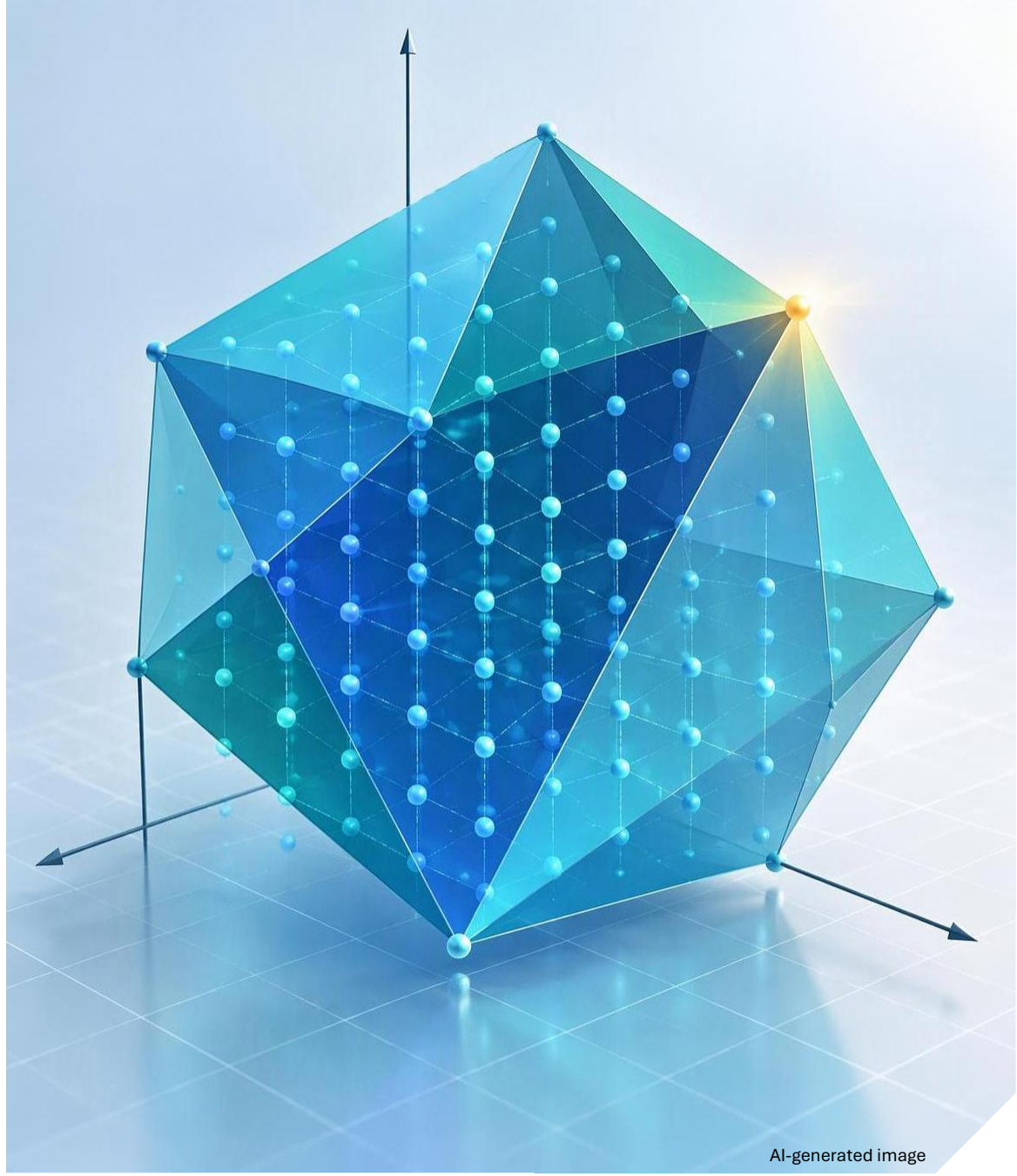
Gurobi®

Conflict Analysis and Restarts in MIP

Scheduling Seminar – schedulingseminar.com
18 Nov. 2026



© Gurobi Optimization, LLC. All rights reserved.

