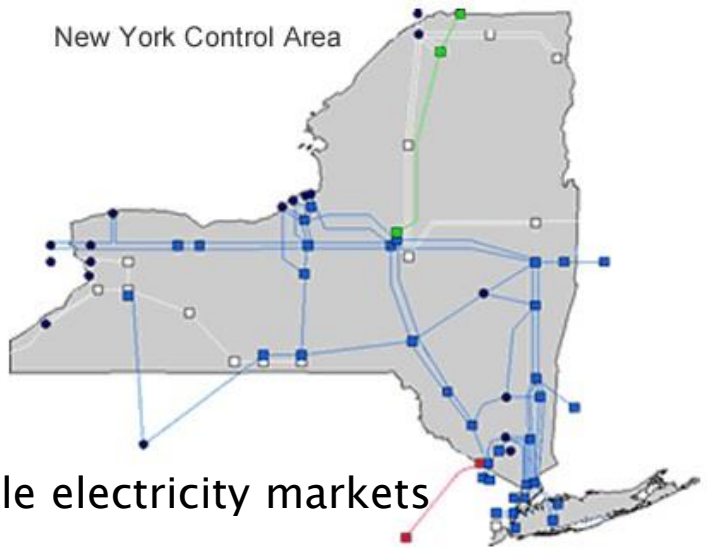
A $5 billion MIP Model?

- All constraints captured in one MIP:
  - original: 25K rows, 20K cols, 800K nz
  - presolved: 6K rows, 6K cols, 160K nz
    - 3600 binary variables
- What does a solution look like:
  - 256 binary variables to set to 1
    - Each captures the particulars of one game
  - the rest set to 0



New York Independent System Operator

What is NYISO?



- NYISO is a non-profit, statutory agency
  - began operating in 1999
- NYISO primary responsibilities:
  - operating New York's high-voltage grid
  - administering and monitoring state's wholesale electricity markets
  - long-term planning for high-voltage grid
    - reliability, future maintenance and expansion
- Size
  - grid encompasses 11,000 miles of transmission lines
  - manages 500 power-generation units
  - administers $7.5 billion in annual transactions

Optimization Plays a Central Role In NYISO Operations

- Day-ahead problem
  - unit-commitment for the next day: which units to be used
    - markets close 5 am
    - commitments must be posted by 11 am
    - 30 minutes available to compute commitments
- Real-time problem
  - solved every 5 minutes, restricted unit-commitment
- Real-time dispatch (EASY!)
  - linear program, solved every 5 minutes
  - determines generator clearing prices

Traditional Solution Approach: Lagrangian Relaxation (LR)

- In use since the early 1980s
- Claimed to produce high-quality solutions
- Solution times are quite good
- Considerable reluctance to make a change
- However in 1999 MIP was demonstrated to be a viable alternative*
  - pressure was mounting to switch

*Rutgers academic-industry meeting sponsored by Federal Energy Regulatory Commission (FERC).

Experience in Other ISOs*

- In 2004, PJM implemented MIP in its day-ahead market
  - estimated of annual production cost savings of $60 million
- In 2006, PJM implemented MIP in its real-time market look-ahead
  - test findings of $100 million in annual savings
- In April 2009, CAISO implemented MIP as part of its Market Redesign and Technology Update
  - estimated savings of $52 million
- In 2009, the Southwest Power Pool (SPP) introduced MIP enhancements to its day-ahead market
  - estimated $103 million in annual benefits

*FERC Staff Report, 2011 (Recent ISO Software Enhancements and Future Software and Modeling Plans)

MIP: Other Advantages

- Maintainability
  - LR code: 100,000 lines of FORTRAN
  - understood by only one or two people
- Transparency
  - MIP formulation has much simpler representation
    - easy to read, interpret, and maintain
  - solutions much easier to defend against legal challenges
- Extensibility
  - extremely difficult to add constraints to LR
  - new constraints easy to add to MIP formulation
    - combined cycle-generator modeling enabled

Solution Implementation

- Solution launch date: Q2 2014
- Computing environment
  - 50 HPUX client machines
  - 12 Linux compute servers (3-production, 3-stage, 3-development)
- Estimated savings: $4M yearly production costs
- Future
  - underlying models are highly nonlinear
    - MIP speed improvements can always be used to improve model accuracy
  - conclusion:
    - continued MIP improvements will lead directly to efficiency gains in electrical-power markets!

Conclusion: Use MIP Instead of Custom Solution!

- Easier to maintain a MIP model than complex source code
- Easier to extend a MIP model than to adapt a custom algorithm
- Easier to demonstrate correctness of a MIP model
- Benefit from performance improvements over time coming from MIP solver vendors
  - algorithmic speed-up exceeded hardware speed-up during the last 20 years!

Solving Mixed Integer Programs

MIP Application Types

- Static MIP
  - Formulate problem
  - Solve it with a black-box MIP algorithm
  - Read solution
  - Potentially adjust problem and iterate
  - most frequent use of MIP in practical applications
- Branch-and-cut
  - Problem has too many constraints to formulate in static fashion
    - e.g., classical TSP model: exponentially many sub-tour elimination constraints
  - Construct partial problem
  - Add violated constraints on demand
    - "Lazy constraint" separation callback: cut off primal infeasible LP or IP solutions
- Branch-and-price
  - Problem has too many variables to formulate in static fashion
    - e.g., many public transport and airline problems are solved via B&P
  - Pricing callback: cut off dual infeasible LP solutions
  - Usually needs problem specific branching rule that is compatible with pricing
  - Heuristic variant: column generation
    - Only apply pricing for the root LP, then solve static MIP with resulting set of variables

MIP Building Blocks

- Presolve
  - Tighten formulation and reduce problem size
- Solve continuous relaxations
  - Ignoring integrality
  - Gives a bound on the optimal integral objective
- Cutting planes
  - Cut off relaxation solutions
- Branching variable selection
  - Crucial for limiting search tree size
- Primal heuristics
  - Find integer feasible solutions

MIP Building Blocks

- Presolve
  - Tighten formulation and reduce problem size
- Solve continuous relaxations
  - Ignoring integrality
  - Gives a bound on the optimal integral objective
- Cutting planes
  - Cut off relaxation solutions
- Branching variable selection
  - Crucial for limiting search tree size
- Primal heuristics
  - Find integer feasible solutions

LP Presolve

- Goal
  - Reduce the problem size
    - Speedup linear algebra during the solution process
- Example

$$x + y + z \leq 5 \qquad (1)$$
$$u - x - z = 0 \qquad (2)$$
$$\ldots\ldots\ldots$$
$$0 \leq x, y, z \leq 1 \qquad (3)$$
$$u \text{ is free} \qquad (4)$$

- Reductions
  - Redundant constraint
    - (3) $\Rightarrow x + y + z \leq 3$, so (1) is redundant
  - Substitution
    - (2) and (4) $\Rightarrow u$ can be substituted with $x + z$

MIP Presolve

- Goals:
  - Reduce problem size
    - Speed-up linear algebra during the solution process
  - Strengthen LP relaxation
  - Identify problem sub-structures
    - Cliques, implied bounds, networks, disconnected components, ...
- Similar to LP presolve, but more powerful:
  - Exploit integrality
    - Round fractional bounds and right hand sides
    - Lifting/coefficient strengthening
    - Probing
  - Does not need to preserve duality
    - We only need to be able to uncrush a primal solution
    - Neither a dual solution nor a basis needs to be uncrushed

MIP Presolve

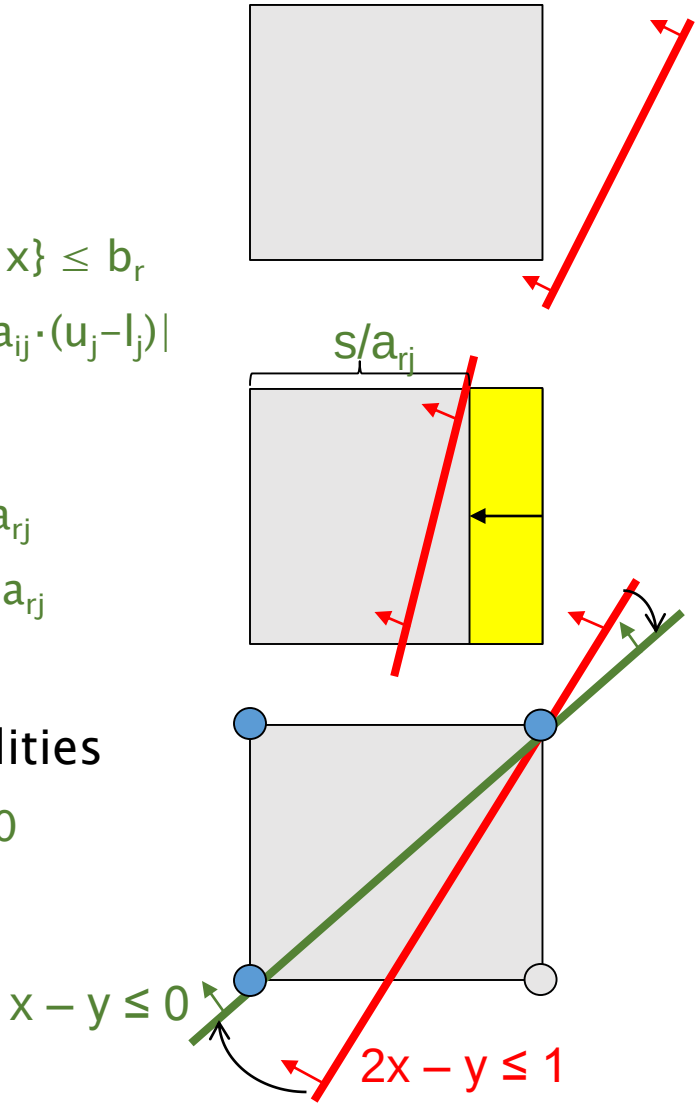
- Goals:
  - Reduce problem size
    - Speed-up linear algebra during the solution process
  - Strengthen LP relaxation
  - Identify problem sub-structures
    - Cliques, implied bounds, networks, disconnected components, ...
- Similar to LP presolve, but more powerful:
  - Exploit integrality
    - Round fractional bounds and right hand sides
    - Lifting/coefficient strengthening

model	without presolve			with presolve		
	rows	cols	LP obj	rows	cols	LP obj
roll3000	2291	1166	11097.1	987	855	11120.0
neos-787933	1897	236376	3.0	41	126	30.0

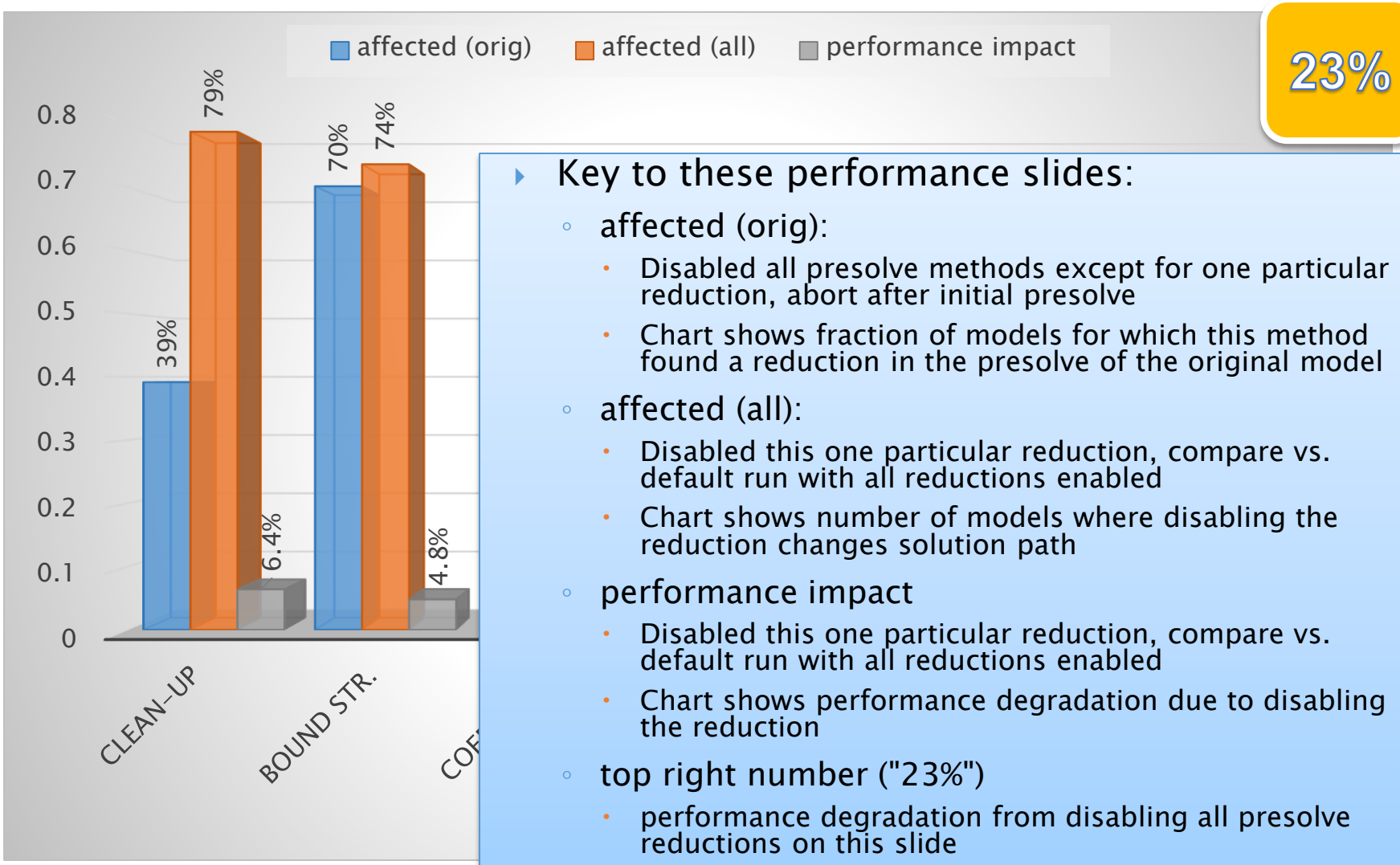
Single-Row Reductions

- Clean-up rows
  - Discard empty rows
  - Discard redundant inequalities: $\sup\{A_r.x\} \leq b_r$
  - Remove coefficients with tiny impact $|a_{ij} \cdot (u_j - l_j)|$
- Bound strengthening
  - $a_{rj} > 0$, $s := b_r - \inf\{A_r.x\} \Rightarrow x_j \leq l_j + s/a_{rj}$
  - $a_{rj} < 0$, $s := b_r - \inf\{A_r.x\} \Rightarrow x_j \geq u_j + s/a_{rj}$
- Coefficient strengthening for inequalities
  - $j \in I$, $a_{rj} > 0$, $t := b_r - \sup\{A_r.x\} + a_{rj} > 0$

    $\Rightarrow a_{rj} := a_{rj} - t$, $b_r := b_r - u_j t$



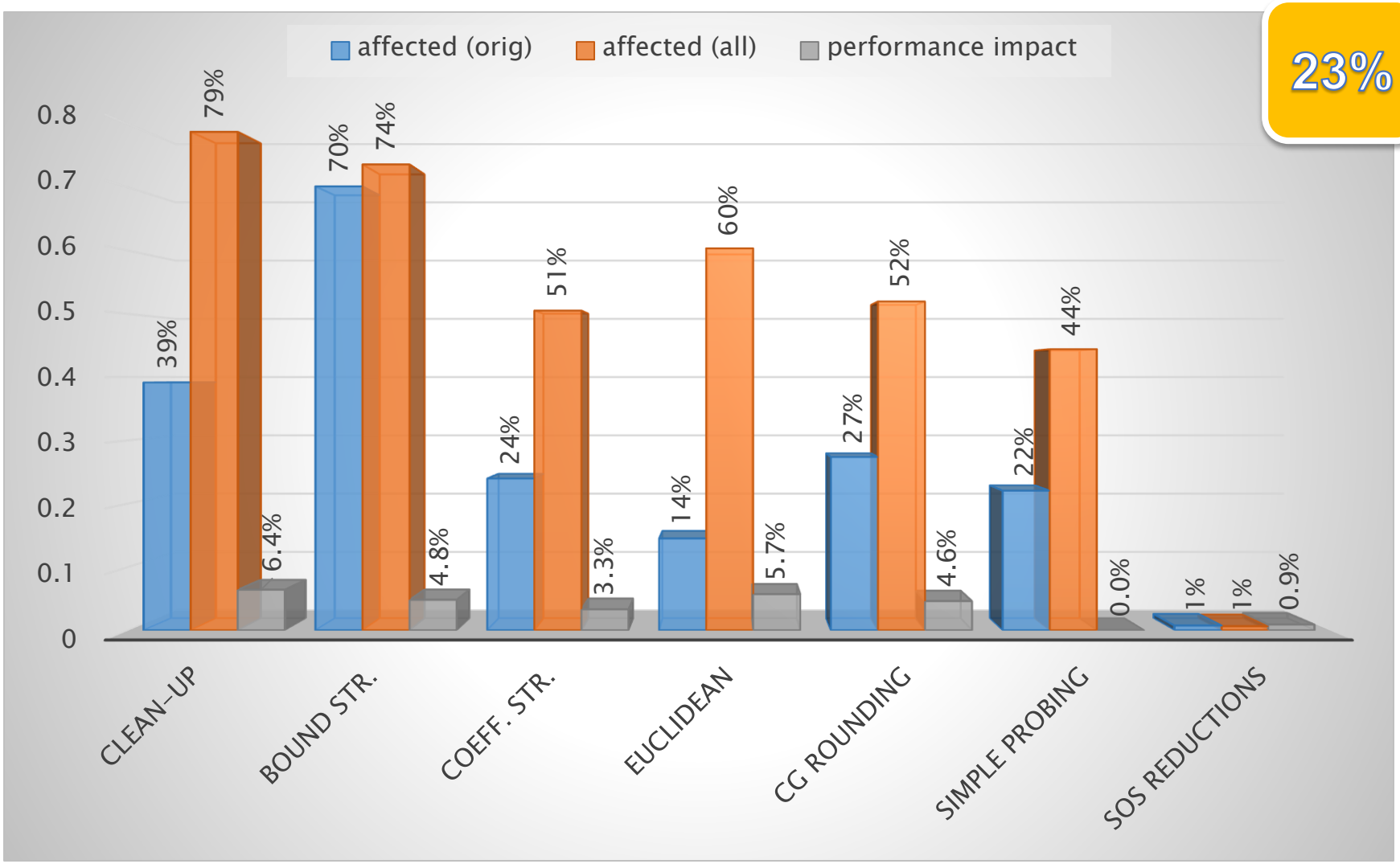
Single-Row Reductions – Performance



23%

- Key to these performance slides:
  - affected (orig):
    - Disabled all presolve methods except for one particular reduction, abort after initial presolve
    - Chart shows fraction of models for which this method found a reduction in the presolve of the original model
  - affected (all):
    - Disabled this one particular reduction, compare vs. default run with all reductions enabled
    - Chart shows number of models where disabling the reduction changes solution path
  - performance impact
    - Disabled this one particular reduction, compare vs. default run with all reductions enabled
    - Chart shows performance degradation due to disabling the reduction
  - top right number ("23%")
    - performance degradation from disabling all presolve reductions on this slide

Single-Row Reductions – Performance

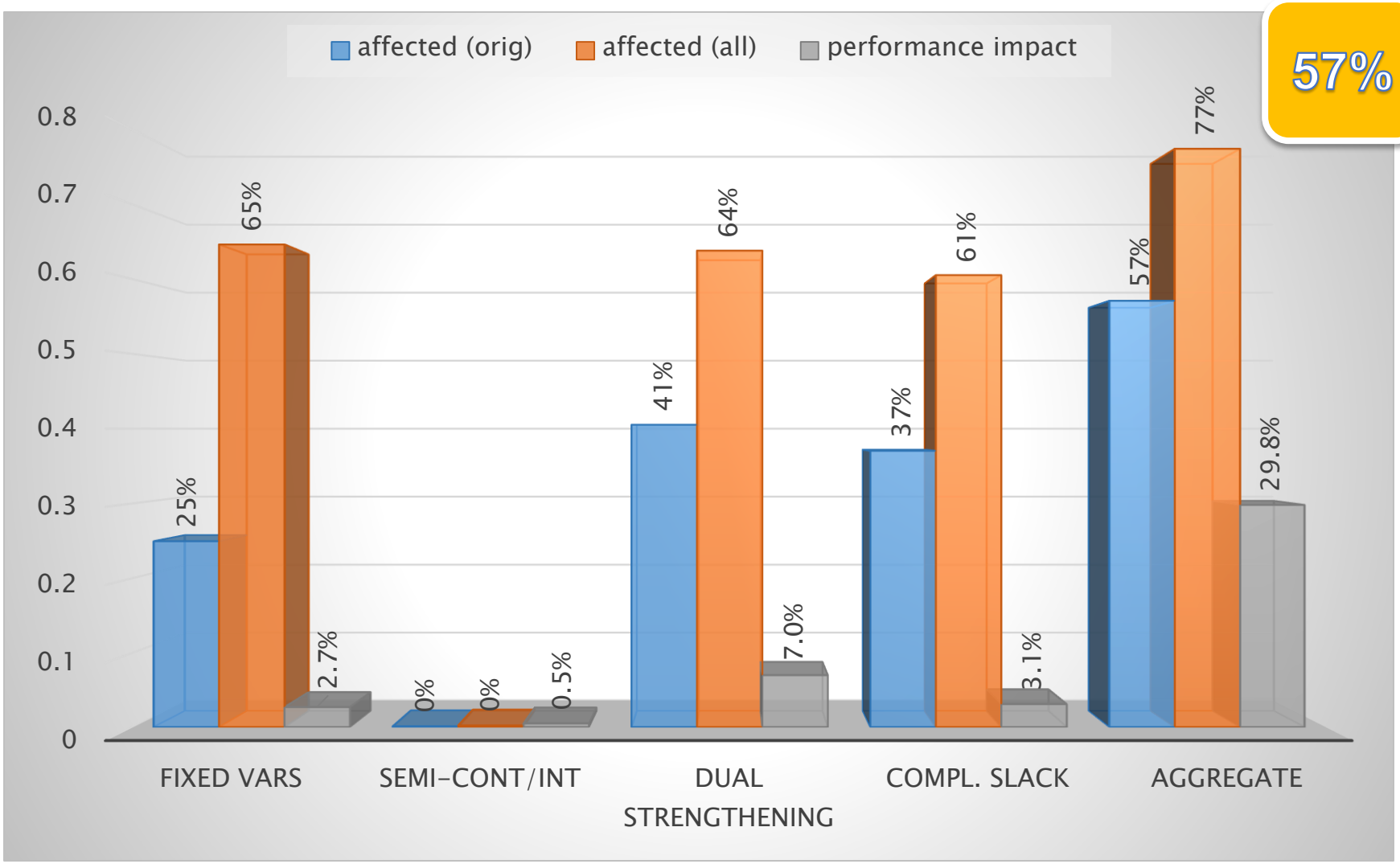


benchmark data based on Gurobi 5.6

Single-Column Reductions

- Remove fixed variables and empty columns
  - If $|u_j - l_j| \leq \epsilon$, fix to some value in $[l_j, u_j]$ and move terms to rhs
  - Choice of value can be very tricky for numerical reasons
- Round bounds of integer variables
- Strengthen semi-continuous and semi-integer variables
- Dual fixing, substitution, and bound strengthening
  - Variable x_j does not appear in equations
  - $c_j \geq 0$, $A_{\cdot j} \geq 0 \quad \Rightarrow \quad x_j := l_j$
  - $c_j \geq 0$, $A_{\cdot j} \geq 0$ except for $a_{ij} < 0$,
    $z = 0 \rightarrow$ row i redundant,
    $z = 1 \rightarrow x_j = u_j$
    $\Rightarrow \; x_j := l_j + (u_j - l_j) \cdot z$
  - $c_j \geq 0$, all rows i with $a_{ij} < 0$ redundant for $x_j \geq t \quad \Rightarrow \quad x_j \leq \max\{l_j, t\}$

Single-Column Reductions – Performance



benchmark data based on Gurobi 5.6

Multi-Row Reductions

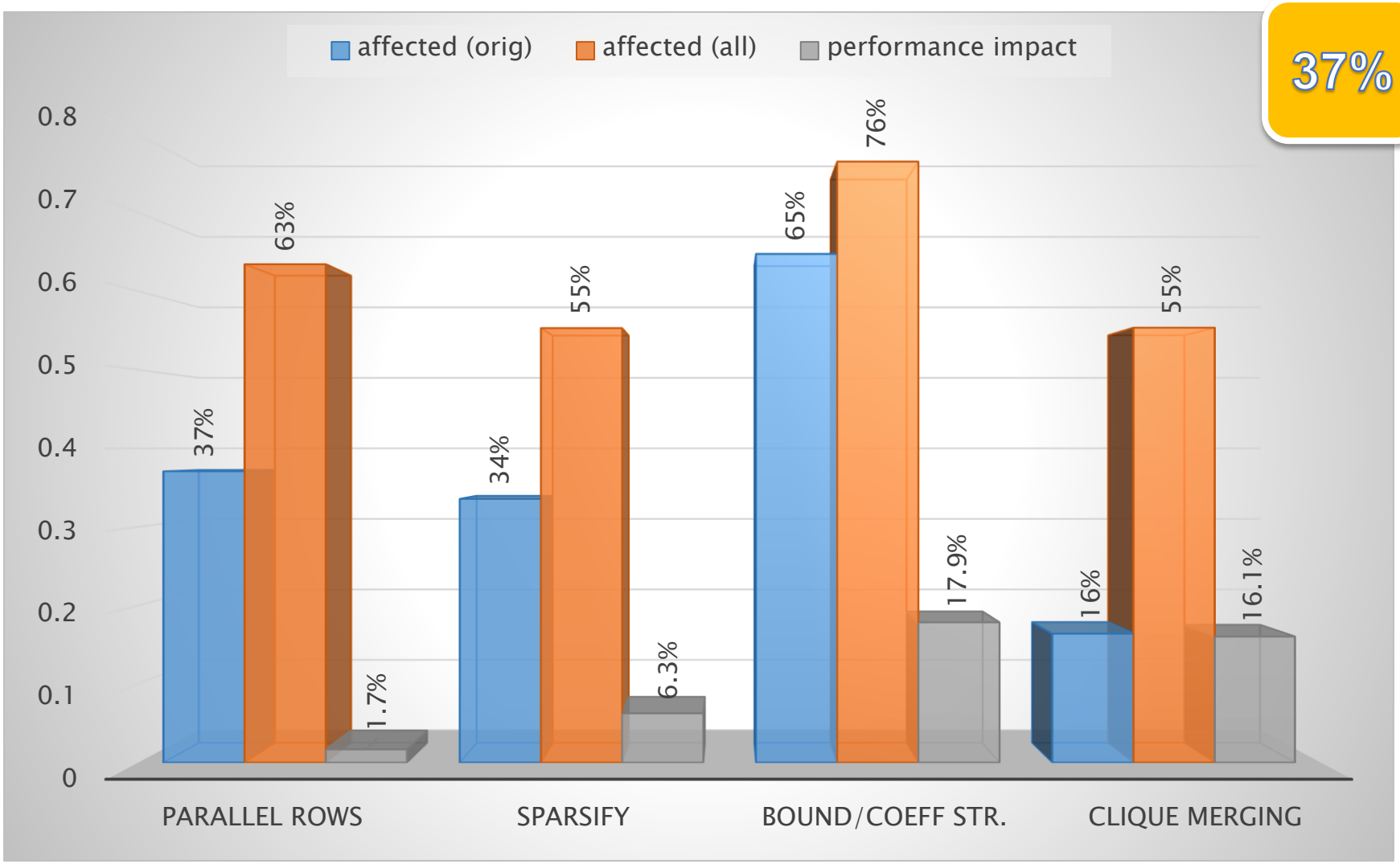
- Parallel rows
  - Search for pairs of rows such that coefficient vectors are parallel to each other
  - Discard the dominated row, or merge two inequalities into an equation
- Sparsify
  - Add equations to other rows in order to cancel non-zeros
  - Can also add inequalities with explicit slack variables
- Multi-row bound and coefficient strengthening
  - Like single-row version, but use other rows to get tighter bound on infimum and supremum $\Rightarrow$ tighter bounds, better coefficients
- Clique merging
  - Merge multiple cliques into larger single clique, e.g.:

$$\begin{aligned} x_1 + x_2 &\leq 1 \\ x_1 + x_3 &\leq 1 \\ x_2 + x_3 &\leq 1 \end{aligned}$$

with binary variables x_1, x_2, x_3 can be merged into

$$x_1 + x_2 + x_3 \leq 1$$

Multi-Row Reductions

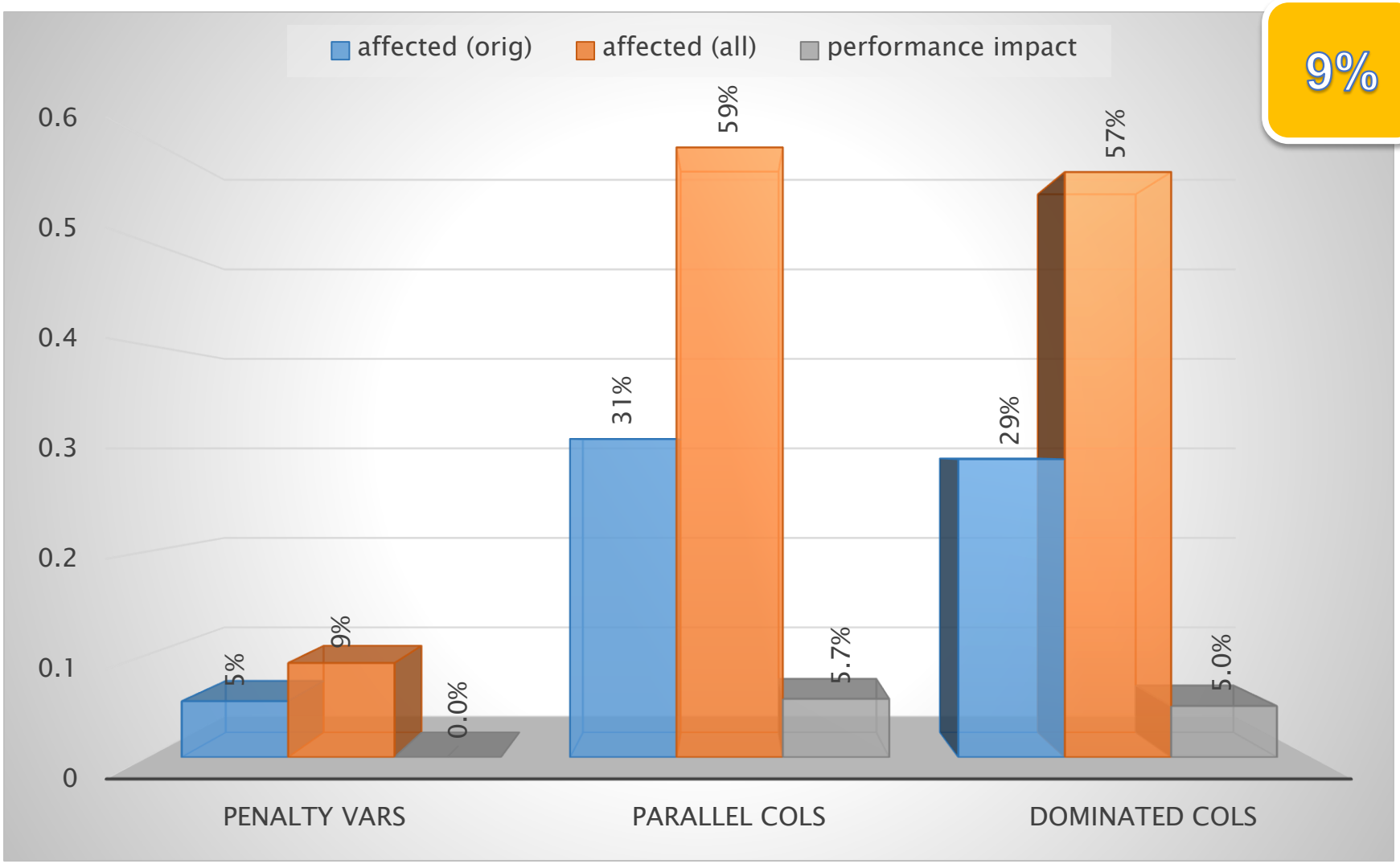


benchmark data based on Gurobi 5.6

Multi-Column Reductions

- Fix redundant penalty variables
  - Penalty variables: $\text{support}(A_{.j}) = 1$
  - Multiple penalty variables in a single constraint
    - Some can be fixed if others can accomplish all that is needed
- Parallel columns (say, columns 1 and 2): $A_{.1} = sA_{.2}$
  - $u_2 = \infty$, $c_1 \geq sc_2$, $2 \notin I$ or $(|s| = 1, \{1,2\} \subseteq I)$: $x_1 := l_1$
  - $l_2 = -\infty$, $c_1 \leq sc_2$, $2 \notin I$ or $(|s| = 1, \{1,2\} \subseteq I)$: $x_1 := u_1$
  - $c_1 = sc_2$, $1,2 \notin I$ or $(|s| = 1, \{1,2\} \subseteq I)$: $x_{1'} := x_1 + sx_2$
  - Detection algorithm: two level hashing plus sorting
- Dominated columns: $A_{.1} \geq sA_{.2}$, only inequalities
  - $u_2 = \infty$, $c_1 \geq sc_2$, $2 \notin I$ or $(|s| = 1, \{1,2\} \subseteq I)$: $x_1 := l_1$
  - Detection algorithm: essentially pair-wise comparison
    - Can be very expensive: needs work limit

Multi-Column Reductions – Performance

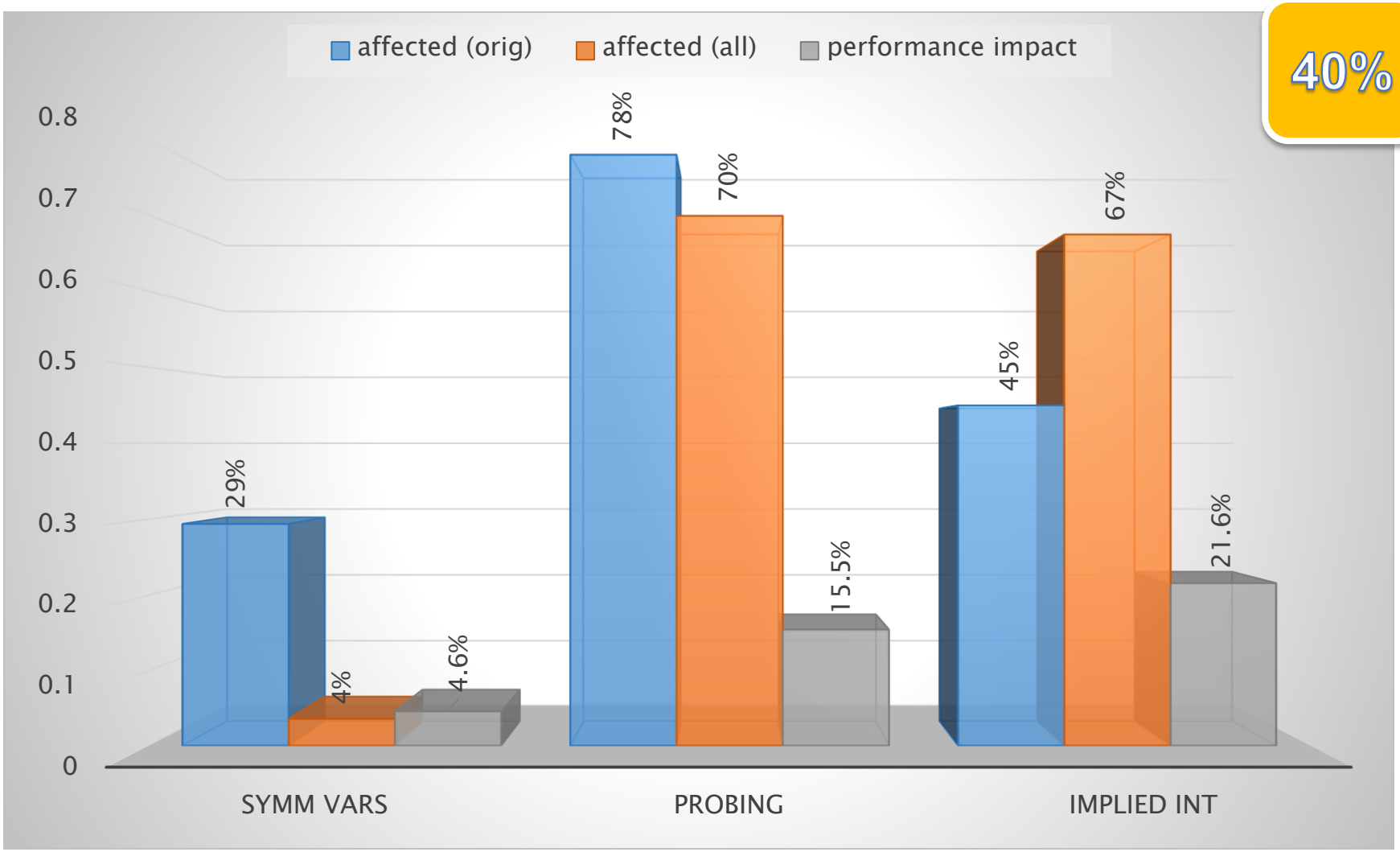


benchmark data based on Gurobi 5.6

Full Problem Reductions

- Symmetric variable substitution
  - Integer variables in same orbit can be aggregated if the involved symmetries do not overlap
  - Continuous variables in same orbit can always be aggregated
  - Issue: symmetry detection can sometimes be time consuming!
- Probing
  - Tentatively fix binary $x = 0$ and $x = 1$, propagate fixing to get domain reductions for other variables
    - $x = 0 \rightarrow y \leq u_0$, $x = 1 \rightarrow y \leq u_1$ $\Rightarrow$ $y \leq \max\{u_0,u_1\}$ (bound strength.)
    - $x = 0 \rightarrow y = l_y$, $x = 1 \rightarrow y = u_y$ $\Rightarrow$ $y := l_y + (u_y - l_y) \cdot x$ (substitution)
    - $ay \leq b$, $x = 1 \rightarrow ay \leq d < b$ $\Rightarrow$ $ay + (b-d) \cdot x \leq b$ (lifting)
  - Sequence dependent
  - Can be very time consuming
    - Needs specialized data structures and algorithms
- Implied Integer Detection
  - Primal version: $ax + y = b$, x integer variables, $a \in \mathbb{Z}^n$, $b \in \mathbb{Z} \Rightarrow y$ integer
  - Dual version:
    - One of the inequalities for y will be tight, but do not know which
    - If all those inequalities lead to primal version of implied integer detection, y is implied integer

Full Problem Reductions



benchmark data based on Gurobi 5.6

MIP Building Blocks

- Presolve
  - Tighten formulation and reduce problem size
- Solve continuous relaxations
  - Ignoring integrality
  - Gives a bound on the optimal integral objective
- Cutting planes
  - Cut off relaxation solutions
- Branching variable selection
  - Crucial for limiting search tree size
- Primal heuristics
  - Find integer feasible solutions

Primal and Dual LP

- Primal Linear Program:

$$\begin{aligned} \min \quad & c^T x \\ s.t. \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

- Weighted combination of constraints (y) and bounds (z) yields

$$y^T Ax + z^T x \geq y^T b \quad (\text{with } z \geq 0)$$

- Dual Linear Program:

$$\begin{aligned} \max \quad & y^T b \\ s.t. \quad & y^T A + z^T = c^T \\ & z \geq 0 \end{aligned}$$

Strong Duality Theorem:

$$c^T x^* = y^{*T} b$$

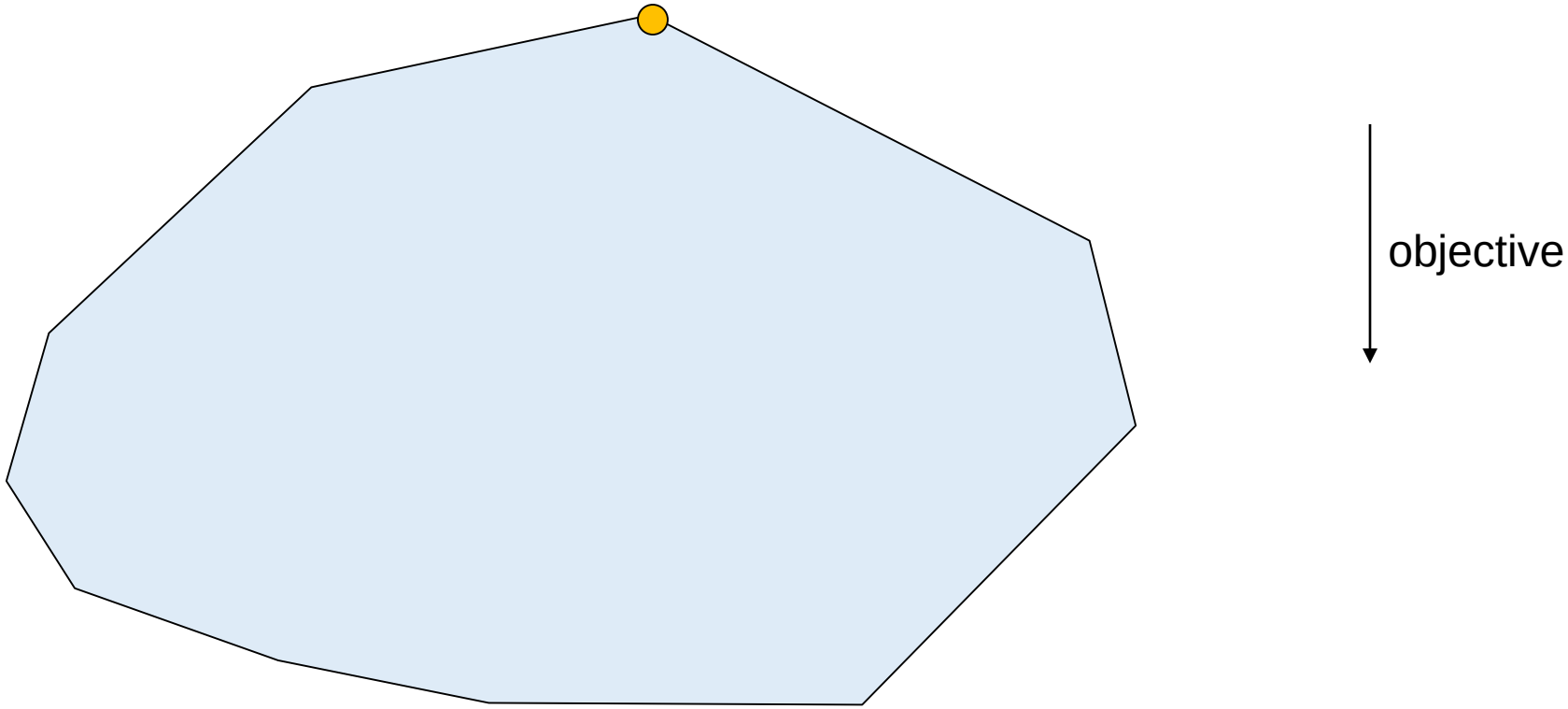
(if primal and dual are both feasible)

Karush-Kuhn-Tucker Conditions

- Conditions for LP optimality:
  - Primal feasibility: $Ax = b$ $(x \geq 0)$
  - Dual feasibility: $A^T y + z = c$ $(z \geq 0)$
  - Complementarity: $x^T z = 0$

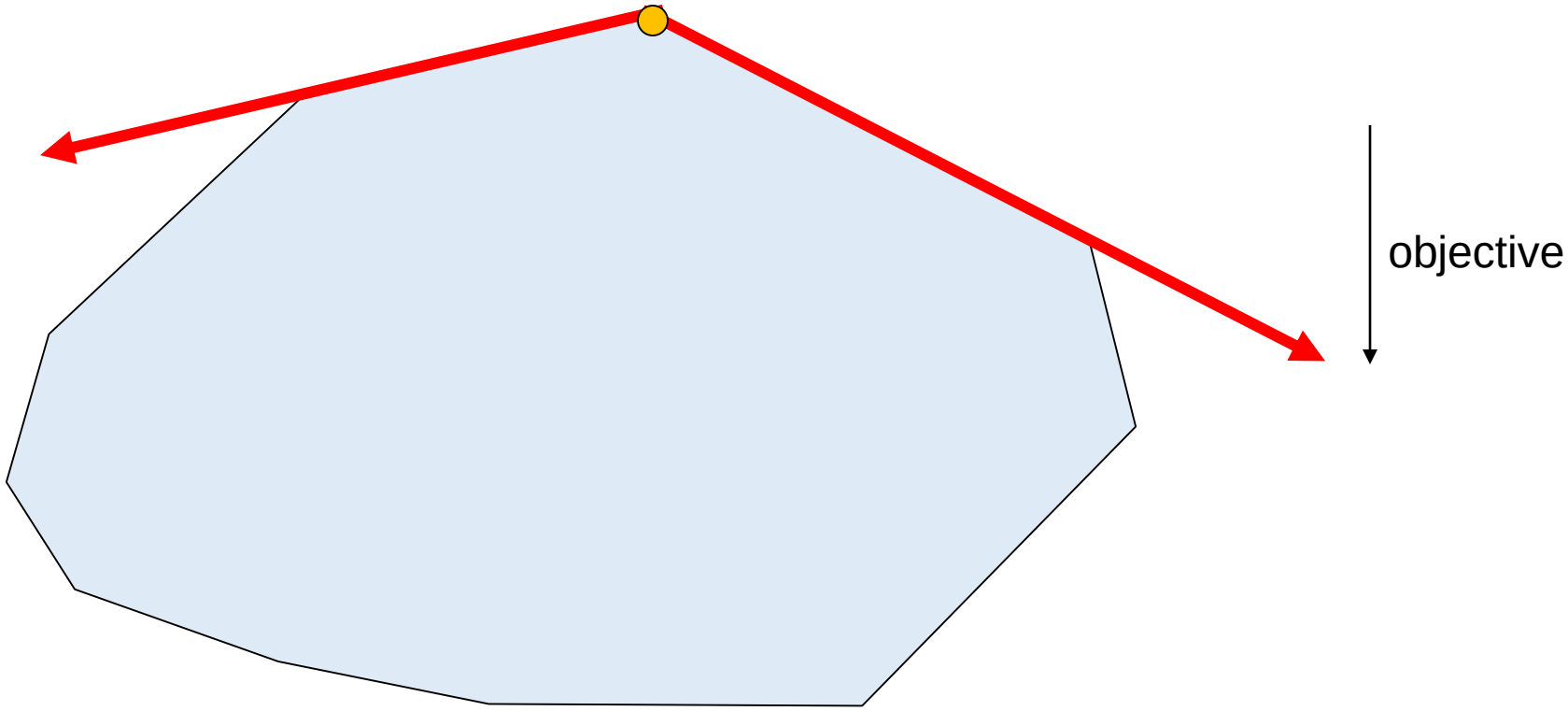
	Primal feas	Dual feas	Complementarity
Primal simplex	Maintain	Goal	Maintain
Dual simplex	Goal	Maintain	Maintain
Barrier	Goal	Goal	Goal

Simplex Algorithm



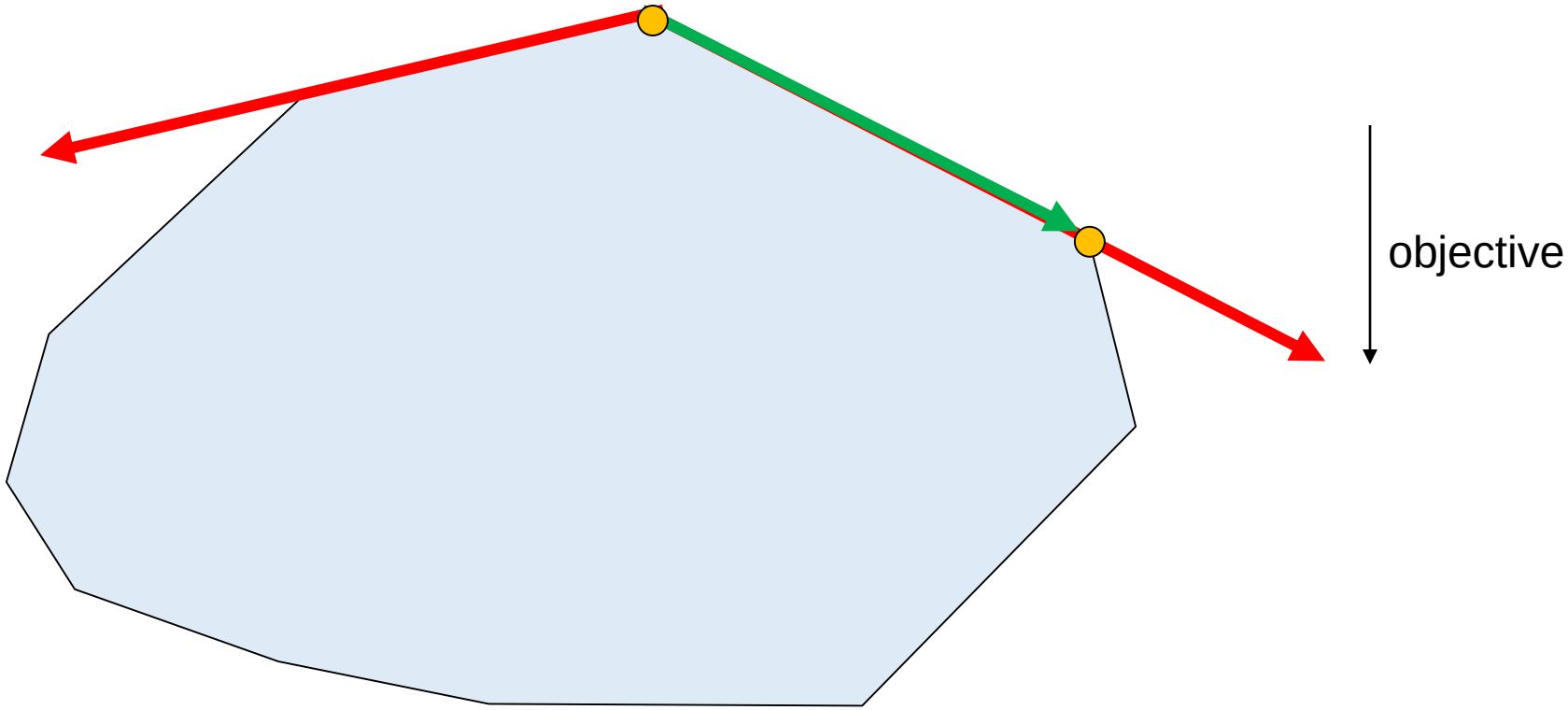
- **Phase 1**: find some feasible vertex solution

Simplex Algorithm



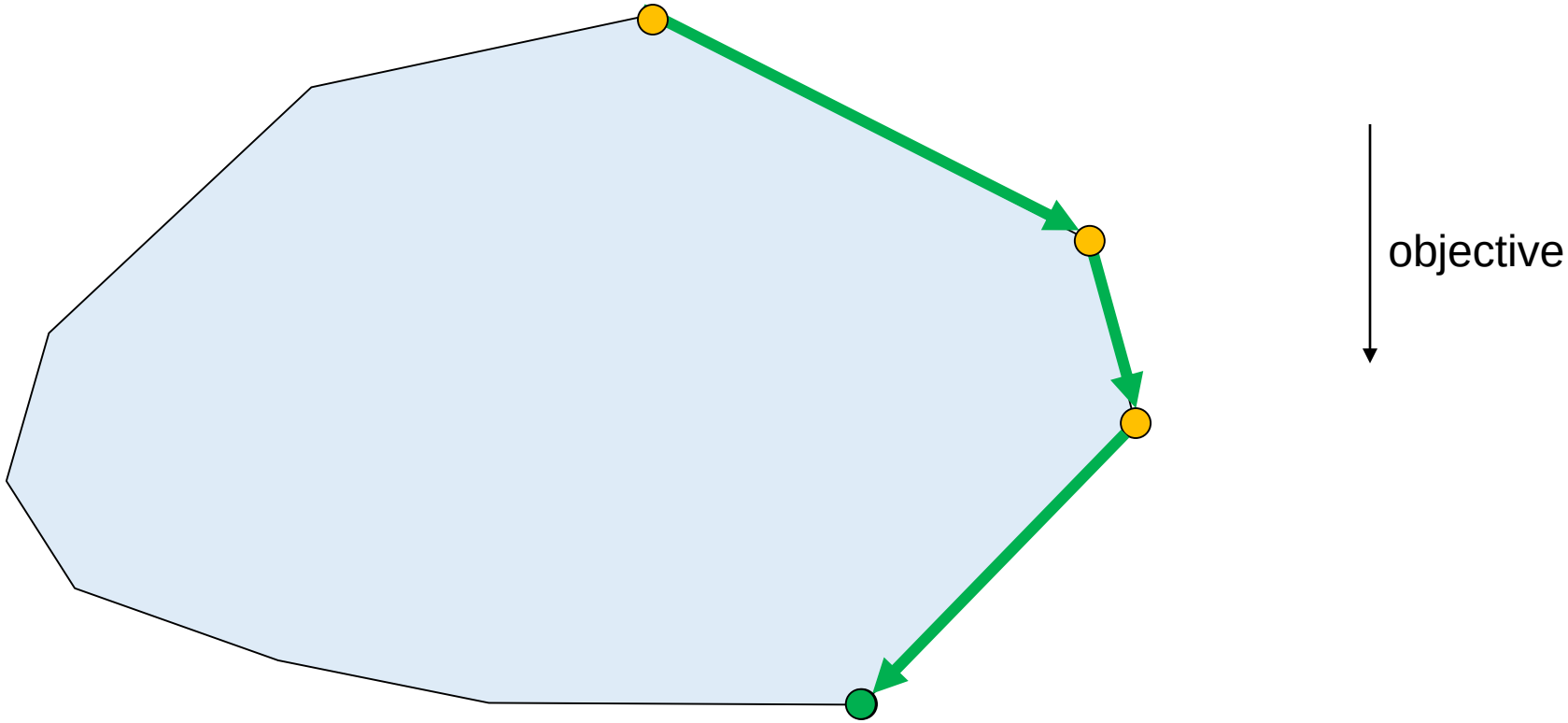
- **Pricing**: find directions in which objective improves and select one of them
  - Gurobi parameters: SimplexPricing, NormAdjust

Simplex Algorithm



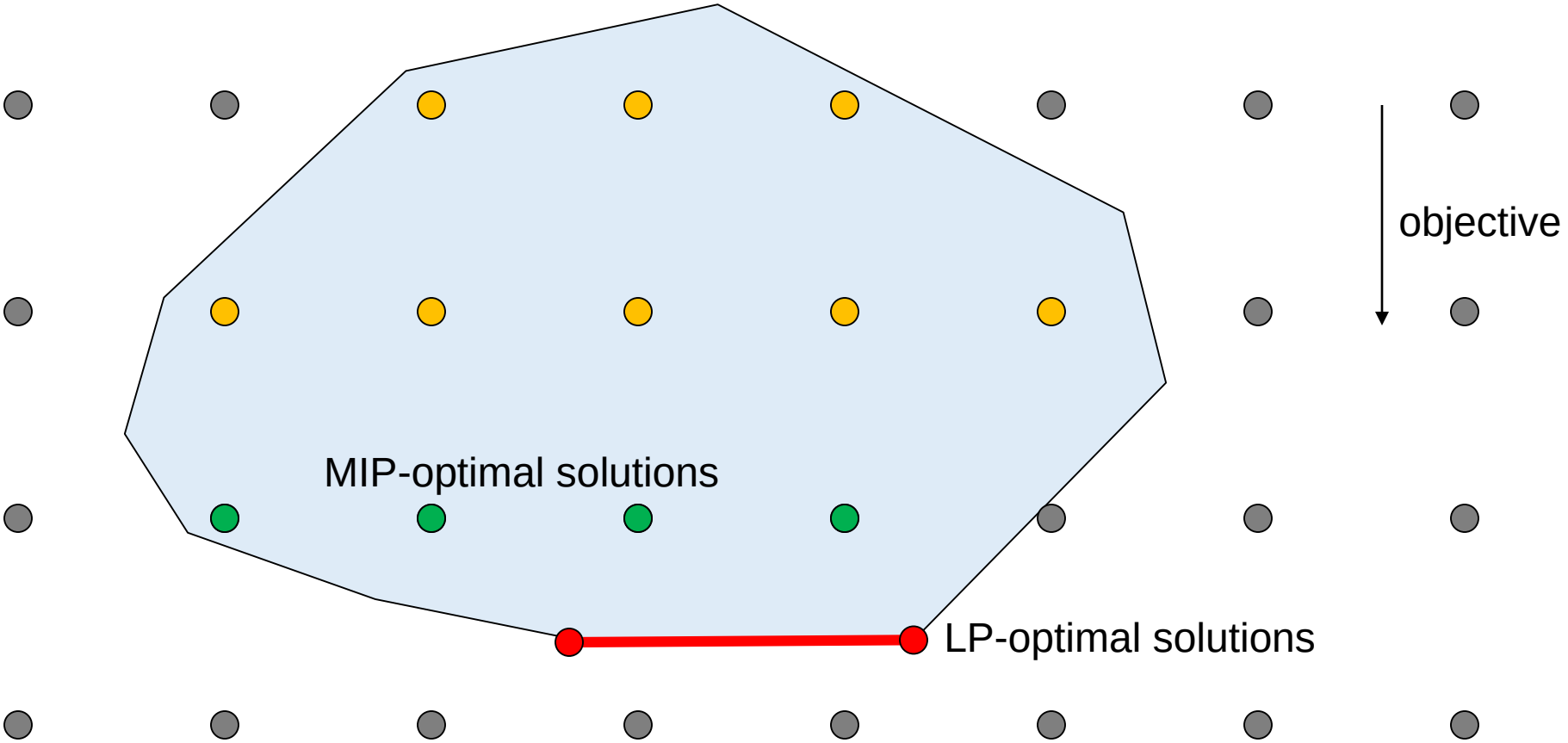
- **Ratio test**: follow outgoing ray until next vertex is reached

Simplex Algorithm

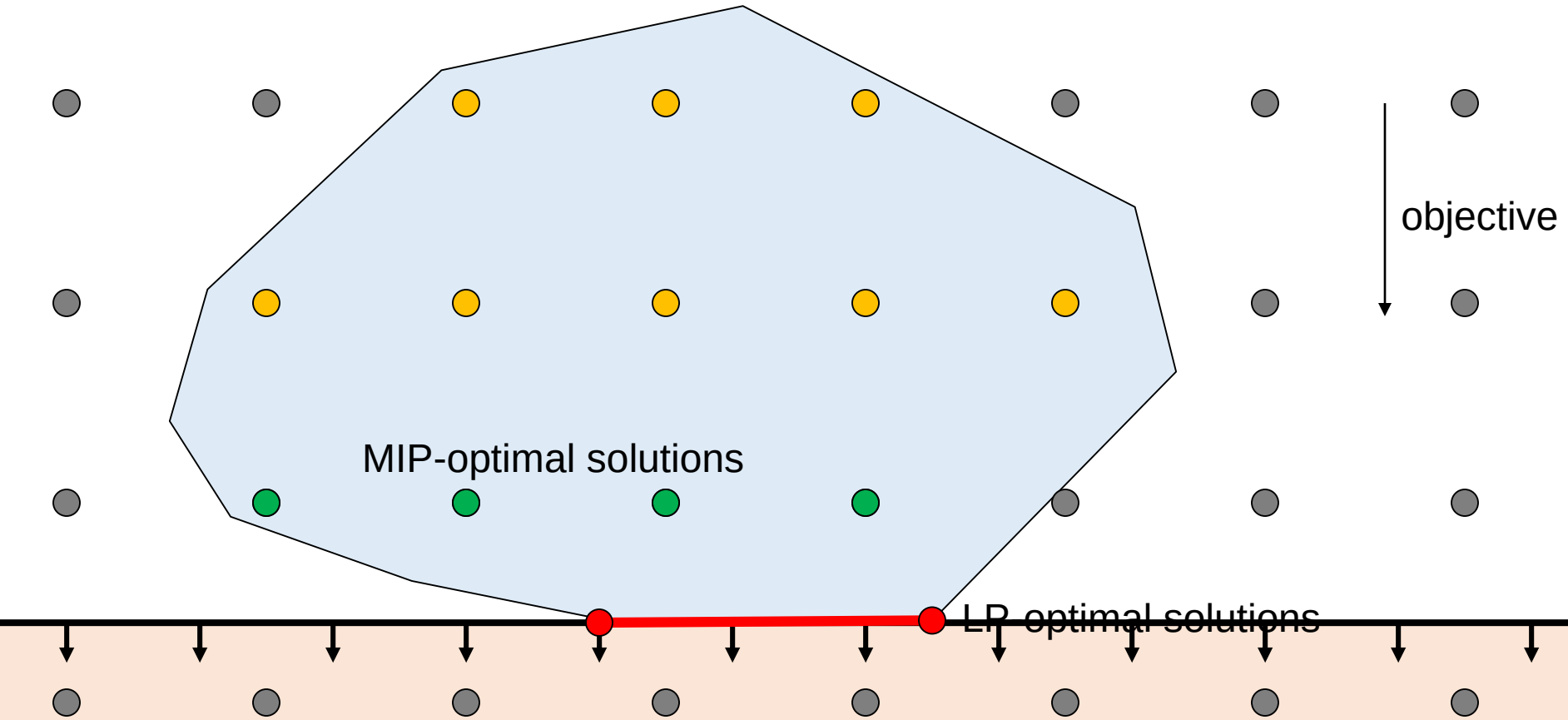


- Iterate until no more improving direction is found

MIP – LP Relaxation



MIP – LP Relaxation



No feasible solutions can be better than an LP optimum

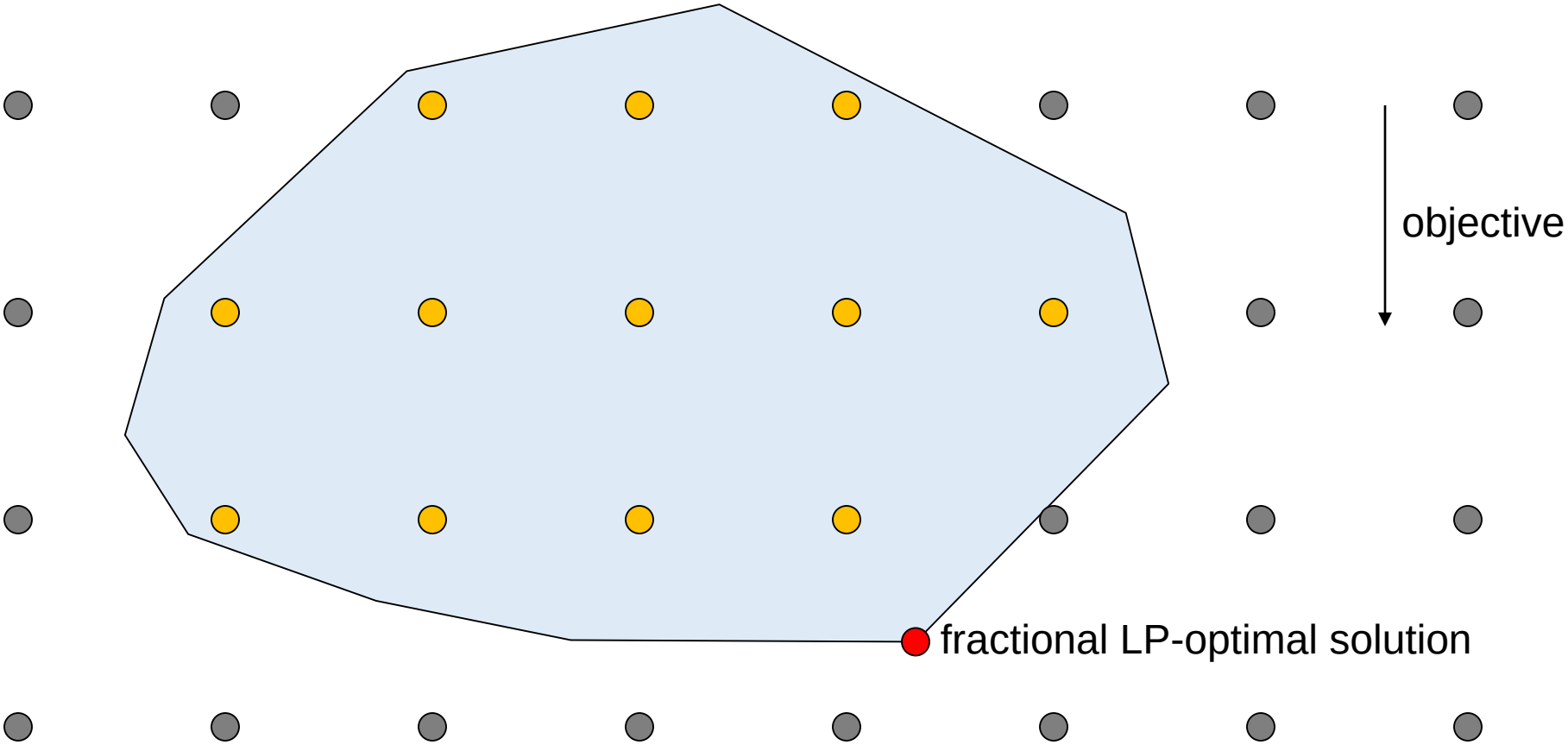
Side Note – Performance Variability

- Measuring performance of MIP is a difficult task
- Seemingly performance neutral changes can have a dramatic impact on the solve time for an individual model
  - Changing the random seed
  - Permuting the columns and rows
    - e.g., store model in *.lp file format and reading it back in
  - Solving model on a different machine
    - Different operating system
    - Different CPU
- Main reason: degeneracy of LP relaxation
  - Most models have multiple optimal solutions to their LP relaxation
  - LP solver's choice of solution is arbitrary
    - Depends on random seed, column and row ordering, tiny differences in numerics
  - Different optimal solution vector leads to different cutting planes, different branching decisions, and thus different search tree
- Conclusion: need big test set to measure performance
  - Use multiple random seeds to artificially increase test set
- More on performance variability: tomorrow

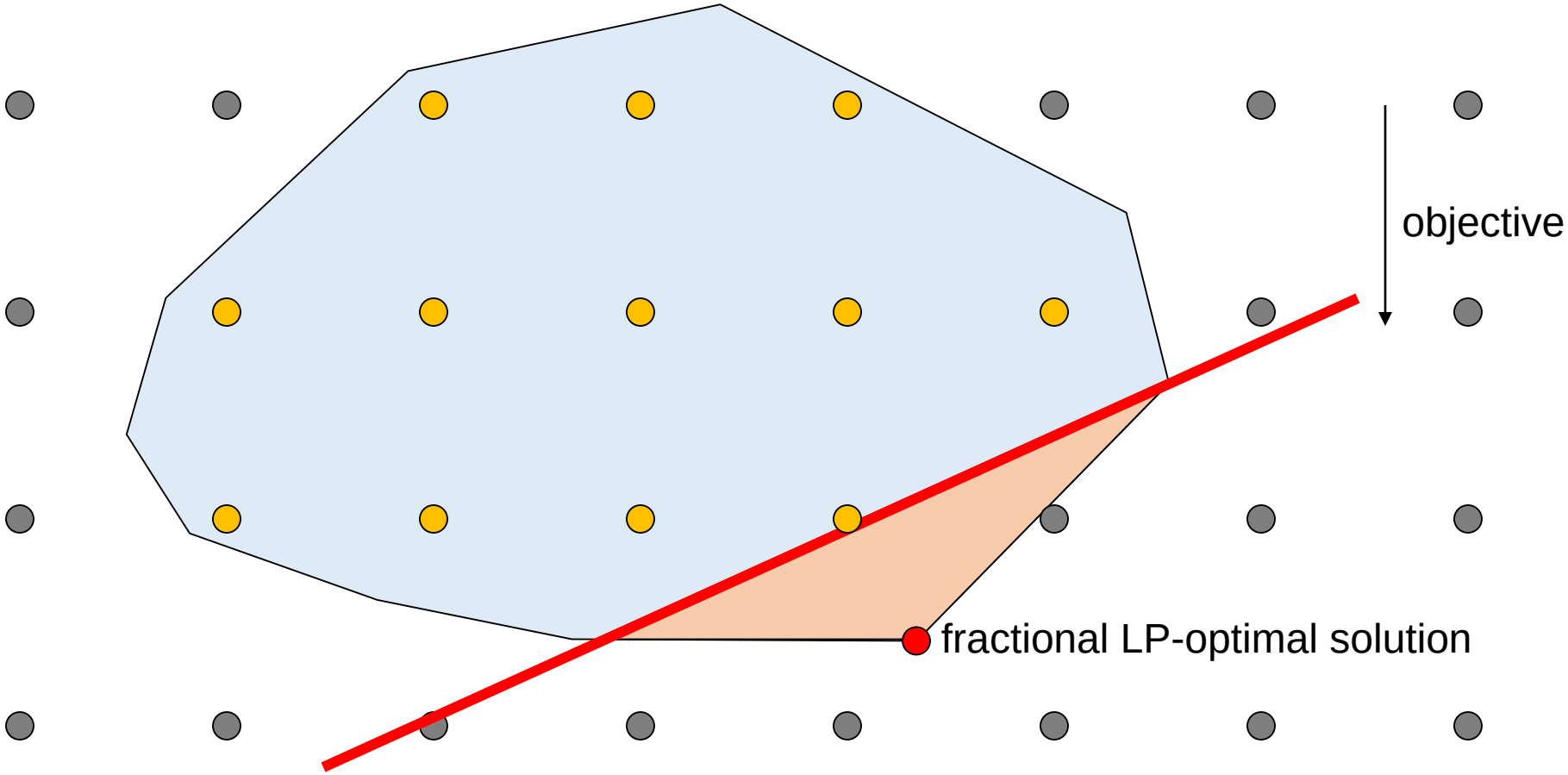
MIP Building Blocks

- Presolve
  - Tighten formulation and reduce problem size
- Solve continuous relaxations
  - Ignoring integrality
  - Gives a bound on the optimal integral objective
- Cutting planes
  - Cut off relaxation solutions
- Branching variable selection
  - Crucial for limiting search tree size
- Primal heuristics
  - Find integer feasible solutions

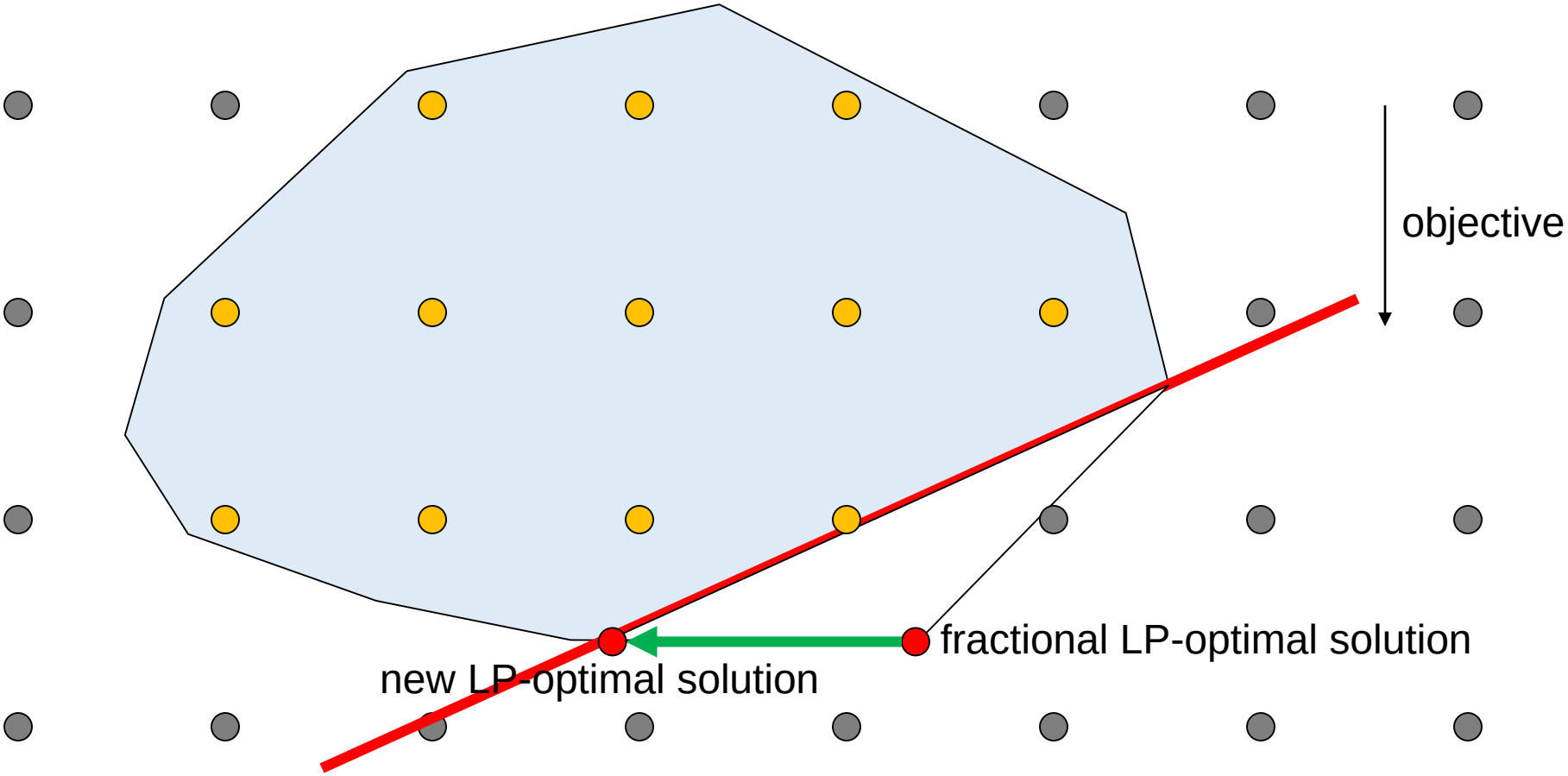
MIP – Cutting Planes



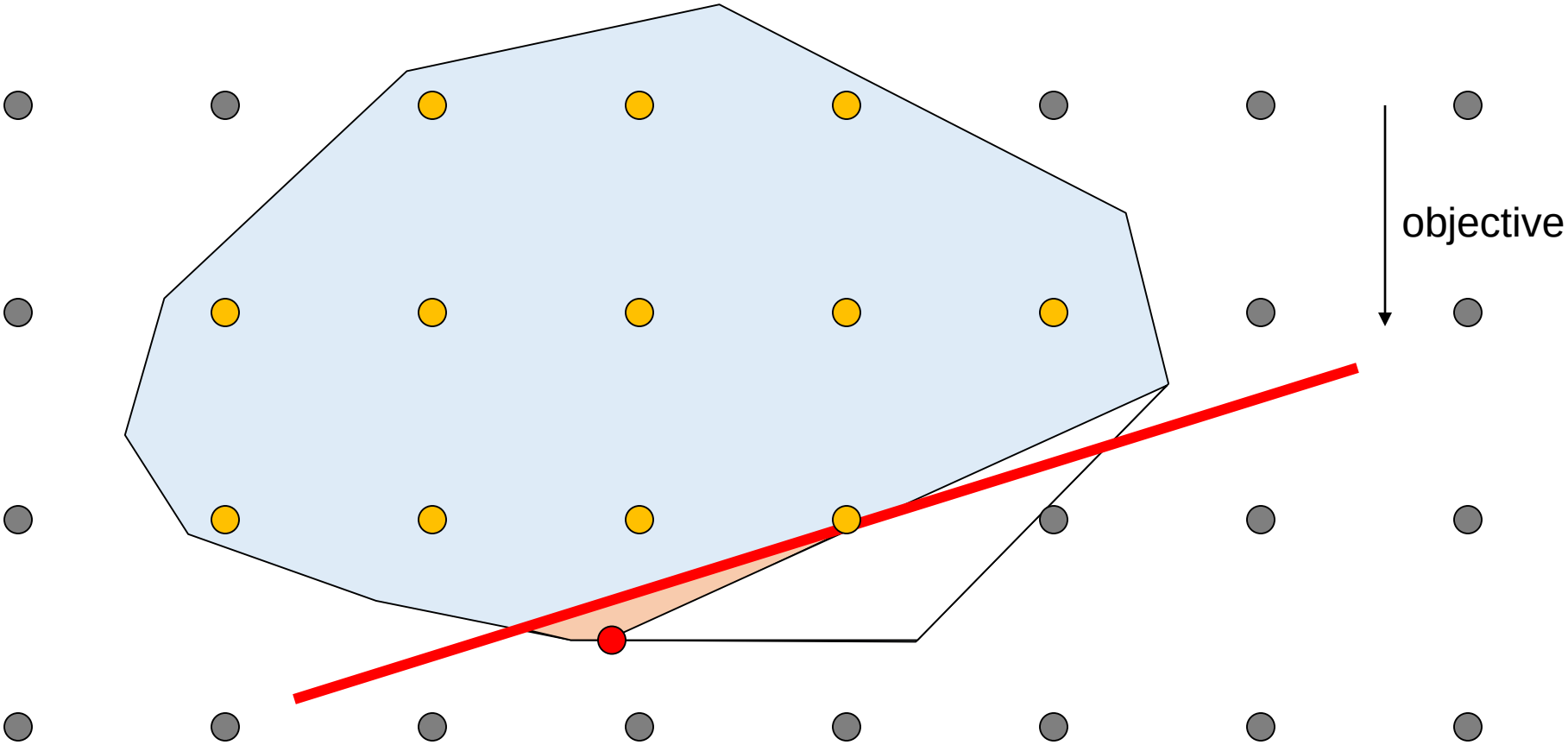
MIP – Cutting Planes



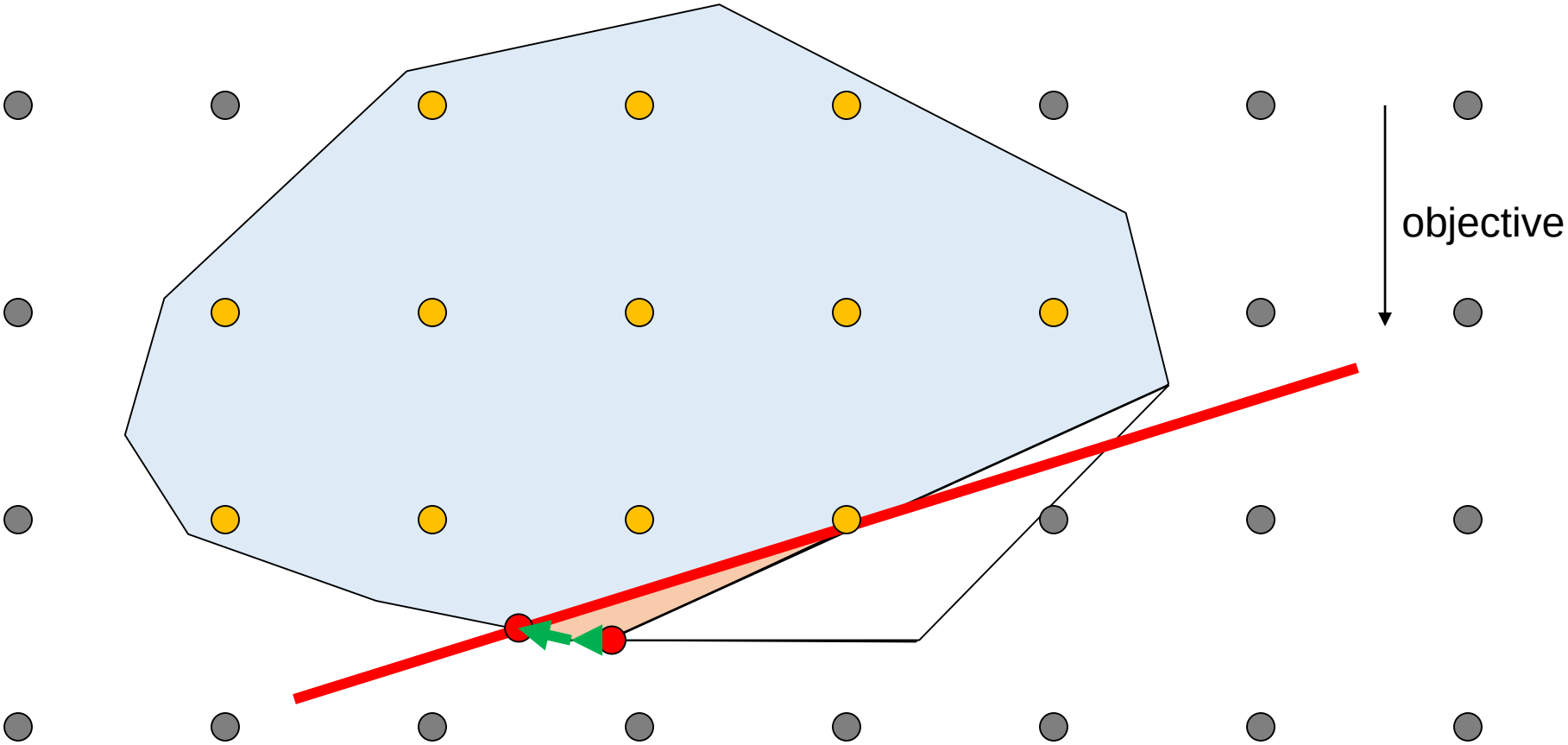
MIP – Cutting Planes



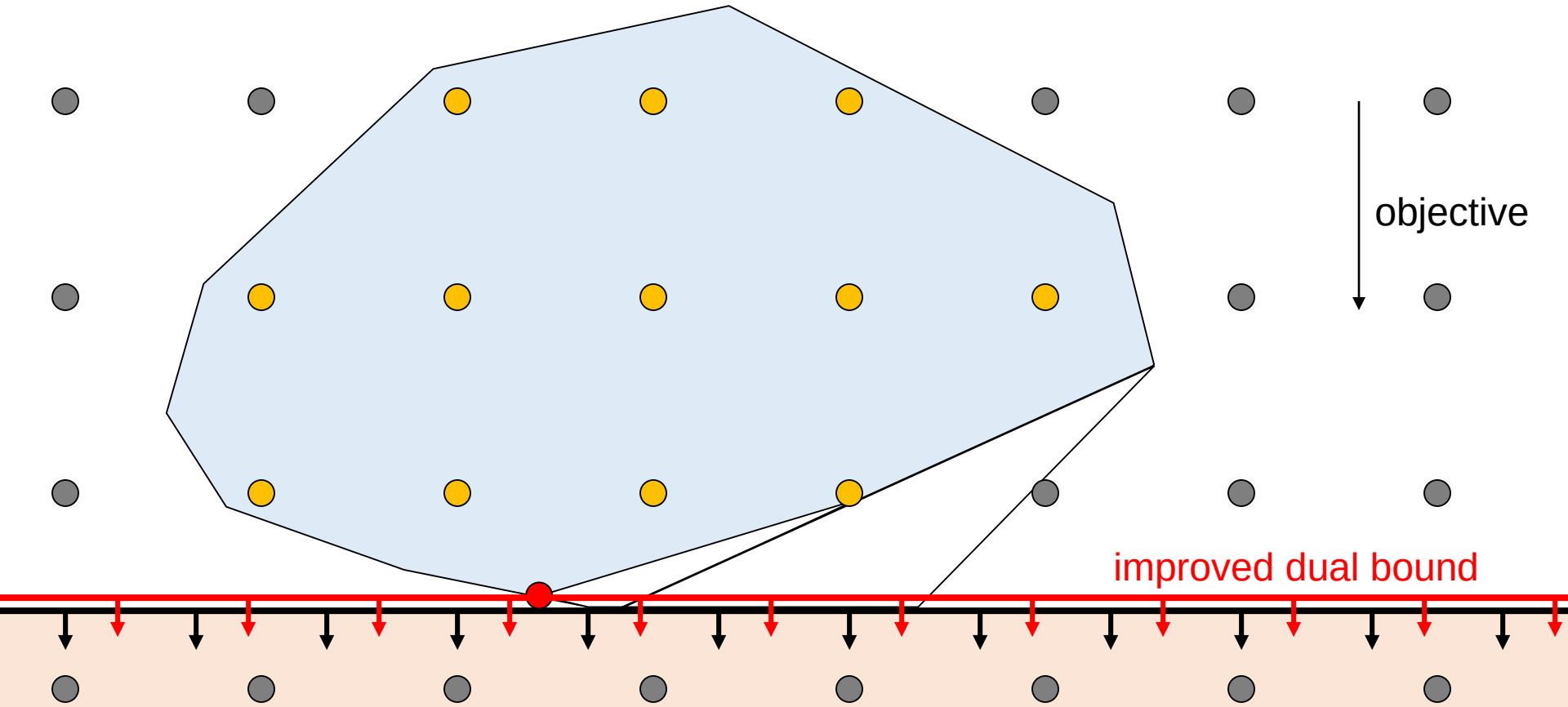
MIP – Cutting Planes



MIP – Cutting Planes



MIP – Cutting Planes



No feasible solutions can be better than an LP optimum

Cutting Planes – Overview

- General-purpose cutting planes
  - Gomory mixed integer cuts
  - Mixed Integer Rounding (MIR) cuts
  - Flow cover cuts
  - Lift-and-project (L&P) cuts
  - Zero-half and mod-k cuts
  - ...
- Structural cuts
  - Implied bound cuts
  - Knapsack cover cuts
  - GUB cover cuts
  - Clique cuts
  - Multi-commodity-flow (MCF) cuts
  - Flow path cuts
  - ...

Chvatal-Gomory Cuts

- Consider a rational polyhedron

  $$P = \{x \in R^n \mid Ax \leq b,\ x \geq 0\},\ A \in \mathbb{Q}^{m \times n},\ b \in \mathbb{Q}^m$$

- We want to find the integer hull $P_I = \text{conv}\{x \in \mathbb{Z}^n \mid Ax \leq b,\ x \geq 0\}$
- Chvatal-Gomory procedure:
  1. Choose non-negative multipliers $\lambda \in \mathbb{R}^m_{\geq 0}$
  2. Aggregated inequality $\lambda^T A x \leq \lambda^T b$ is valid for P because $\lambda \geq 0$
  3. Relaxed inequality $\lfloor \lambda^T A \rfloor x \leq \lambda^T b$ is still valid for P because $x \geq 0$
  4. Rounded Inequality $\lfloor \lambda^T A \rfloor x \leq \lfloor \lambda^T b \rfloor$ is still valid for P_I because $x \in \mathbb{Z}^n$
- CG procedure suffices to generate all non-dominated valid inequalities for P_I in a finite number of iterations!
  - $P^{(0)} = P$
  - $P^{(k)} = P^{(k-1)} \cap \{\text{CG cuts for } P^{(k-1)}\}$: k-th CG closure of P - is a polyhedron!
  - CG rank of a valid inequality for P_I: minimum k s.t. inequality is valid for $P^{(k)}$

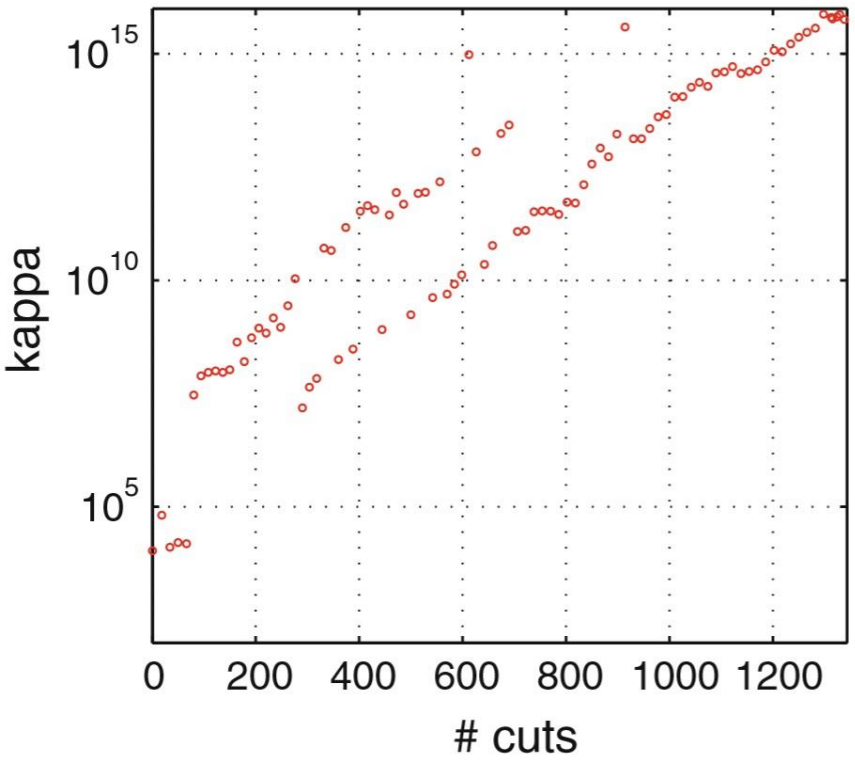
Gomory Fractional Cuts

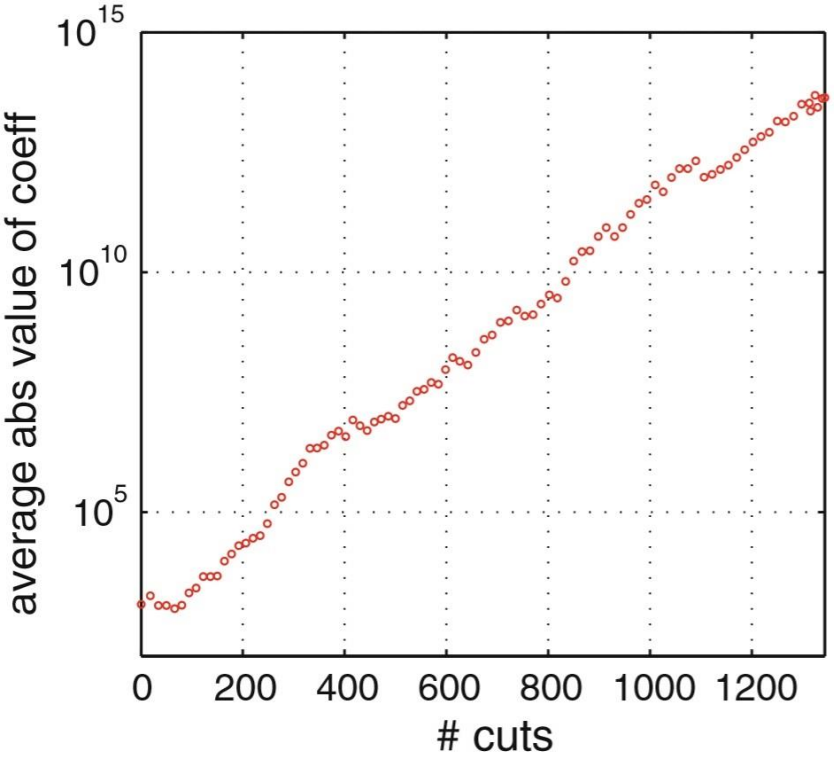
- How to find the multipliers λ in the Chvatal–Gomory procedure?
- Read them from an optimal simplex tableau!

$$x_i + \lfloor (A_B^{-1})_i . A_N \rfloor \, x_N \leq \lfloor (A_B^{-1})_i . b \rfloor$$

- Note: new slack variable is integer
  - Hence, procedure can be iterated
- Gomory (1963): These cuts lead to an integral LP solution in a finite number of iterations.
- Conclusion:
  - Integer programs with rational data can be solved by pure cutting plane algorithms in finite time
  - What about mixed integer programs?
  - And is this result useful in practice?

Gomory Cuts – Numerics





- Pictures stolen from Zanette, Fischetti and Balas (2011):

  "Lexicography and degeneracy: can a pure cutting plane algorithm work?"

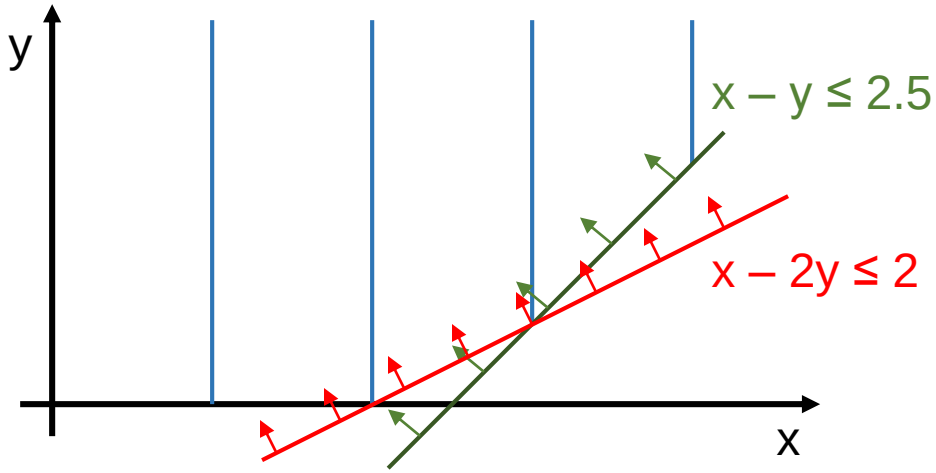
- Evolution of optimal basis matrix condition number and average absolute coefficient values of Gomory cuts for model "sentoy"
  - Gomory fractional cuts added in rounds

Mixed Integer Rounding Cuts

- Consider $S := \{(x,y) \in \mathbb{Z} \times \mathbb{R}_{\geq 0} \mid x - y \leq b\}$.

  Then,
  $$x - \frac{1}{1 - f_0} y \leq \lfloor b \rfloor$$
  is valid for S with $f_0 := b - \lfloor b \rfloor$.

- Example: $x - y \leq 2.5$
- MIR cut: $x - 2y \leq 2$



Mixed Integer Rounding Cuts

- Consider $S := \{(x,y) \in \mathbb{Z} \times \mathbb{R}_{\geq 0} \mid x - y \leq b\}$.

  Then,

  $$x - \frac{1}{1 - f_0} y \leq \lfloor b \rfloor$$

  is valid for S with $f_0 := b - \lfloor b \rfloor$.

- Consider $S := \{(x,y) \in \mathbb{Z}^p_{\geq 0} \times \mathbb{R}^q_{\geq 0} \mid ax + dy \leq b\}$.

  Then,

  $$\sum \left(\lfloor a_i \rfloor + \frac{max\{f_i - f_0, 0\}}{1 - f_0} \right) x_i + \sum \left(\frac{min\{d_j, 0\}}{1 - f_0} \right) y_j \leq \lfloor b \rfloor$$

  is valid for S with $f_i := a_i - \lfloor a_i \rfloor$, $f_0 := b - \lfloor b \rfloor$.

Mixed Integer Rounding Cuts

- General idea just like Chvatal-Gomory cuts:
  1. Choose non-negative multipliers $\lambda \in \mathbb{R}^m_{\geq 0}$
  2. Aggregated inequality $\lambda^T Ax \leq \lambda^T b$ is valid for P because $\lambda \geq 0$
  3. Apply MIR formula to aggregated inequality to produce cutting plane
- Cut separation procedure of Marchand and Wolsey (1998, 2001):
  1. Start with one constraint of the problem (do this for each one), call this the "current aggregated inequality"
  2. Apply MIR procedure to current aggregated inequality
     (a) Complement variables if LP solution is closer to upper bound
     (b) For each a_j in constraint and each of $\delta \in \{1,2,4,8\}$ divide the constraint by $\delta|a_j|$
     (c) Apply MIR formula to resulting scaled constraint
     (d) Choose most violated cut from this set of MIR cuts
     (e) Check if complementing one more (or one less) variable yields larger violation
  3. If no violated cut was found (and did not yet reach aggregation limit):
     (a) Add another problem constraint to the current aggregated inequality such that a continuous variable with LP value not at a bound is canceled
     (b) Go to 2

Gomory Mixed Integer Cuts

- Just an alternative way to aggregate constraints
- Read them from an optimal simplex tableau:
  - Let i be a basis index with $x_i^* \notin \mathbb{Z}$
  - Choose $\lambda^T = (A_B^{-1})_i$.
  - Resulting aggregated inequality: $x_i + (A_B^{-1})_i . A_N x_N \leq (A_B^{-1})_i . b$
- Apply MIR formula on resulting aggregated inequality
- In theory, always produces a violated cutting plane
- Practical issues:
  - Gomory Mixed Integer Cuts can be pretty dense
  - Numerics (in particular for higher rank cuts) can be very challenging
- But:
  - If done right, GMICs (together with MIRs) are currently the most important cutting planes in practice

Knapsack Cover Cuts

- A (binary) knapsack is a constraint $ax \leq b$ with
  - $a_i \geq 0$ the weight of item i, $i = 1,\ldots,n$
  - $b \geq 0$ the capacity of the knapsack
- An index set $C \subseteq \{1,\ldots,n\}$ is called a *cover*, if $\sum_{i \in C} a_i > b$
- A cover C entails a cover inequality

$$\sum_{i \in C} x_i \leq |C| - 1$$

- Interesting for cuts: minimal covers

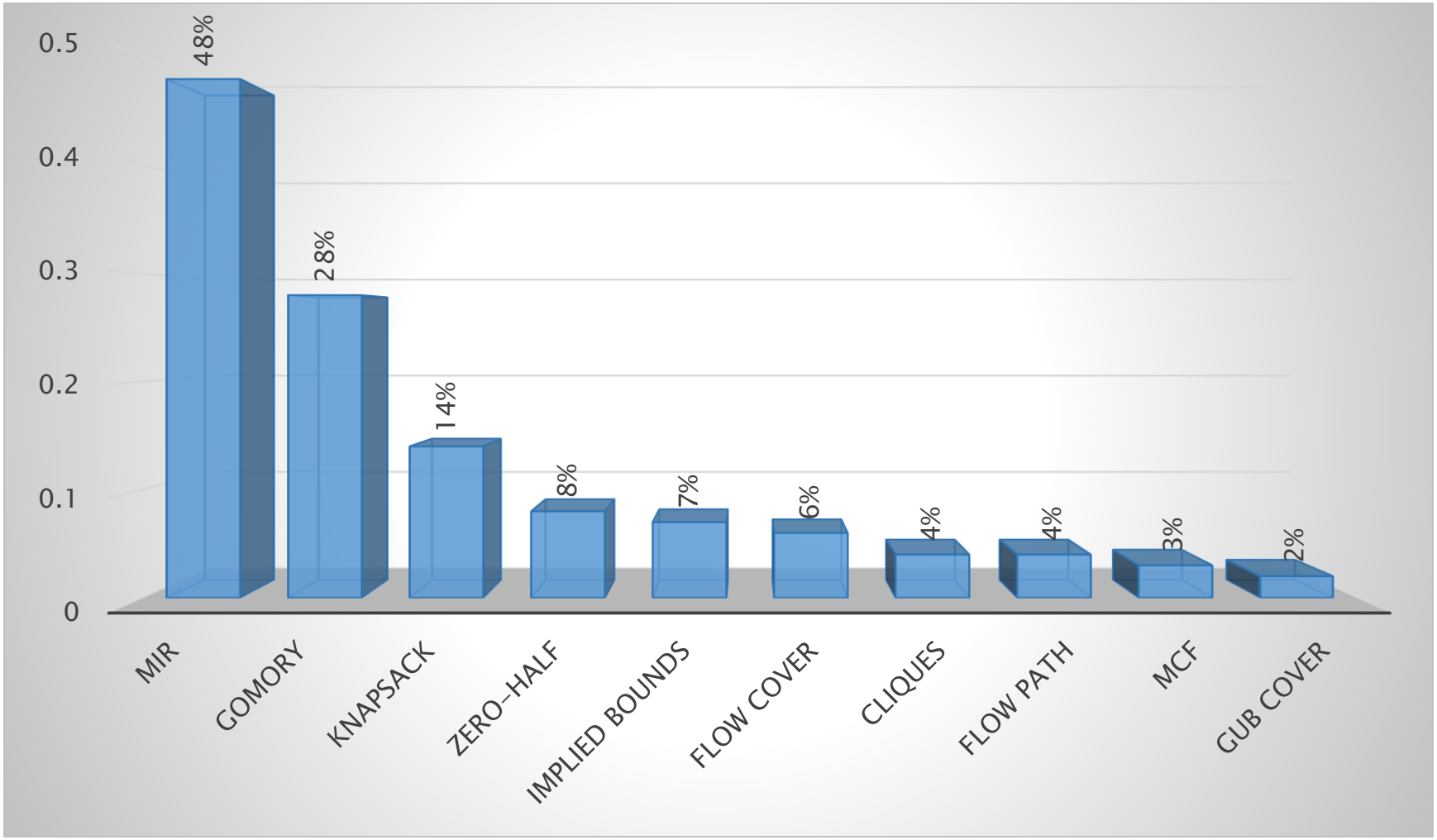
$$\sum_{i \in C} a_i > b \quad \text{and} \quad \sum_{i \in C'} a_i \leq b \ \text{ for all } \ C' \subsetneq C$$

Knapsack Cover Cuts – Example

- Consider knapsack $3x_1 + 5x_2 + 8x_3 + 10x_4 + 17x_5 \leq 24$, $x \in \{0,1\}^5$
- A minimal cover is $C = \{1,2,3,4\}$
- Resulting cover inequality: $x_1 + x_2 + x_3 + x_4 \leq 3$

- Lifting
  - If $x_5 = 1$, then $x_1 + x_2 + x_3 + x_4 \leq 1$
  - Hence, $x_1 + x_2 + x_3 + x_4 + 2x_5 \leq 3$ is valid
  - Need to solve knapsack problem $\alpha_j := d_0 - \max\{dx \mid ax \leq b - a_j\}$ to find lifting coefficient for variable x_j
    - Use dynamic programming to solve knapsack problem

Cutting Planes – Performance

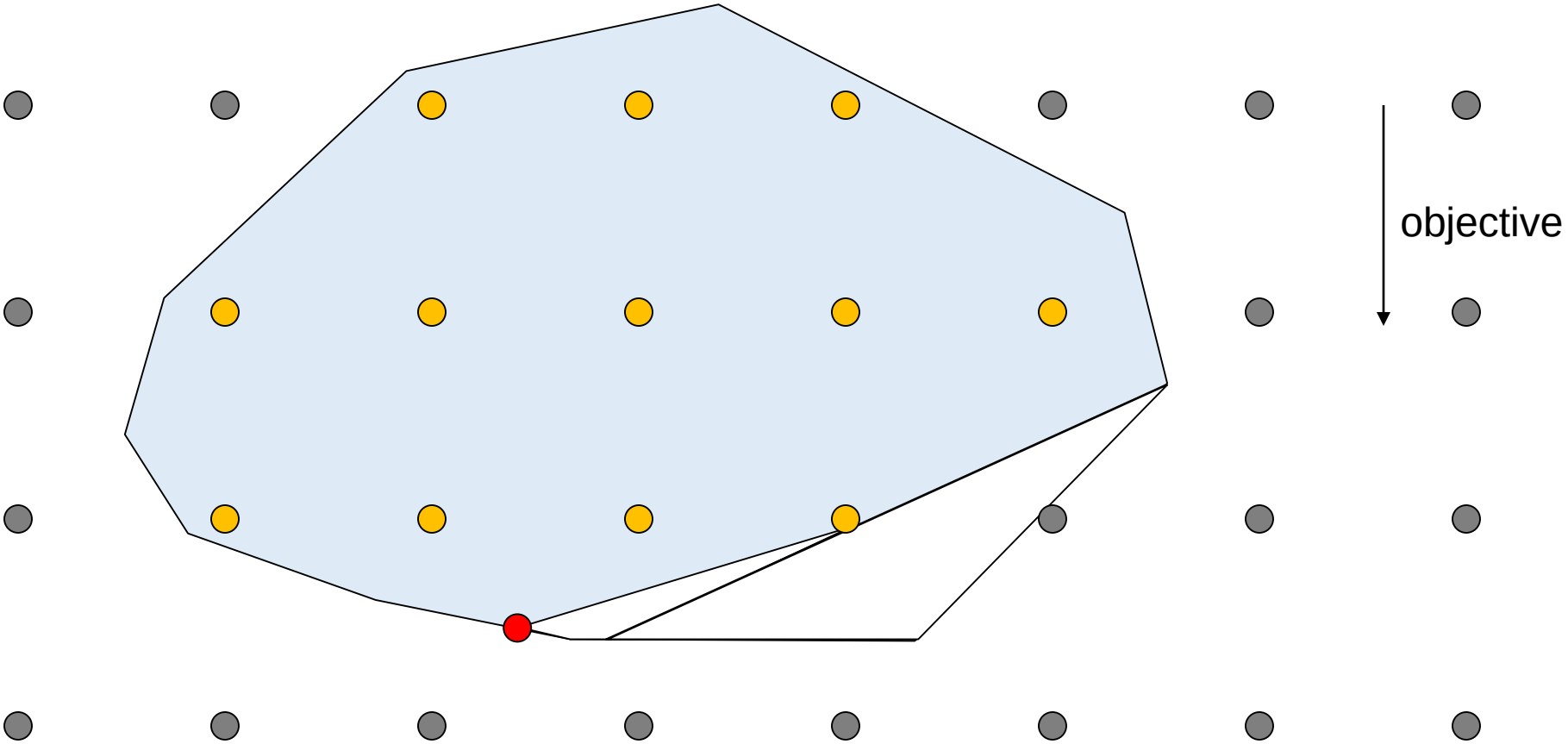


Achterberg and Wunderling: "Mixed Integer Programming: Analyzing 12 Years of Progress" (2013)
benchmark data based on CPLEX 12.5

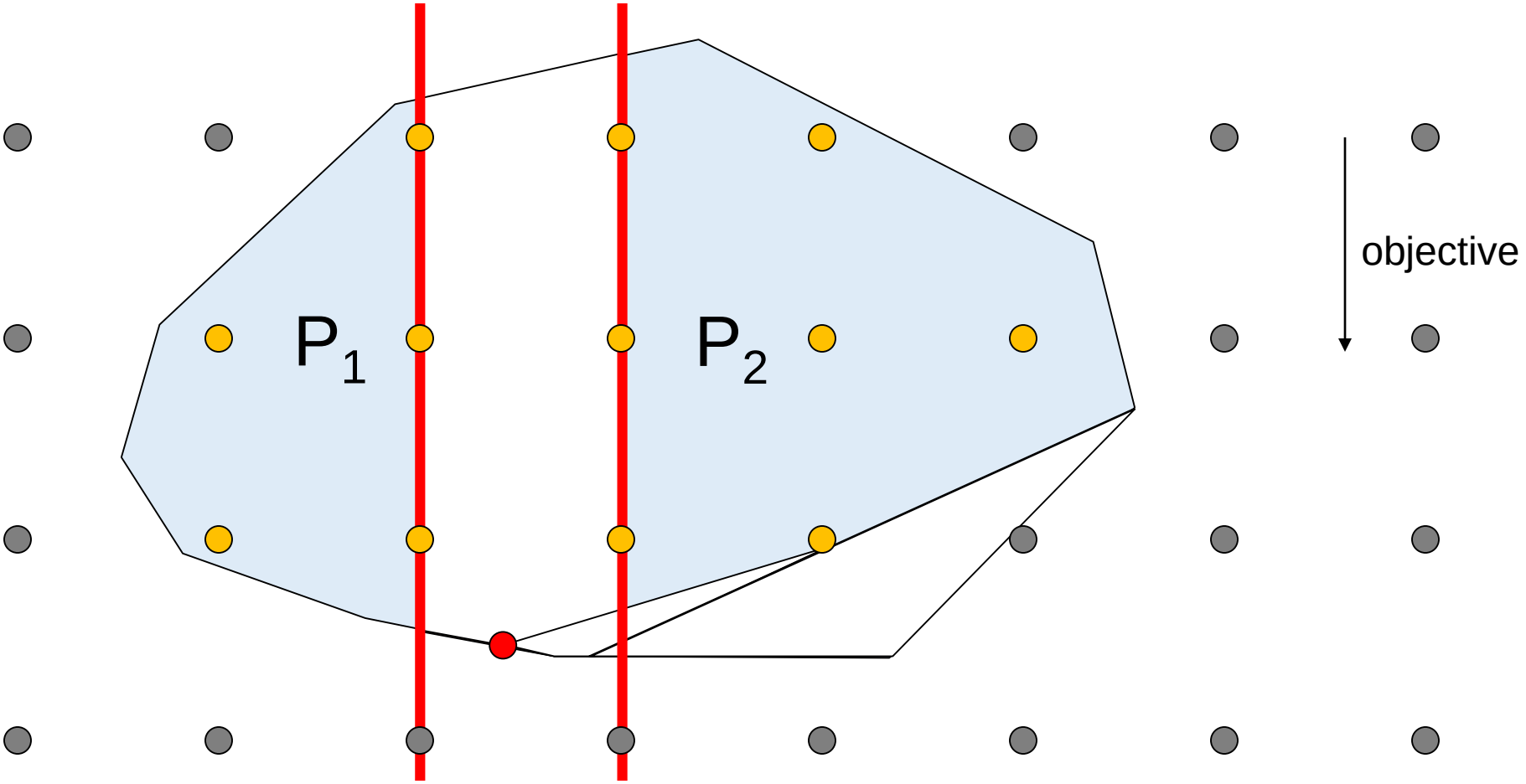
MIP Building Blocks

- Presolve
  - Tighten formulation and reduce problem size
- Solve continuous relaxations
  - Ignoring integrality
  - Gives a bound on the optimal integral objective
- Cutting planes
  - Cut off relaxation solutions
- Branching variable selection
  - Crucial for limiting search tree size
- Primal heuristics
  - Find integer feasible solutions

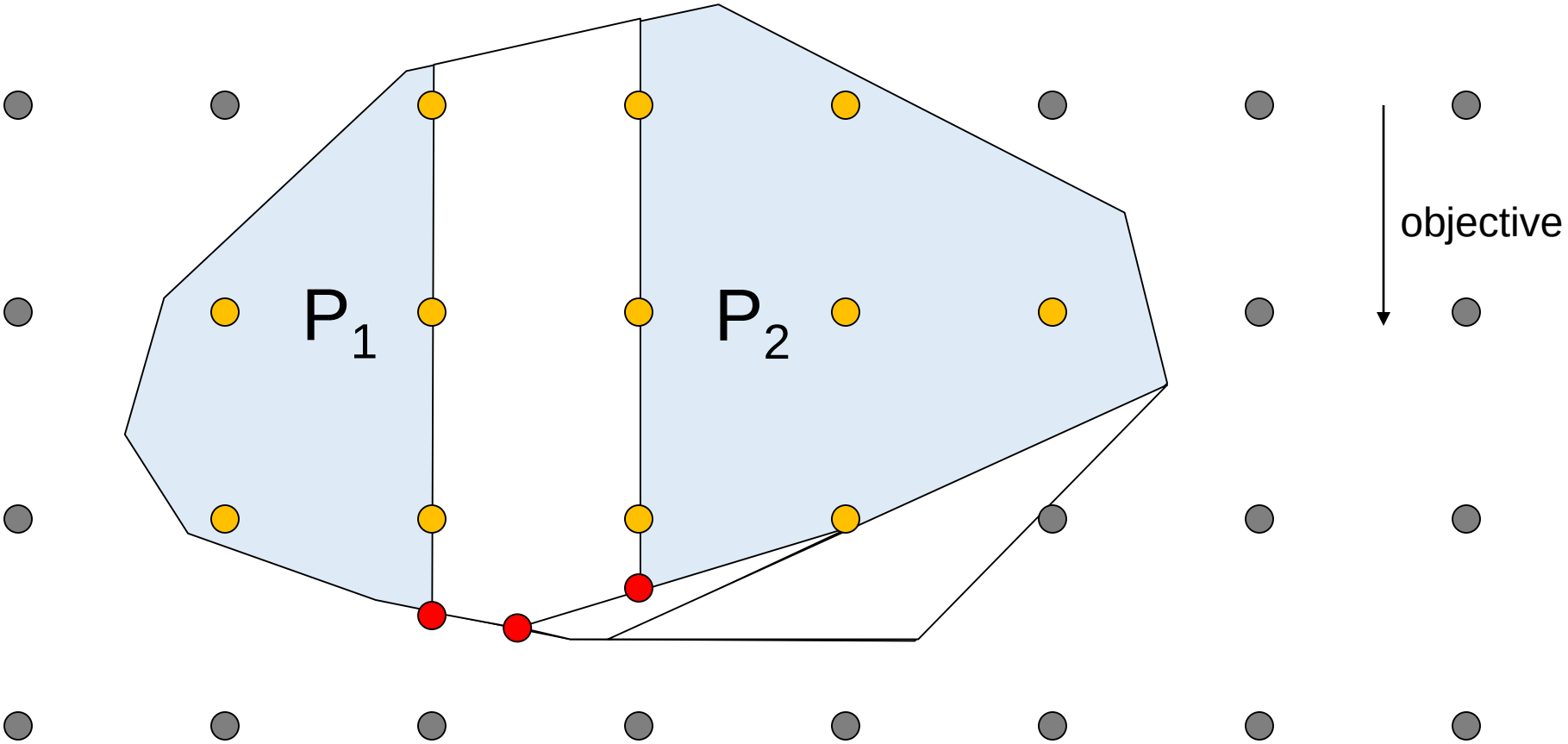
MIP – Branching



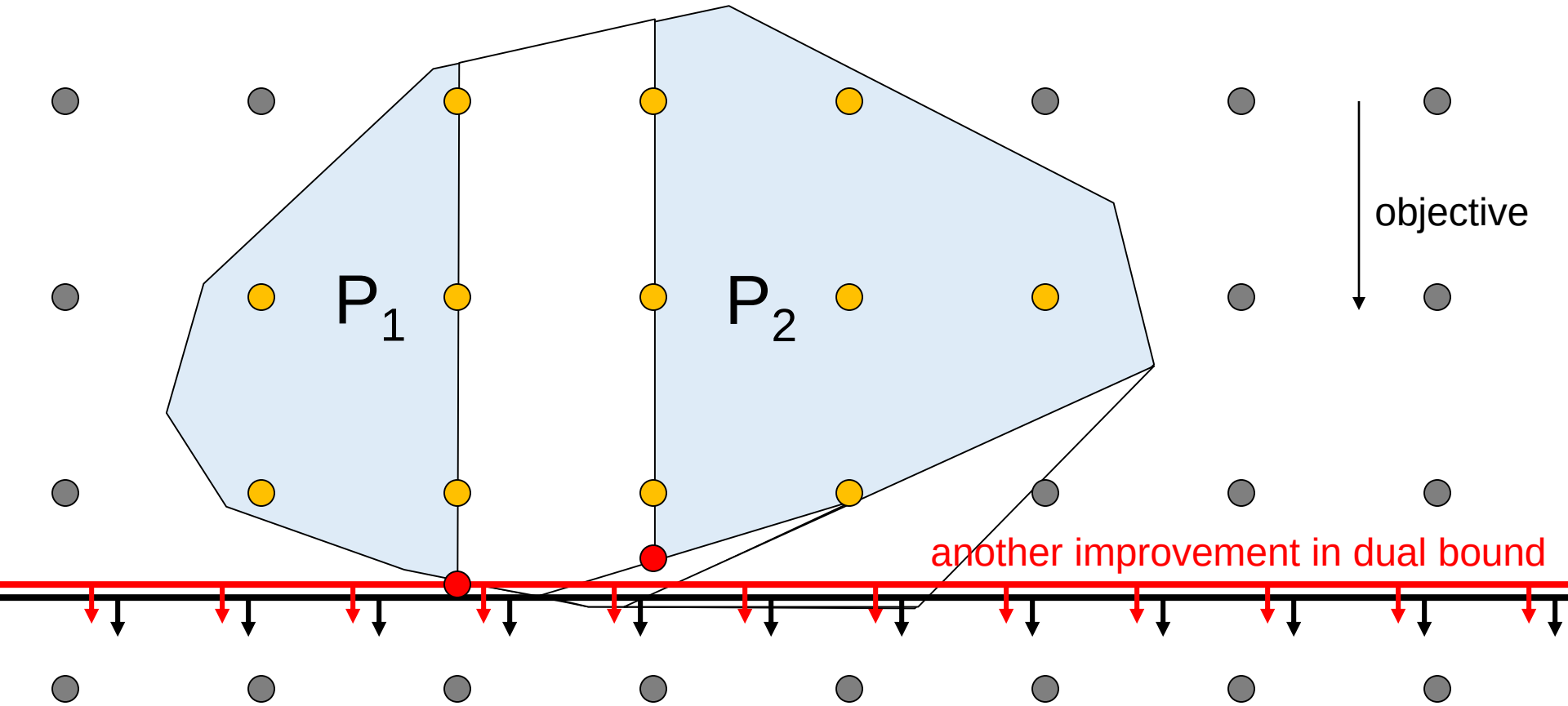
MIP – Branching



MIP – Branching



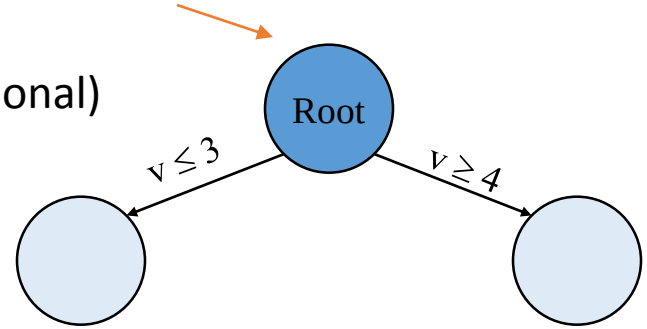
MIP – Branching



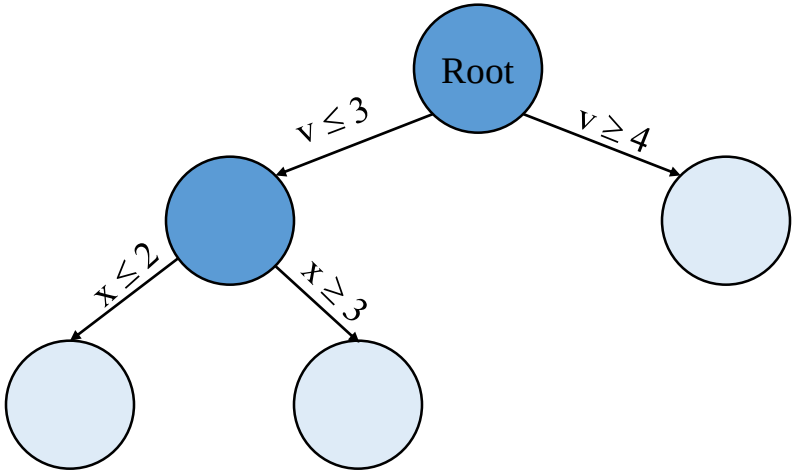
LP based Branch-and-Bound

Solve LP relaxation:

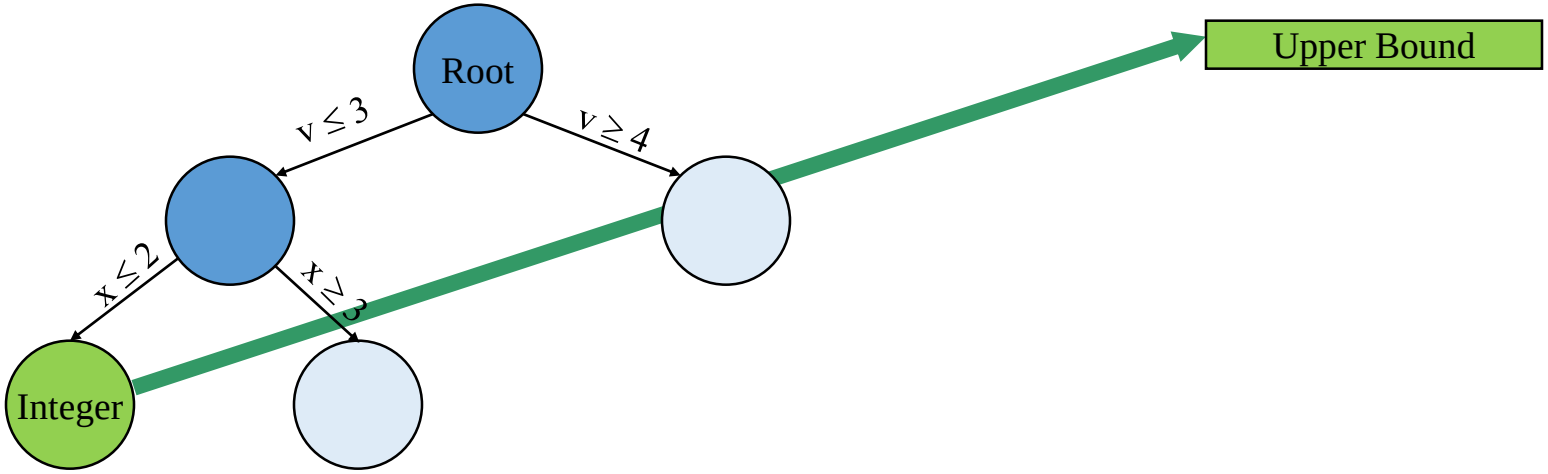
v=3.5 (fractional)



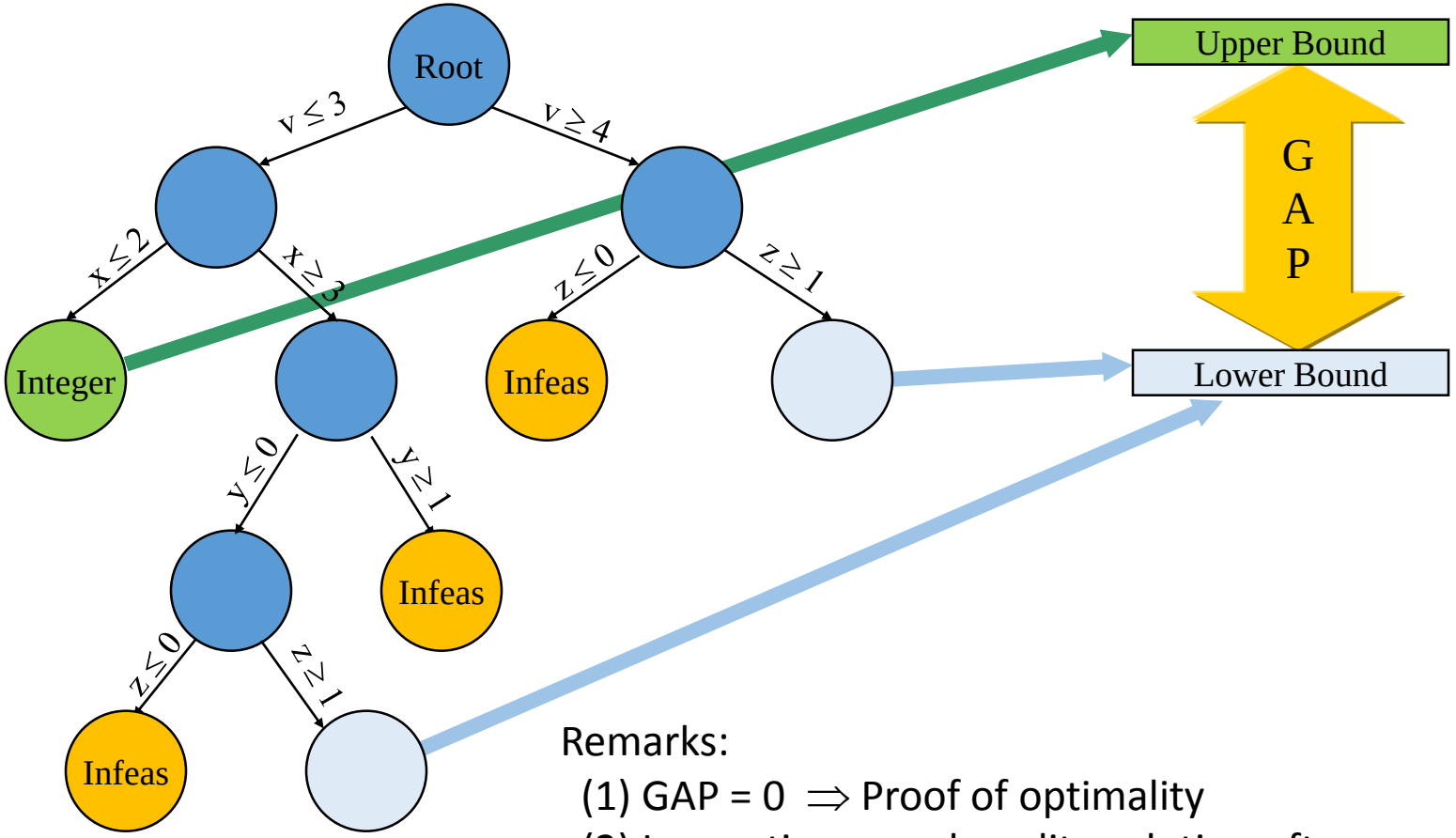
LP based Branch-and-Bound



LP based Branch-and-Bound



LP based Branch-and-Bound

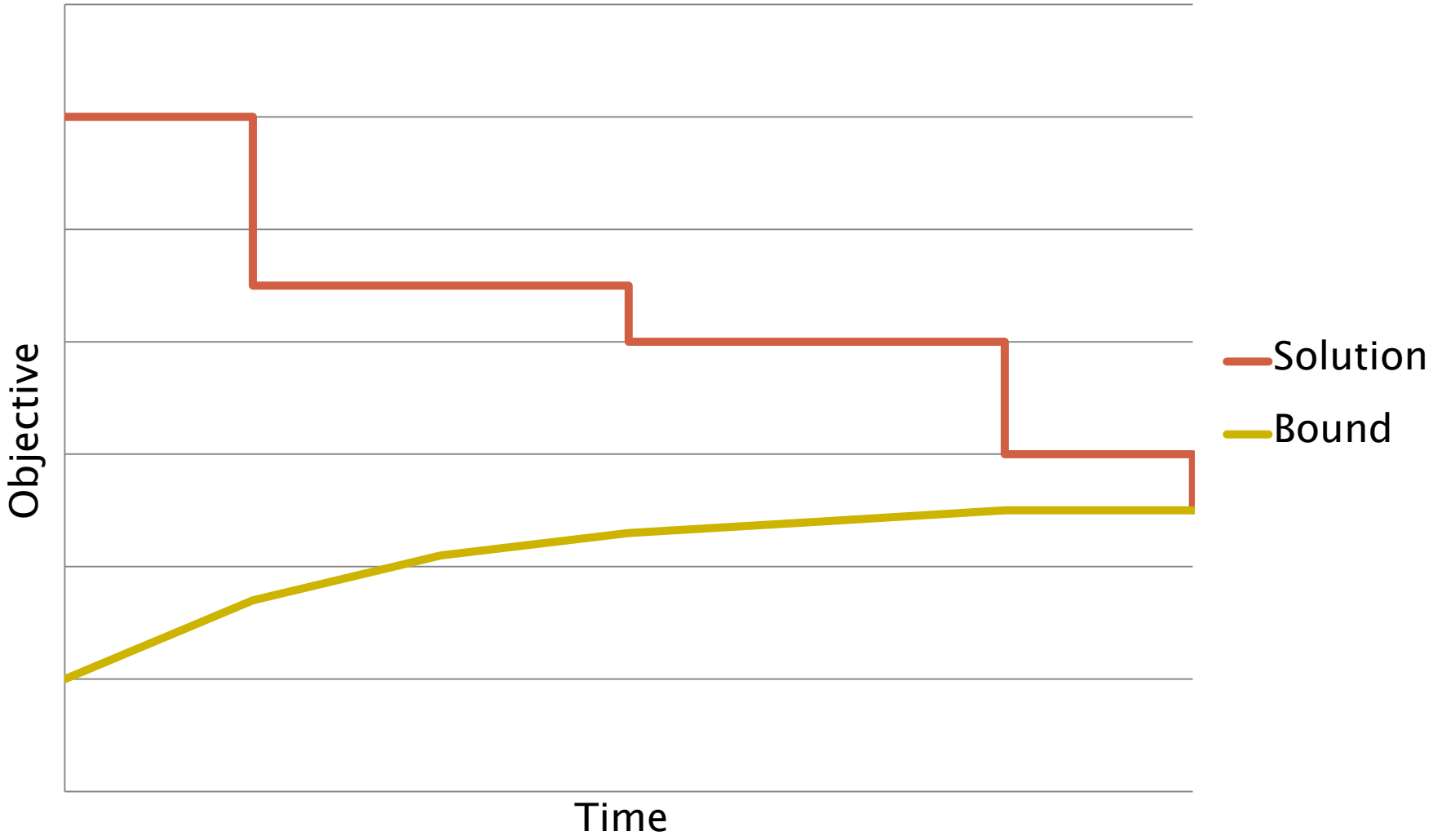


Remarks:

(1) GAP = 0 $\Rightarrow$ Proof of optimality

(2) In practice: good quality solution often enough

Solving a MIP Model



Branching Variable Selection

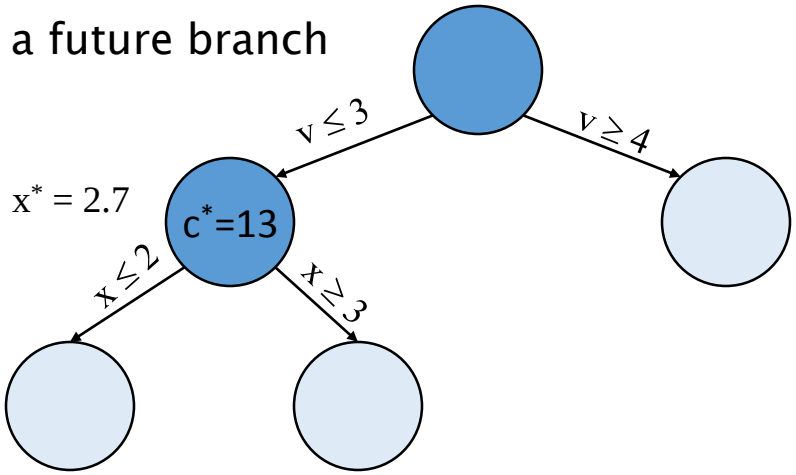
- Given a relaxation solution x^*
  - Branching candidates:
    - Integer variables x_j that take fractional values
      - $x_j = 3.7$ produces two child nodes ($x \leq 3$ or $x \geq 4$)
  - Need to pick a variable to branch on
    - Choice is crucial in determining the size of the overall search tree

Branching Variable Selection

- What's a good branching variable?
  - Superb: fractional variable infeasible in both branch directions
  - Great: infeasible in one direction
  - Good: both directions move the objective
- Expensive to predict which branches lead to infeasibility or big objective moves
  - Strong branching
    - Truncated LP solve for every possible branch at every node
    - Rarely cost effective
  - Need a quick estimate

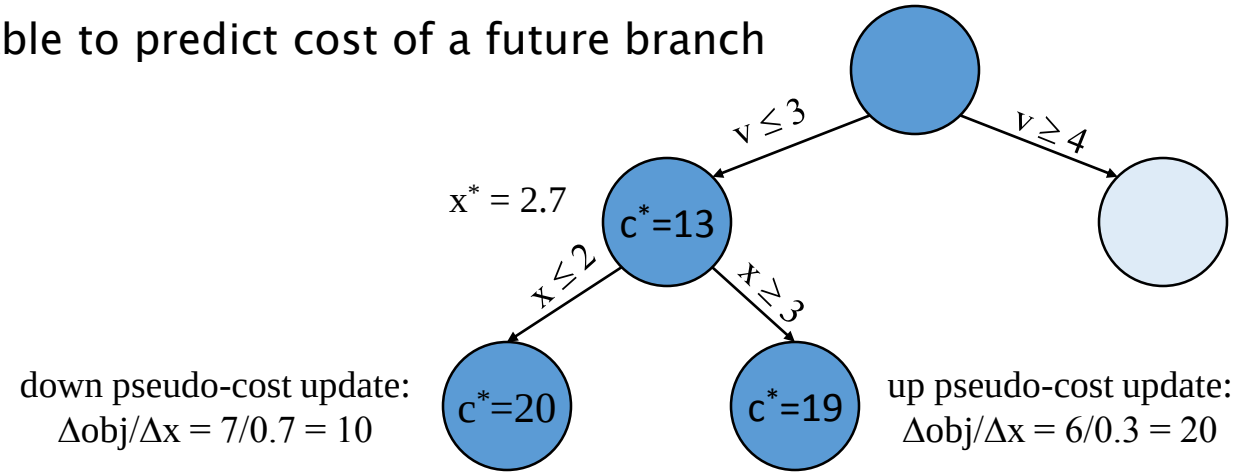
Pseudo-Costs

- Use historical data to predict impact of a branch:
  - Record $\text{cost}(x_j) = \Delta\text{obj} \,/\, \Delta x_j$ for each branch
  - Store results in a pseudo-cost table
    - Two entries per integer variable
      - Average down cost
      - Average up cost
  - Use table to predict cost of a future branch



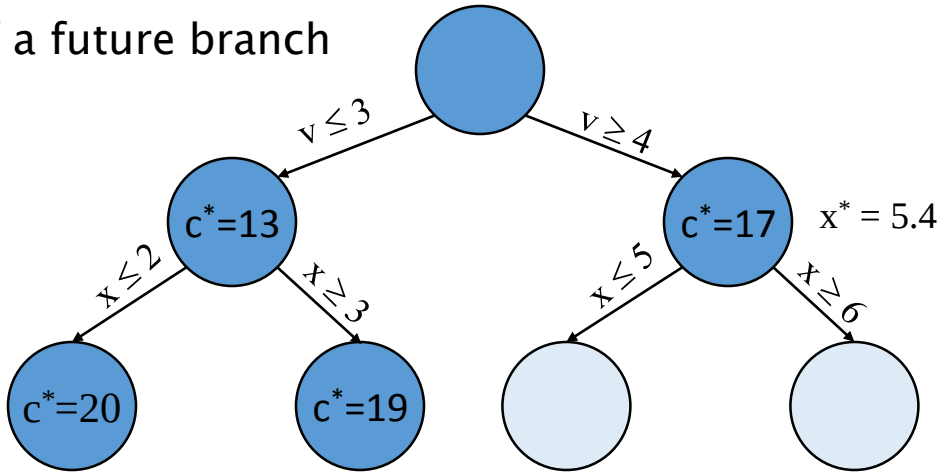
Pseudo-Costs

- Use historical data to predict impact of a branch:
  - Record $\text{cost}(x_j) = \Delta\text{obj} \,/\, \Delta x_j$ for each branch
  - Store results in a pseudo-cost table
    - Two entries per integer variable
      - Average down cost
      - Average up cost
  - Use table to predict cost of a future branch



Pseudo-Costs

- Use historical data to predict impact of a branch:
  - Record $\text{cost}(x_j) = \Delta\text{obj} \,/\, \Delta x_j$ for each branch
  - Store results in a pseudo-cost table
    - Two entries per integer variable
      - Average down cost
      - Average up cost
  - Use table to predict cost of a future branch



pseudo costs:
downcost(x) = 10
upcost(x) = 20

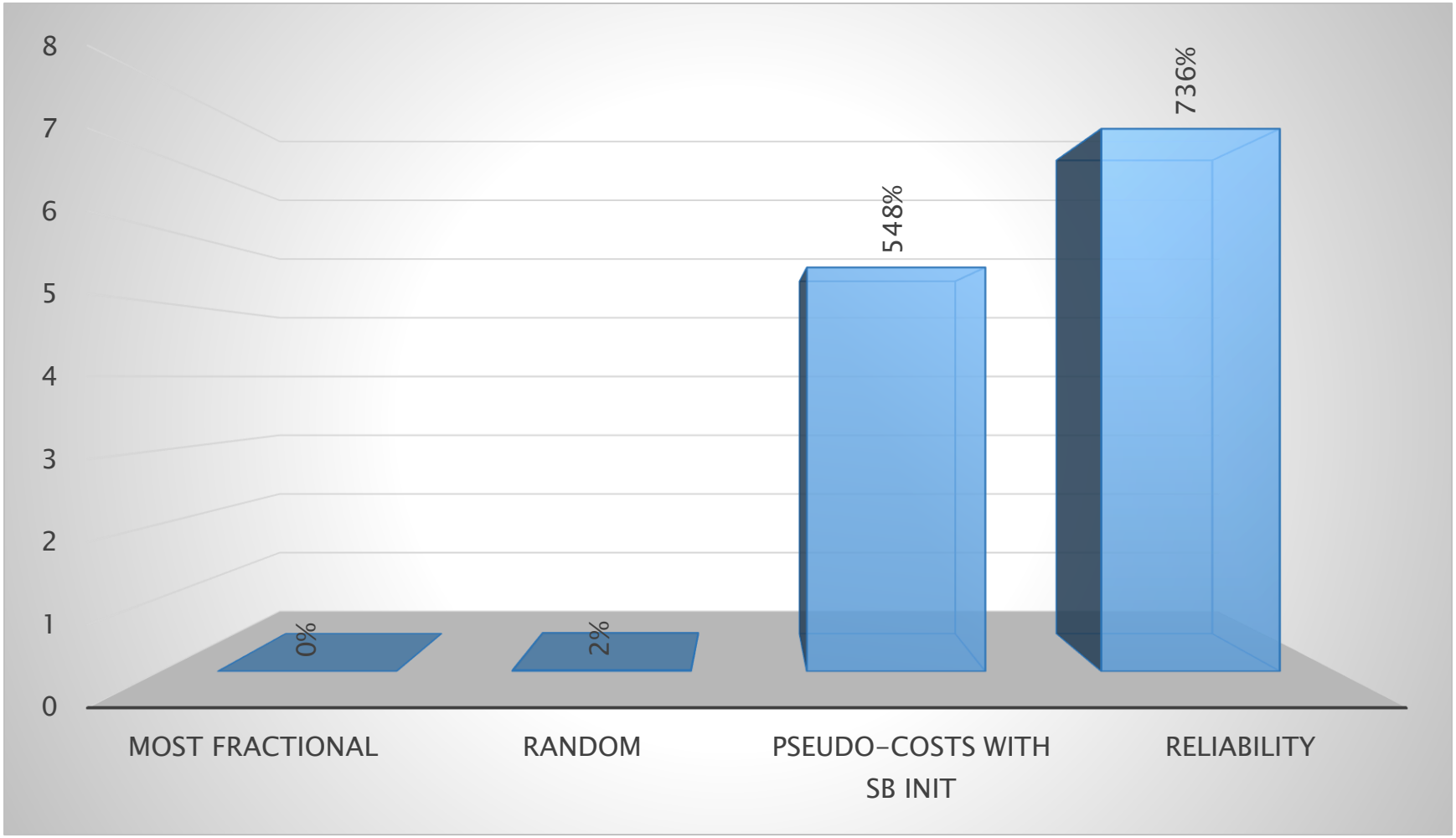
down estimate:
c' = 17 + 0.4 · 10 = 21

up estimate:
c' = 17 + 0.6 · 20 = 29

Pseudo-Costs Initialization

- What do you do when there is no history?
  - E.g., at the root node
- Initialize pseudo-costs [Linderoth & Savelsbergh, 1999]
  - Always compute up/down cost (using strong branching) for new fractional variables
    - Initialize pseudo-costs for every fractional variable at root
- Reliability branching [Achterberg, Koch & Martin, 2005]
  - Do not rely on historical data until pseudo-cost for a variable has been recomputed r times

Branching Rules – Performance

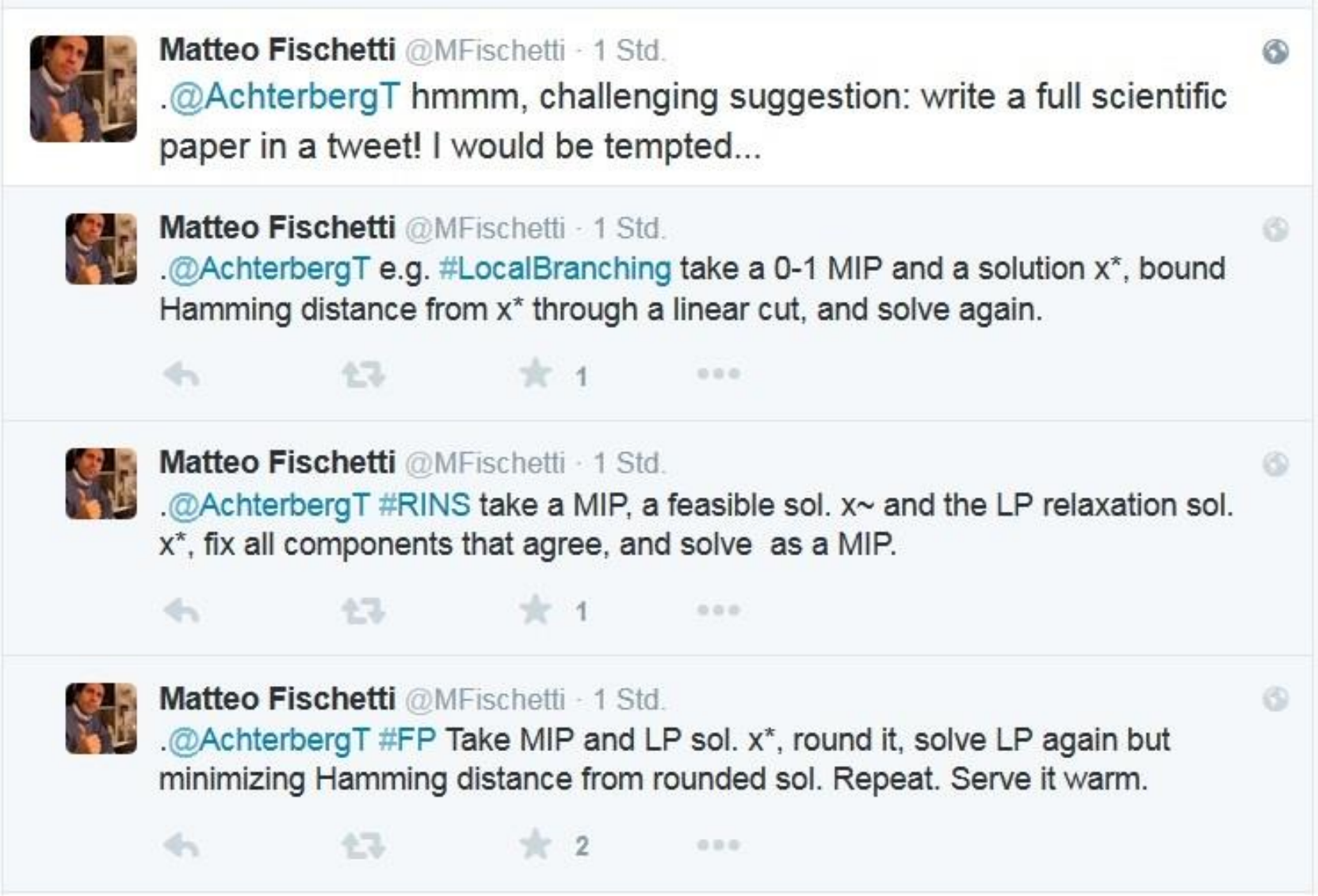


Achterberg and Wunderling: "Mixed Integer Programming: Analyzing 12 Years of Progress" (2013)
benchmark data based on CPLEX 12.5
Achterberg, Koch, and Martin: "Branching Rules Revisited" (2005)

MIP Building Blocks

- Presolve
  - Tighten formulation and reduce problem size
- Solve continuous relaxations
  - Ignoring integrality
  - Gives a bound on the optimal integral objective
- Cutting planes
  - Cut off relaxation solutions
- Branching variable selection
  - Crucial for limiting search tree size
- Primal heuristics
  - Find integer feasible solutions

Primal Heuristics Explained on Twitter



Matteo Fischetti @MFischetti · 1 Std.

.@AchterbergT hmmm, challenging suggestion: write a full scientific paper in a tweet! I would be tempted...

Matteo Fischetti @MFischetti · 1 Std.

.@AchterbergT e.g. #LocalBranching take a 0-1 MIP and a solution x*, bound Hamming distance from x* through a linear cut, and solve again.

1

Matteo Fischetti @MFischetti · 1 Std.

.@AchterbergT #RINS take a MIP, a feasible sol. x~ and the LP relaxation sol. x*, fix all components that agree, and solve as a MIP.

1

Matteo Fischetti @MFischetti · 1 Std.

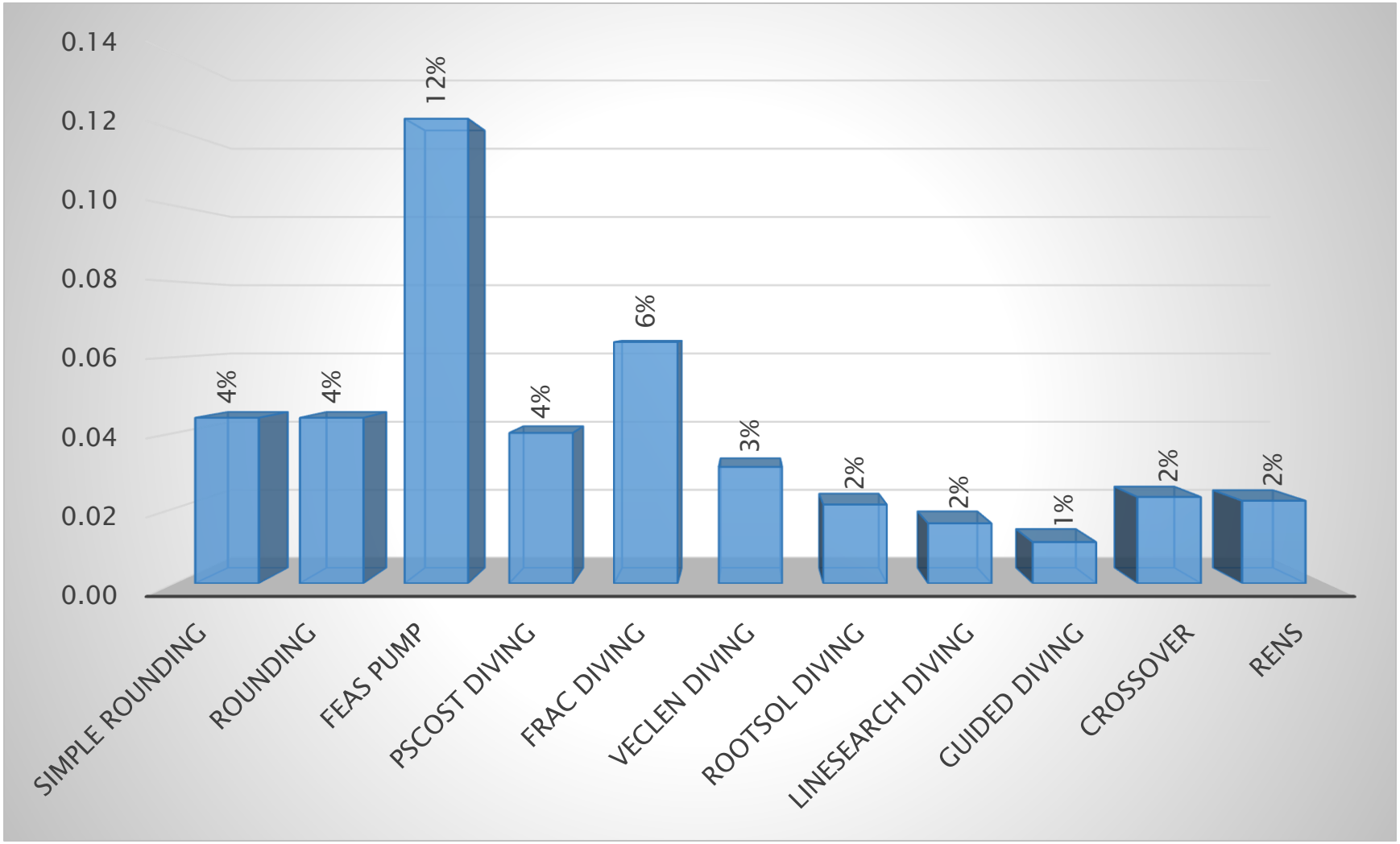
.@AchterbergT #FP Take MIP and LP sol. x*, round it, solve LP again but minimizing Hamming distance from rounded sol. Repeat. Serve it warm.

2

Primal Heuristics

- Try to find good integer feasible solutions quickly
  - Better pruning during search due to better bound
  - Reach desired gap faster
  - Often important in practice: quality of solution after fixed amount of time
- Start heuristics
  - Try to find integer feasible solution, usually "close" to LP solution
    - Rounding heuristics: round LP solution to integral values
      - Potentially, try to fix constraint infeasibilities
    - Fix-and-dive heuristics: fix variables, propagate, resolve LP
    - Feasibility pump: push LP solution towards integrality by modifying objective
    - RENS: Solve sub-MIP in neighborhood of LP solution
- Improvement heuristics
  - Given integer feasible solution, try to find better solution
    - 1-Opt and 2-Opt: Modify one or two variables to get better objective
    - Local Branching: Solve sub-MIP in neighborhood of MIP solution
    - Mutation: Solve sub-MIP in neighborhood of MIP solution
    - Crossover: Solve sub-MIP in neighborhood of 2 or more MIP solutions
    - RINS: Solve sub-MIP in neighborhood of LP and MIP solution

Primal Heuristics – Performance

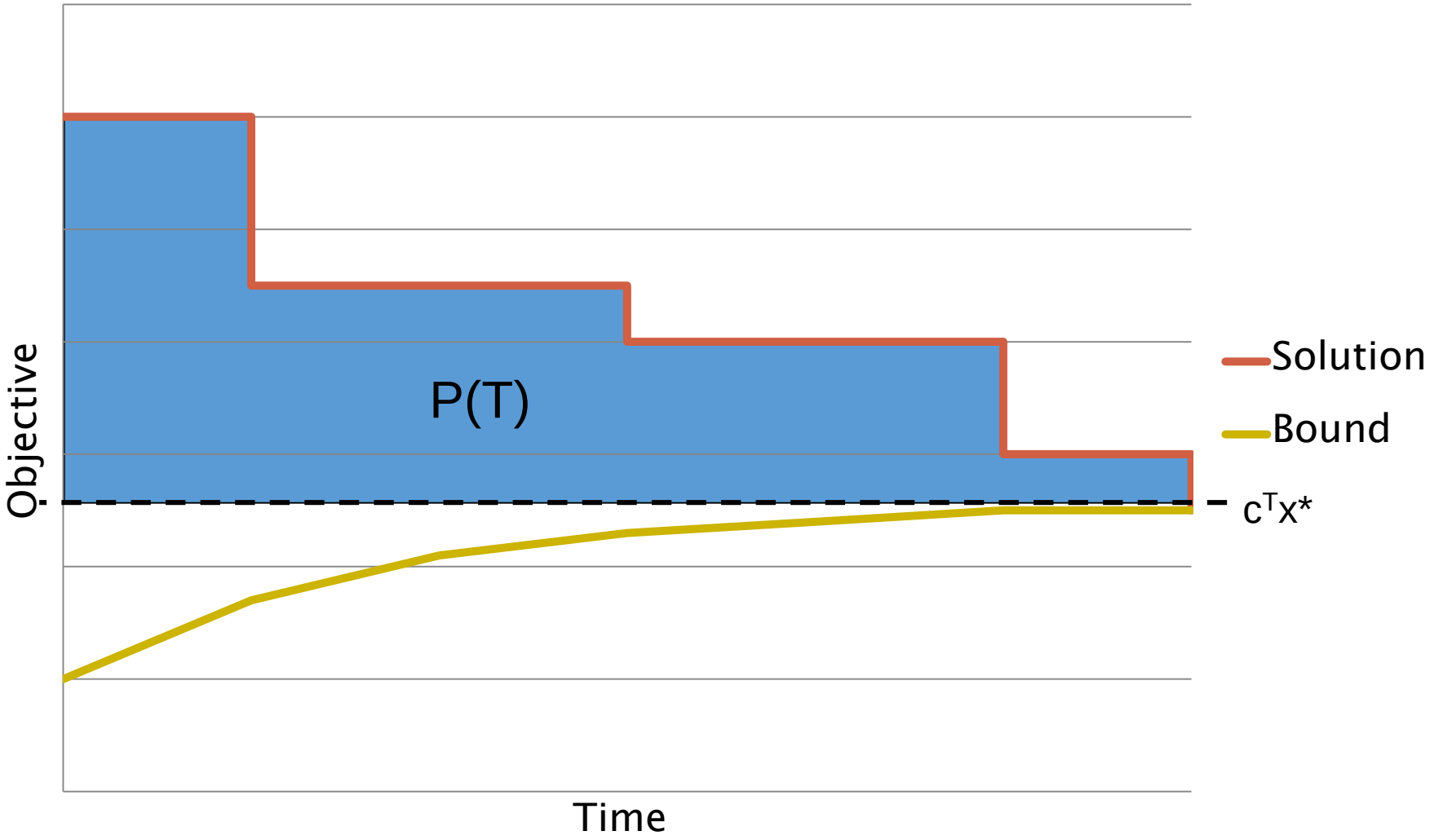


Berthold: "Primal Heuristics for Mixed Integer Programs" (2006)
benchmark data based on SCIP 0.82b

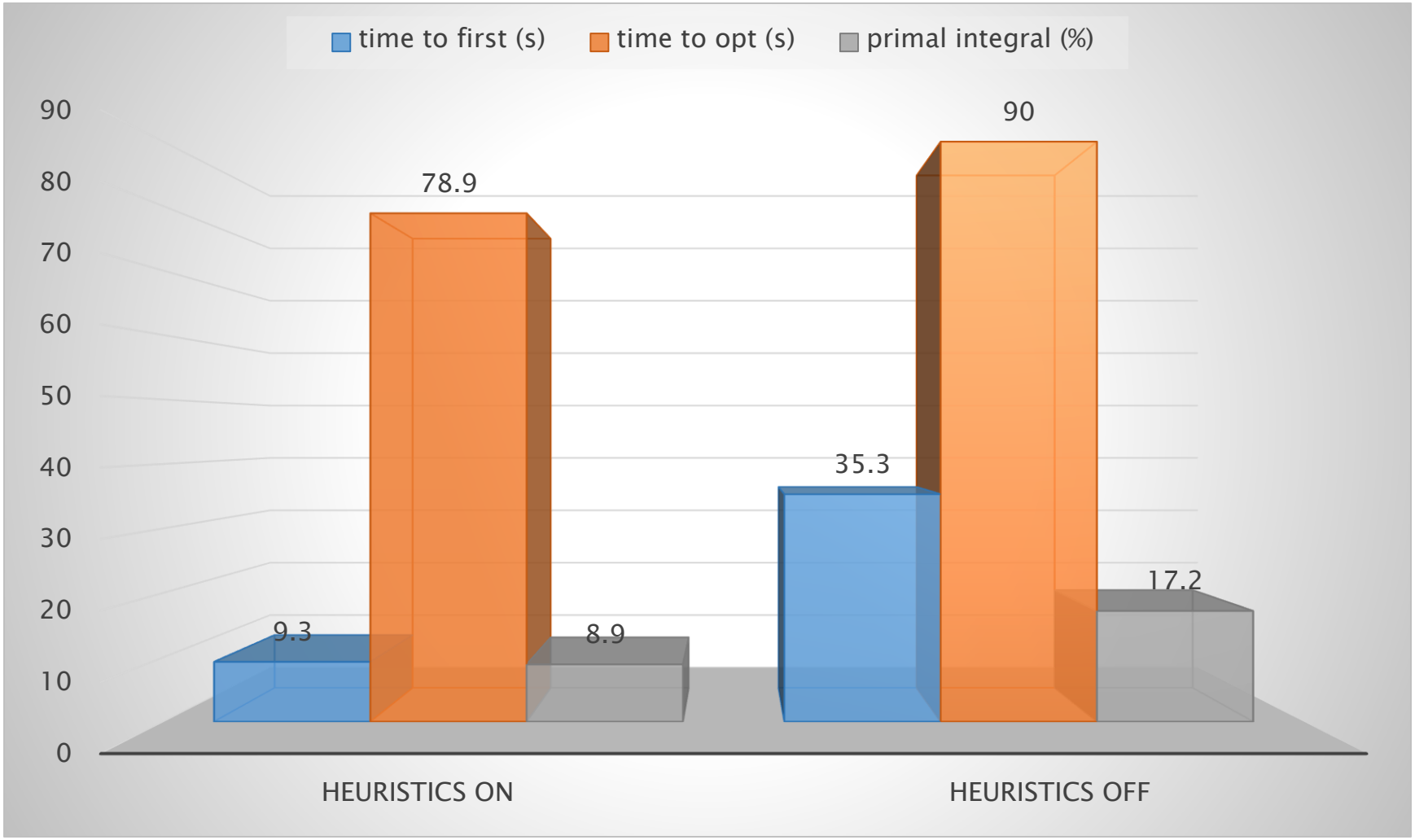
Primal Heuristics – Measuring Performance

- Is time to optimality a good measure to assess impact of heuristics?
  - Goal of heuristics is to provide good solutions quickly
  - Faster progress in dual bound due to additional pruning is only secondary
  - Often important for practitioners:
    - Find any feasible solution quickly to validate that model is reasonable
    - Find good solution in reasonable time frame
- Primal gap: $\gamma^p(\tilde{x}) = \frac{|c^T x^* - c^T \tilde{x}|}{max\{|c^T x^*|, |c^T \tilde{x}|\}}$
- Primal gap function: $p(t) = \begin{cases} 1, & \text{if no incumbent until time } t \\ \gamma^p(\tilde{x}(t)), & \text{with } \tilde{x}(t) \text{ being incumbent at time } t \end{cases}$
- Primal integral: $P(T) = \int_{t=0}^{T} p(t)dt$

Primal Integral



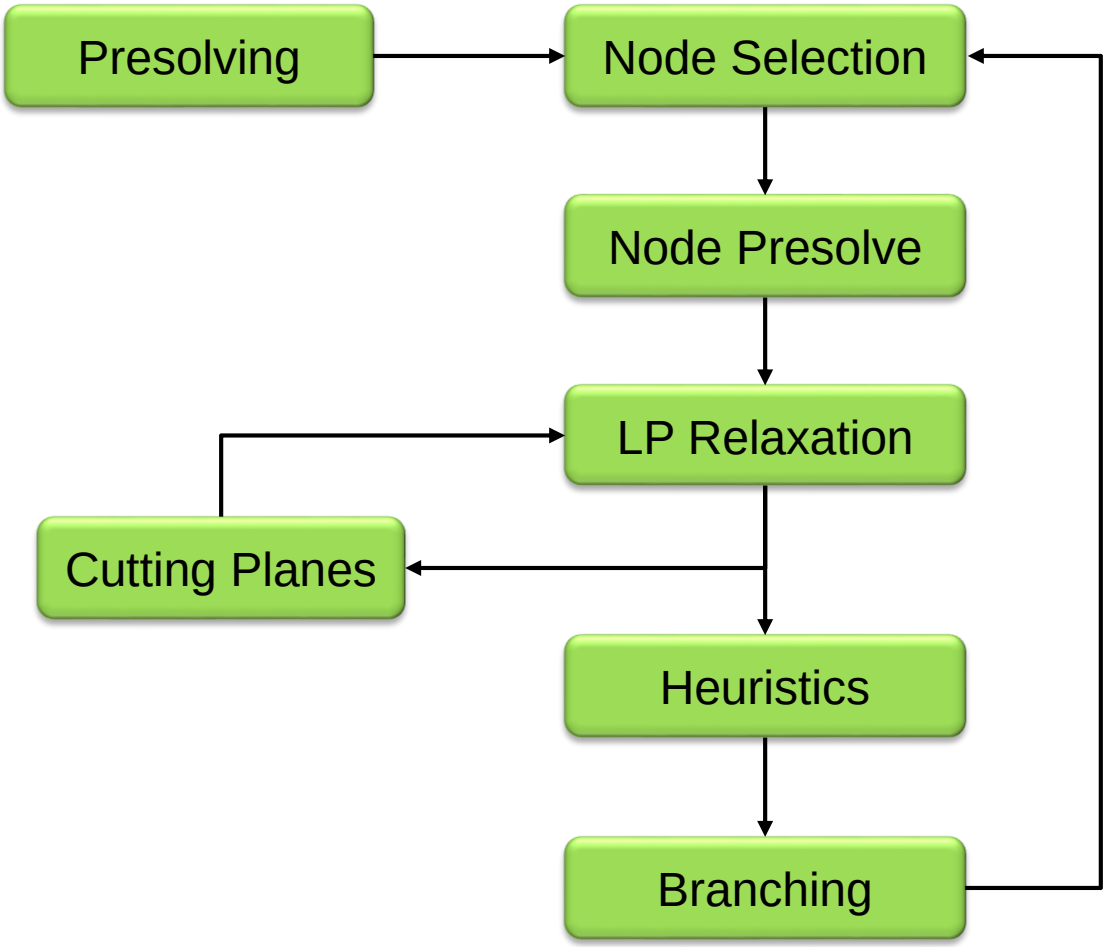
Primal Heuristics – Performance



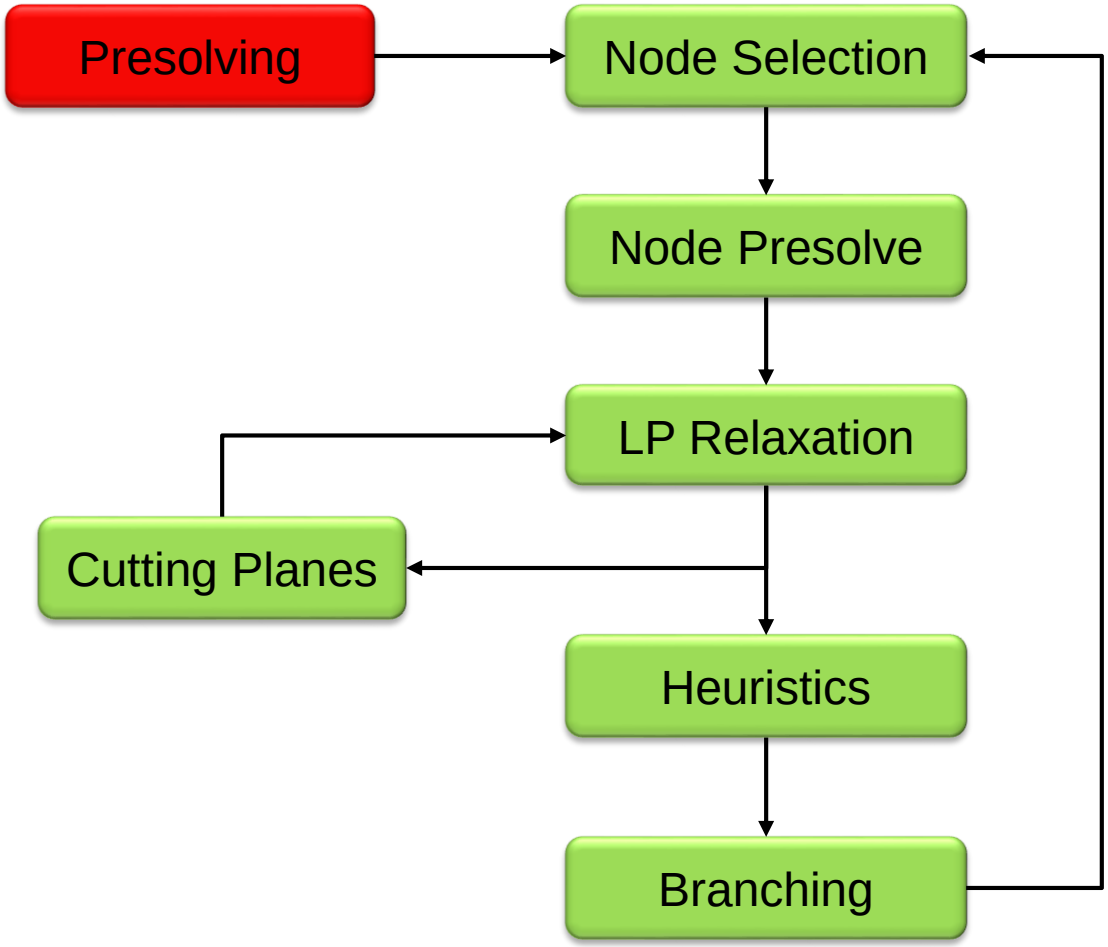
Berthold (2014): "Heuristic algorithms in global MINLP solvers"
benchmark data based on SCIP 3.0.2

Putting It All Together

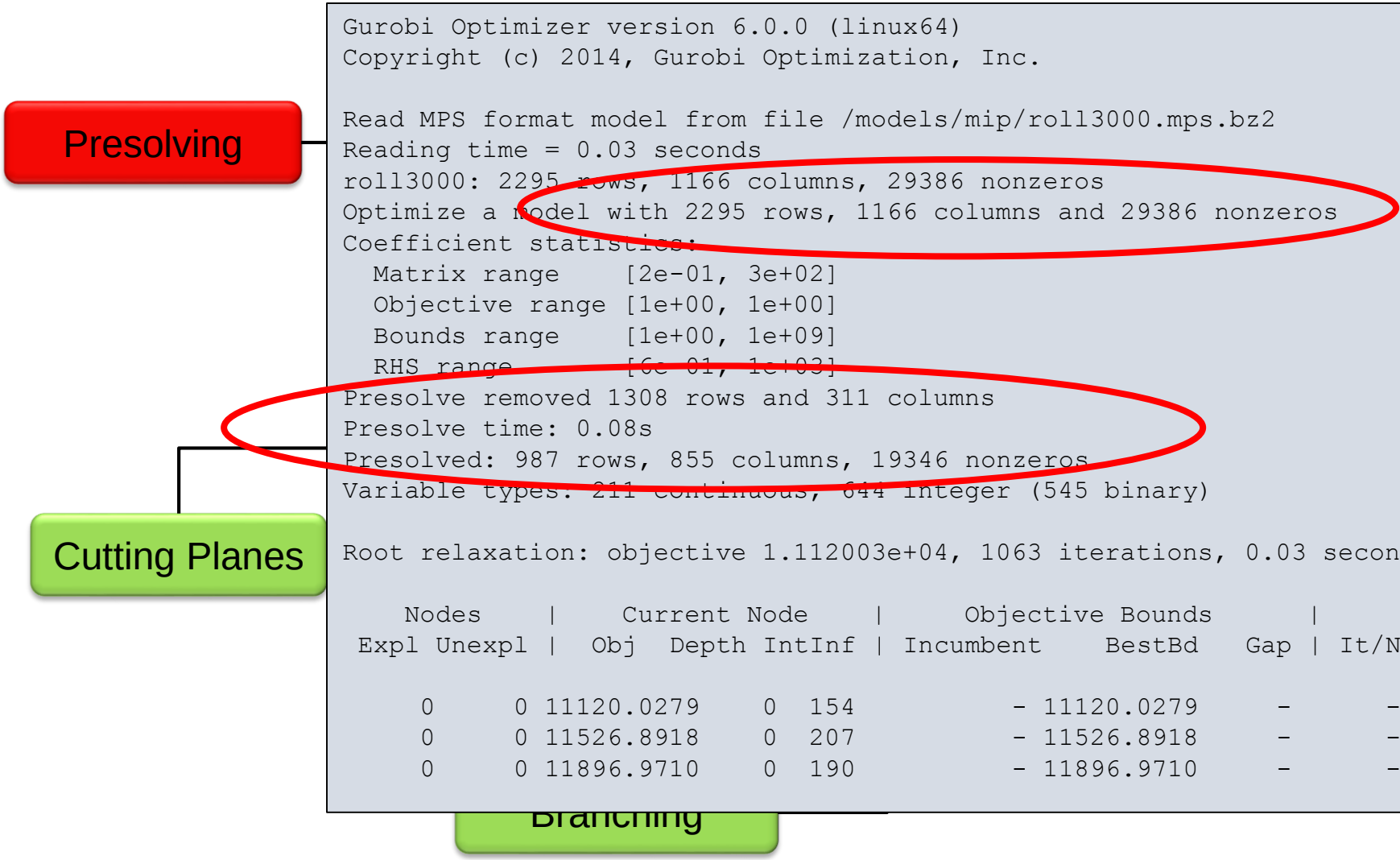
Branch-and-Cut



Branch-and-Cut

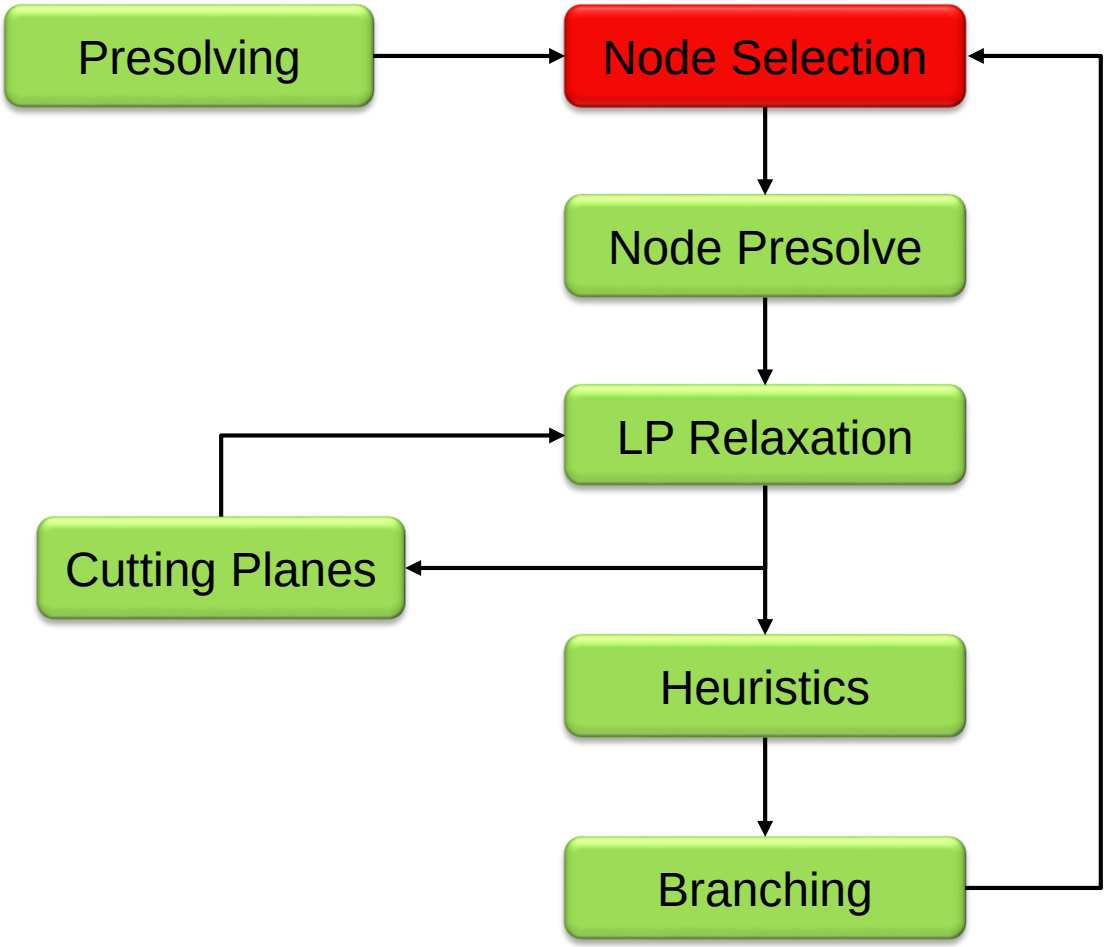


Branch-and-Cut

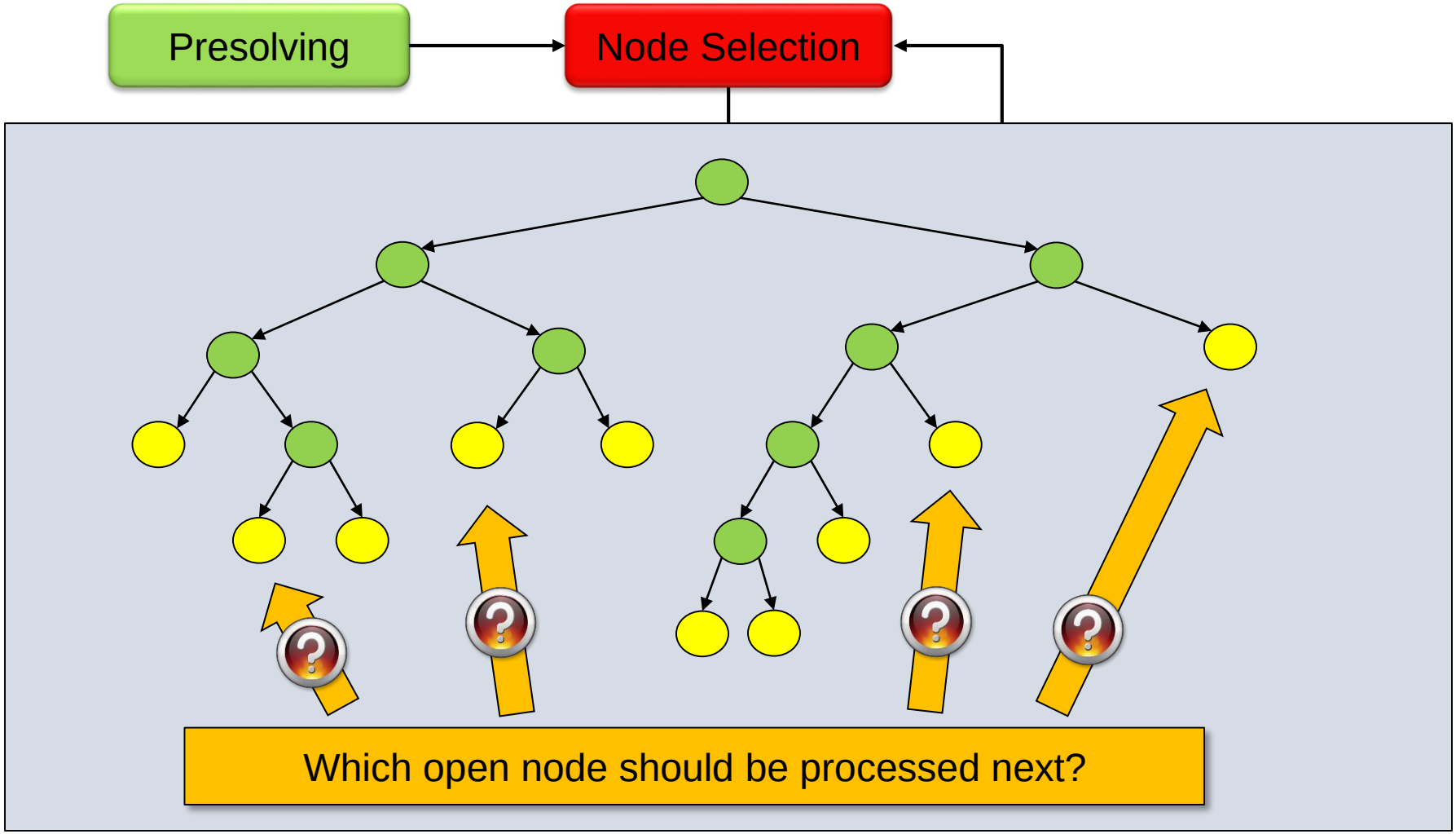



```
Gurobi Optimizer version 6.0.0 (linux64)
Copyright (c) 2014, Gurobi Optimization, Inc.

Read MPS format model from file /models/mip/roll3000.mps.bz2
Reading time = 0.03 seconds
roll3000: 2295 rows, 1166 columns, 29386 nonzeros
Optimize a model with 2295 rows, 1166 columns and 29386 nonzeros
Coefficient statistics:
  Matrix range     [2e-01, 3e+02]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 1e+09]
  RHS range        [6e-01, 1e+03]
Presolve removed 1308 rows and 311 columns
Presolve time: 0.08s
Presolved: 987 rows, 855 columns, 19346 nonzeros
Variable types: 211 continuous, 644 integer (545 binary)

Root relaxation: objective 1.112003e+04, 1063 iterations, 0.03 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Wo
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Noo

     0     0 11120.0279    0  154          - 11120.0279      -      -
     0     0 11526.8918    0  207          - 11526.8918      -      -
     0     0 11896.9710    0  190          - 11896.9710      -      -
```

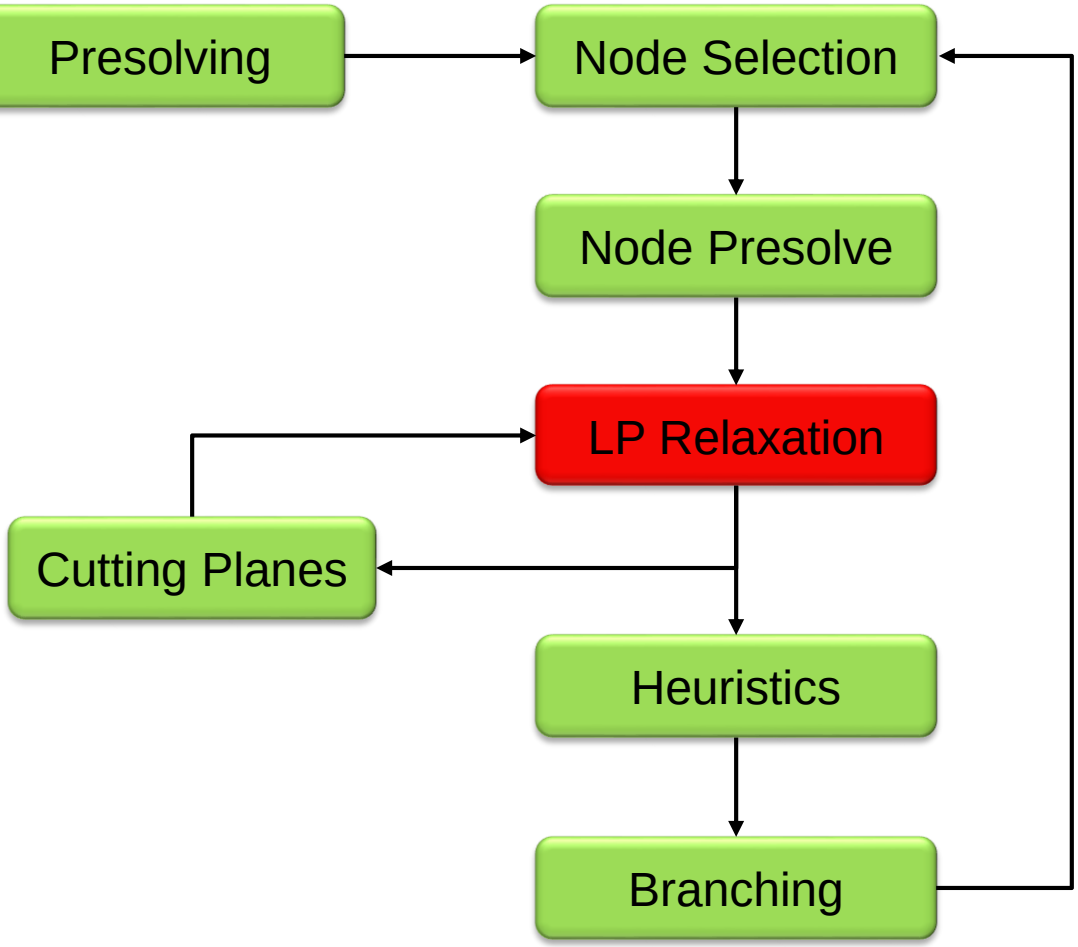
Branch-and-Cut



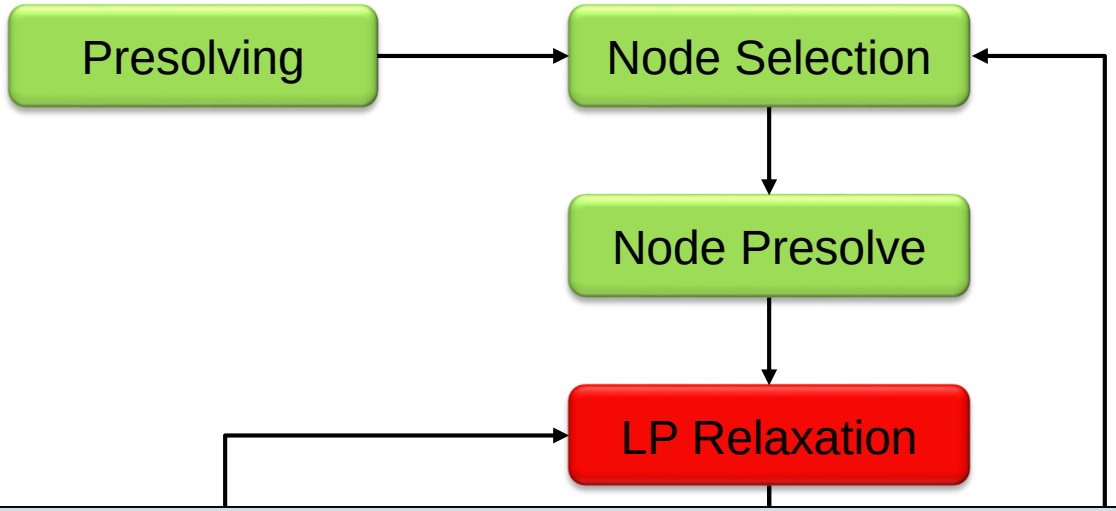
Branch-and-Cut



Branch-and-Cut

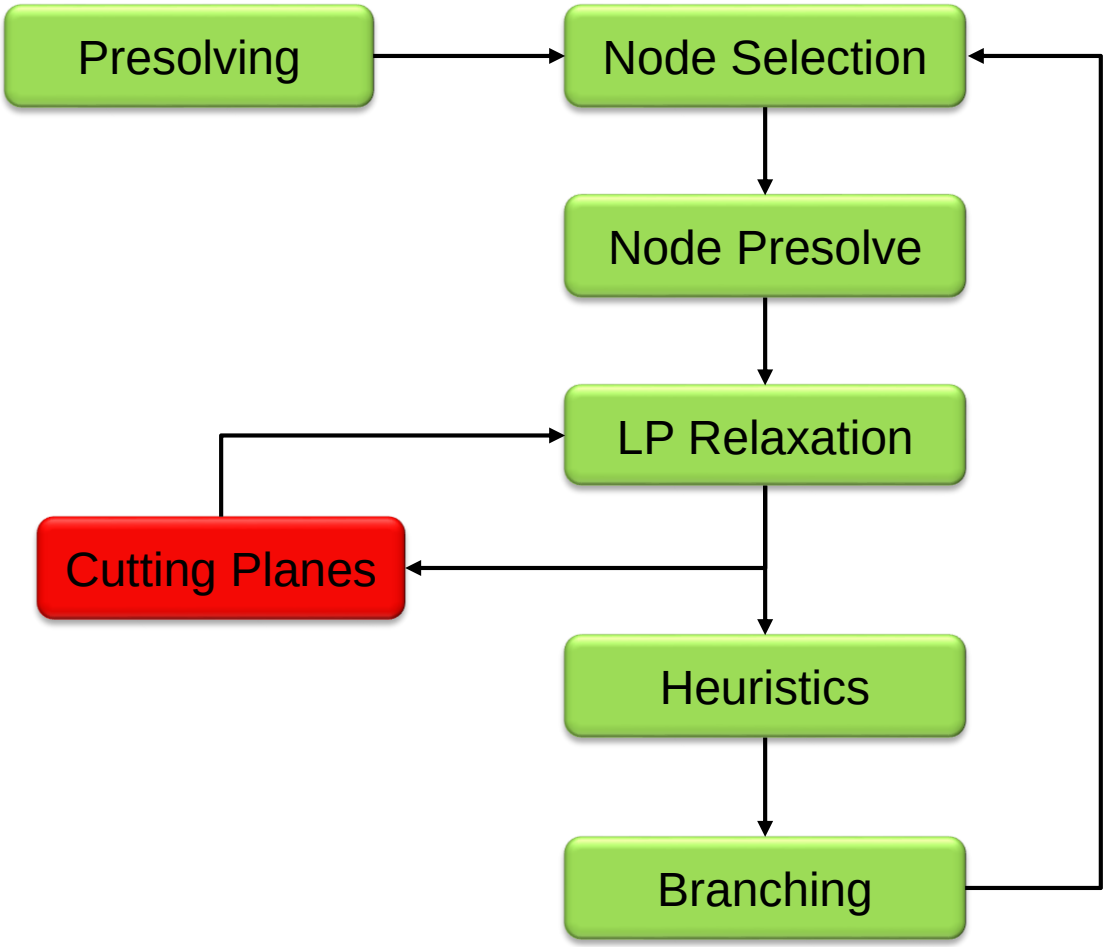


Branch-and-Cut

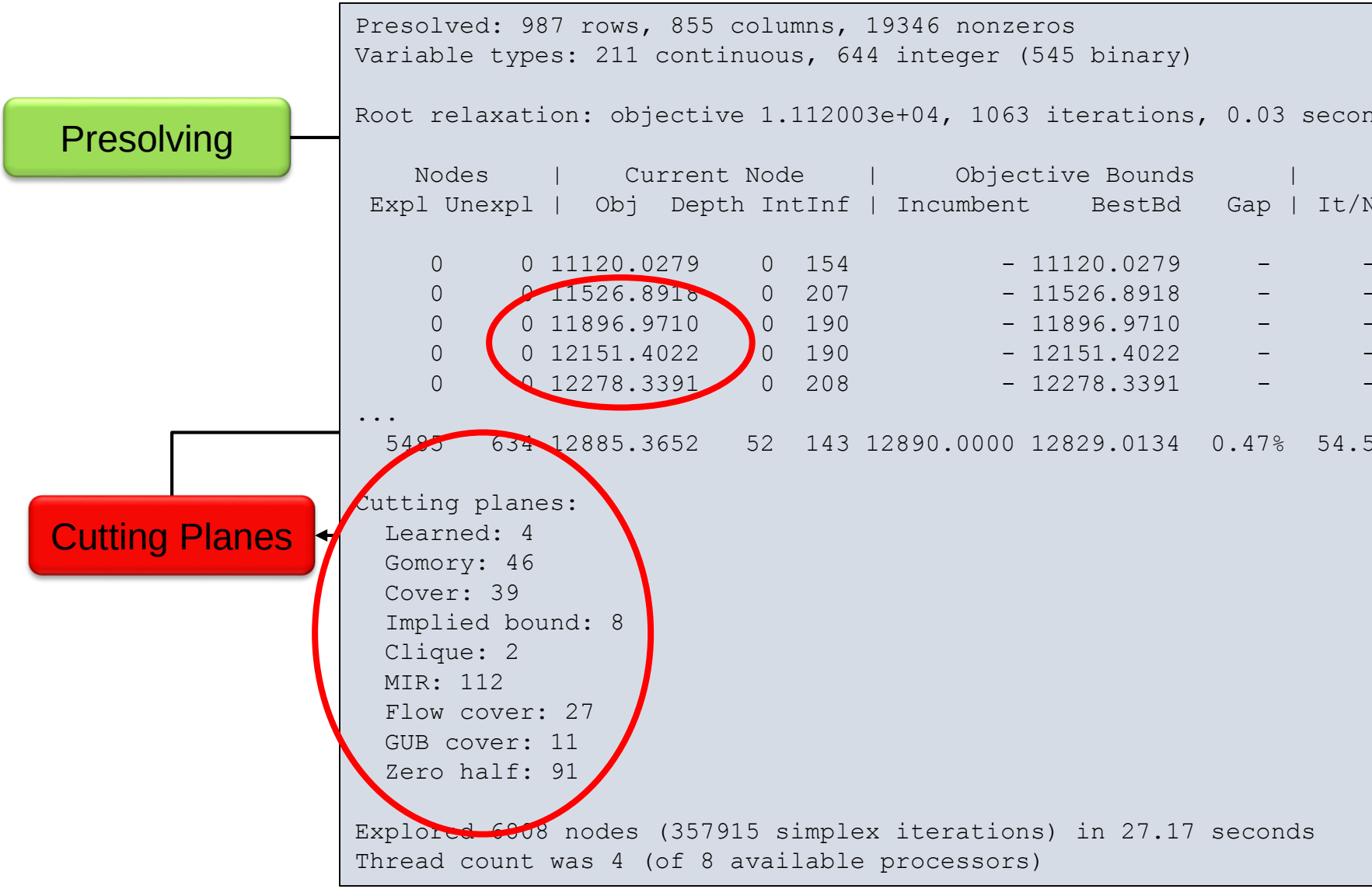



```
Presolved: 987 rows, 855 columns, 19346 nonzeros
Variable types: 211 continuous, 644 integer (545 binary)

Root relaxation: objective 1.112003e+04, 1063 iterations, 0.03 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 11120.0279    0  154          - 11120.0279      -      -    0s
     0     0 11526.8918    0  207          - 11526.8918      -      -    0s
     0     0 11896.9710    0  190          - 11896.9710      -      -    0s
...
H  327   218                    13135.000000 12455.2162  5.18%   42.6    1s
H  380   264                    13093.000000 12455.2162  4.87%   41.6    1s
H  413   286                    13087.000000 12455.2162  4.83%   41.4    1s
  1066   702 12956.2676   31  192 13087.0000 12629.5426  3.50%   37.6    5s
```

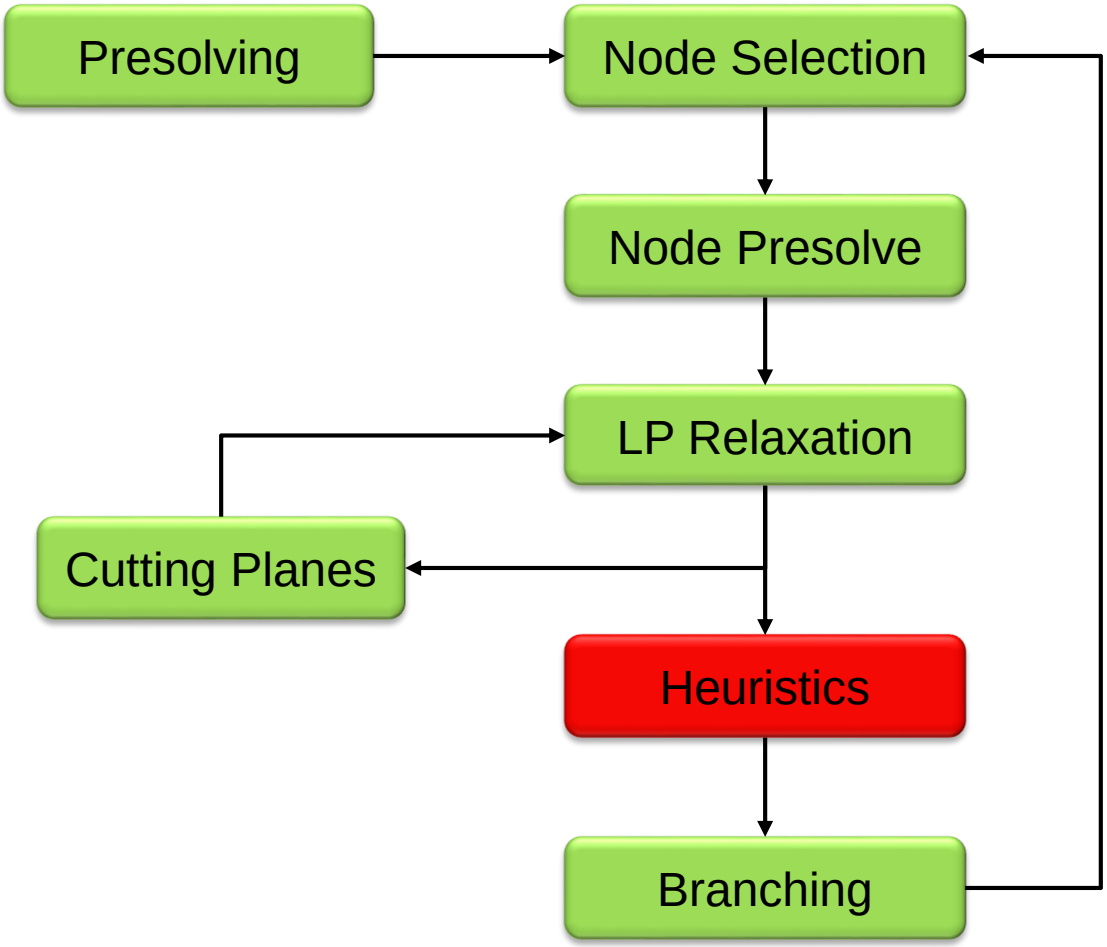
Branch-and-Cut



Branch-and-Cut



Branch-and-Cut



Branch-a

```
Presolved: 987 rows, 855 columns, 19346 nonzeros
Variable types: 211 continuous, 644 integer (545 binary)

Root relaxation: objective 1.112003e+04, 1063 iterations, 0.03 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Tim

     0     0 11120.0279    0  154          - 11120.0279      -       -    0s
     0     0 11526.8918    0  207          - 11526.8918      -       -    0s
     0     0 11896.9710    0  190          - 11896.9710      -       -    0s
...
     0     0 12448.7684    0  181          - 12448.7684      -       -    0s
H    0     0                       16129.000000 12448.7684  22.8%       -    0s
H    0     0                       15890.000000 12448.7684  21.7%       -    0s
     0     2 12448.7684    0  181 15890.0000 12448.7684  21.7%       -    0s
H  142   129                       15738.000000 12450.7195  20.9%   43.8    1s
H  212   189                       14596.000000 12453.8870  14.7%   42.3    1s
H  217   181                       13354.000000 12453.8870  6.74%   42.6    1s
*  234   181              40       13319.000000 12453.8870  6.50%   42.1    1s
H  254   190                       13307.000000 12453.8870  6.41%   41.3    1s
H  284   194                       13183.000000 12453.8870  5.53%   42.6    1s
H  286   194                       13169.000000 12453.8870  5.43%   42.7    1s
```

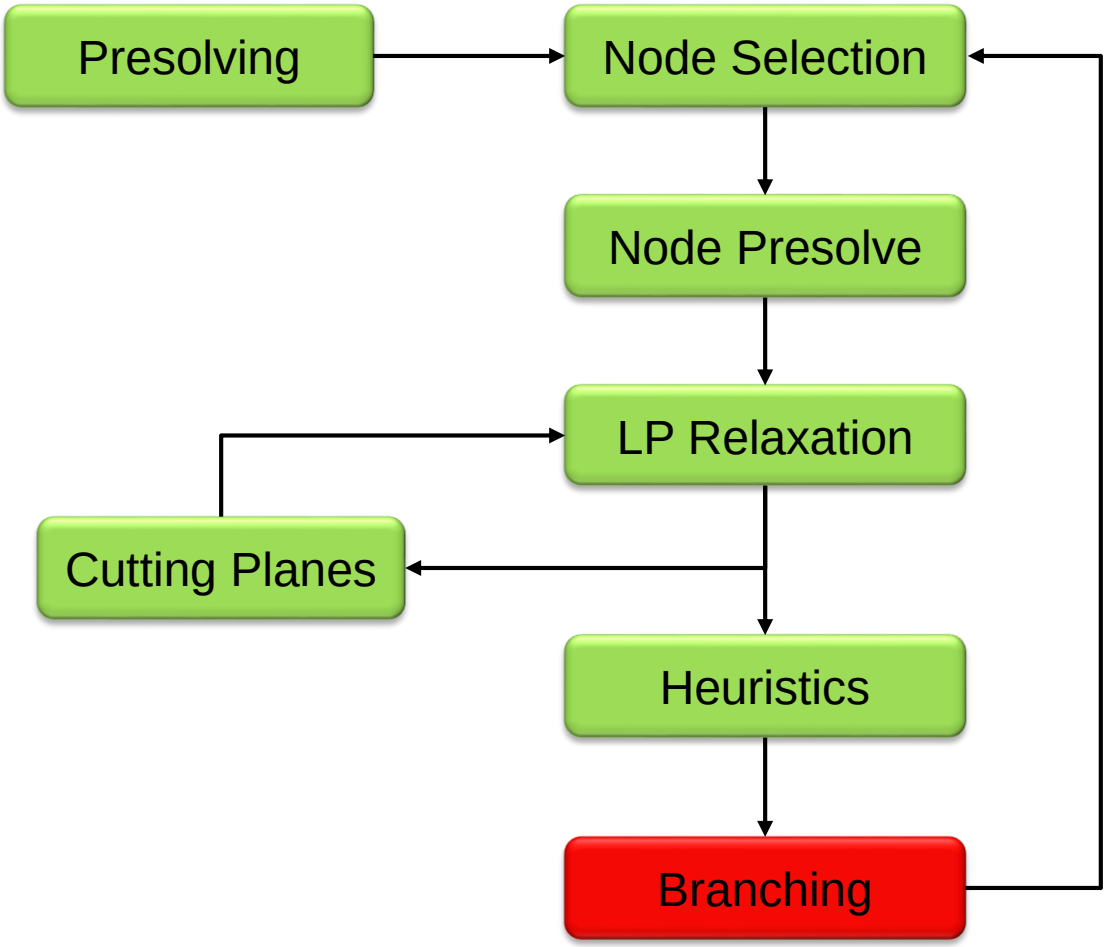
Presolving

Cutting Pla

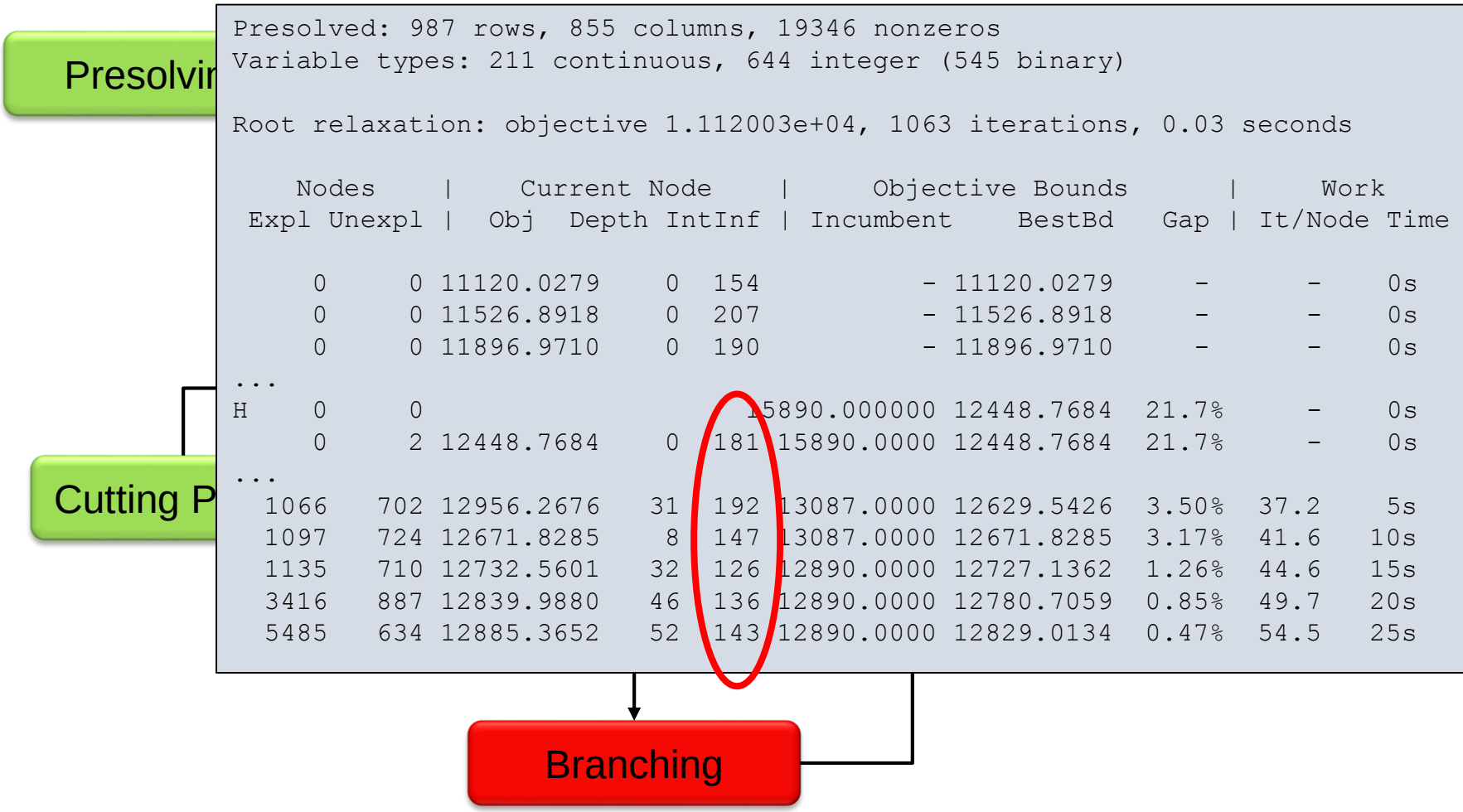
Heuristics

Branching

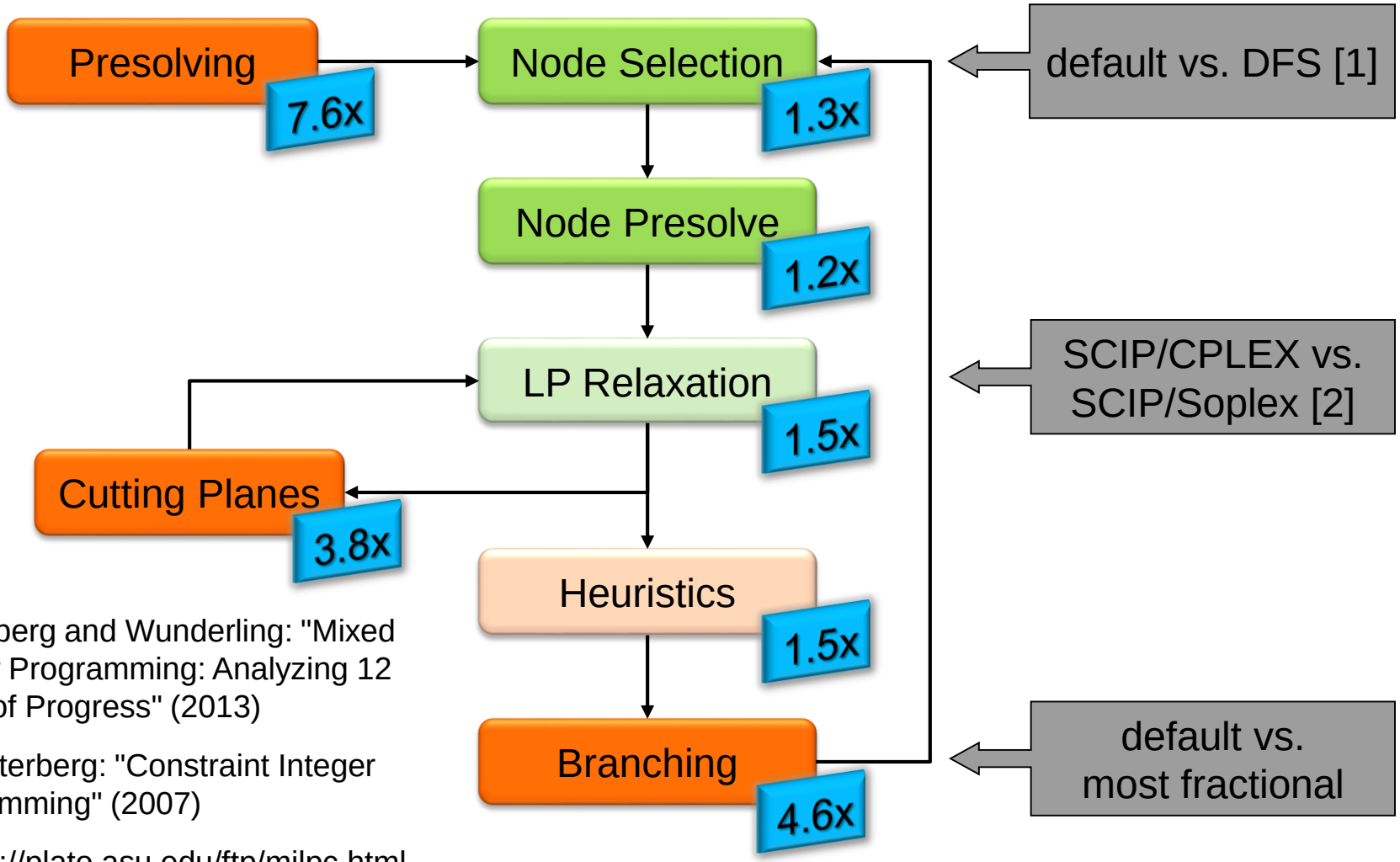
Branch-and-Cut



Branch-and-Cut



Performance Impact of MIP Solver Components (CPLEX 12.5 or SCIP)



Achterberg and Wunderling: "Mixed Integer Programming: Analyzing 12 Years of Progress" (2013)

[1] Achterberg: "Constraint Integer Programming" (2007)

[2] http://plato.asu.edu/ftp/milpc.html

Parallelization

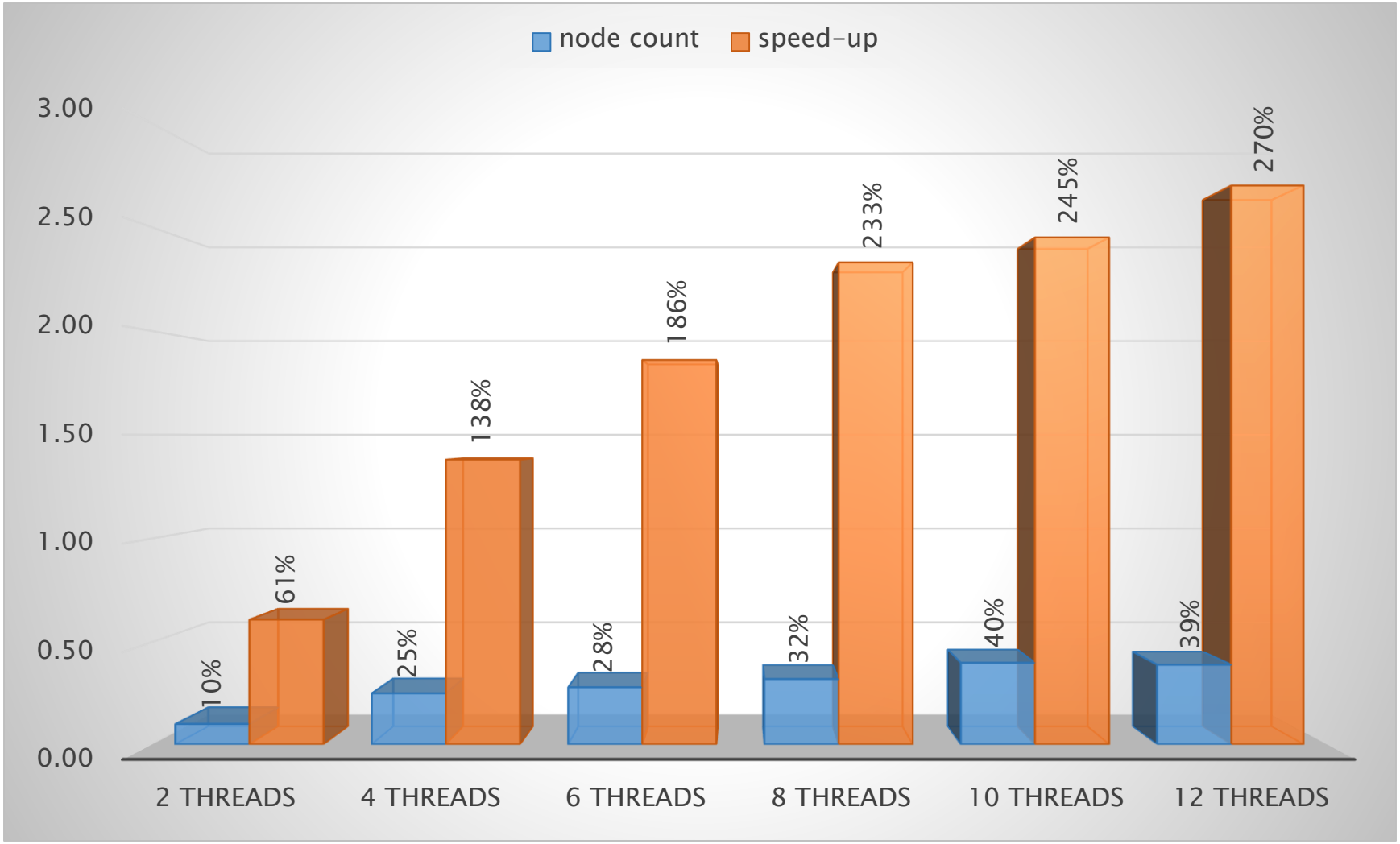
- Parallelization opportunities
  - Parallel probing during presolve
    - Almost no improvement
  - Use barrier or concurrent LP for initial LP relaxation solve
    - Only helps for large models
  - Run heuristics or other potentially useful algorithms in parallel to the root cutting plane loop
    - Moderate performance improvements: 20-25%
    - Does not scale beyond a few threads
  - Solve branch-and-bound nodes in parallel
    - Main speed-up for parallel MIP
    - Performance improvement depends a lot on shape of search tree
    - Typically scales relatively well up to 8 to 16 threads

Parallelization

- Parallelization issues
  - Determinism
  - Load balancing
  - CPU heat and memory bandwidth
    - Additional threads slow down main thread
  - Root node does not parallelize well
    - Sequential runtime of root node imposes limits on parallelization speed-up
    - Amdahl's law
  - A dive in the search tree cannot be parallelized
    - Parallelization only helps if significant number of dives necessary to solve model

- More on parallel MIP: tomorrow

Parallel MIP – Performance



Achterberg and Wunderling: "Mixed Integer Programming: Analyzing 12 Years of Progress" (2013)
benchmark data based on CPLEX 12.5, models with ≥ 100 seconds solve time