Mixed Integer Program

Mathematical Definition

- Minimize a linear objective function
- Such that a number of linear constraints are satisfied
- And a given subset of the variables takes integer values

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b \\ & x_j \in \mathbb{Z} \quad \text{for all } j \in I \end{aligned}$$

- Why should we care about mixed integer programming (MIP)?
  - 70% of Fortune 500 companies are Gurobi customers
  - The vast majority of them solve MIPs
    - Gurobi also solves LPs, QPs, NLPs, MIQPs, MIQCPs, and MINLPs

Gurobi

© Gurobi Optimization, LLC. All rights reserved.

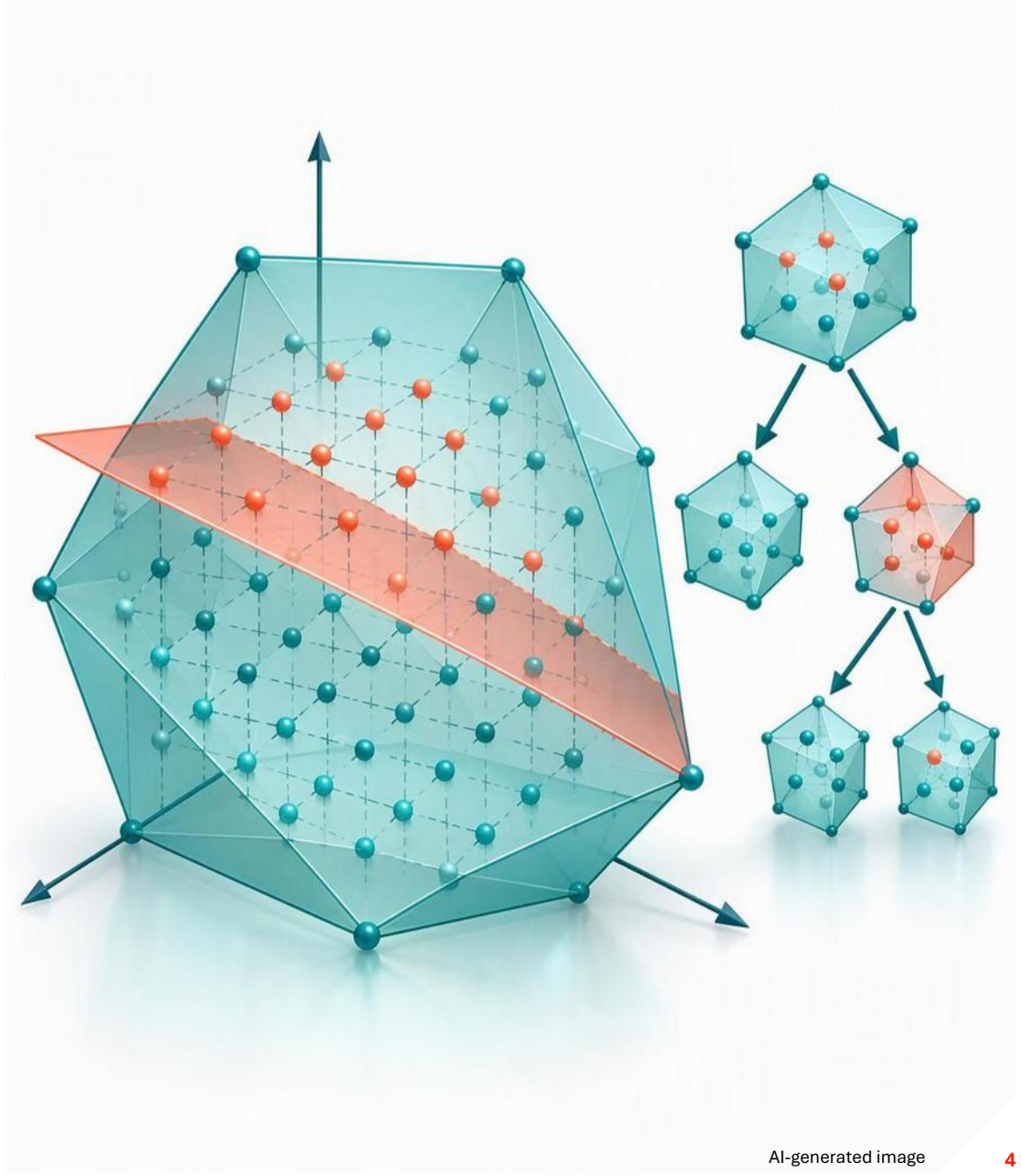
Applications of Mixed Integer Programming

- 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
- ...

© Gurobi Optimization, LLC. All rights reserved.

MIP Algorithms

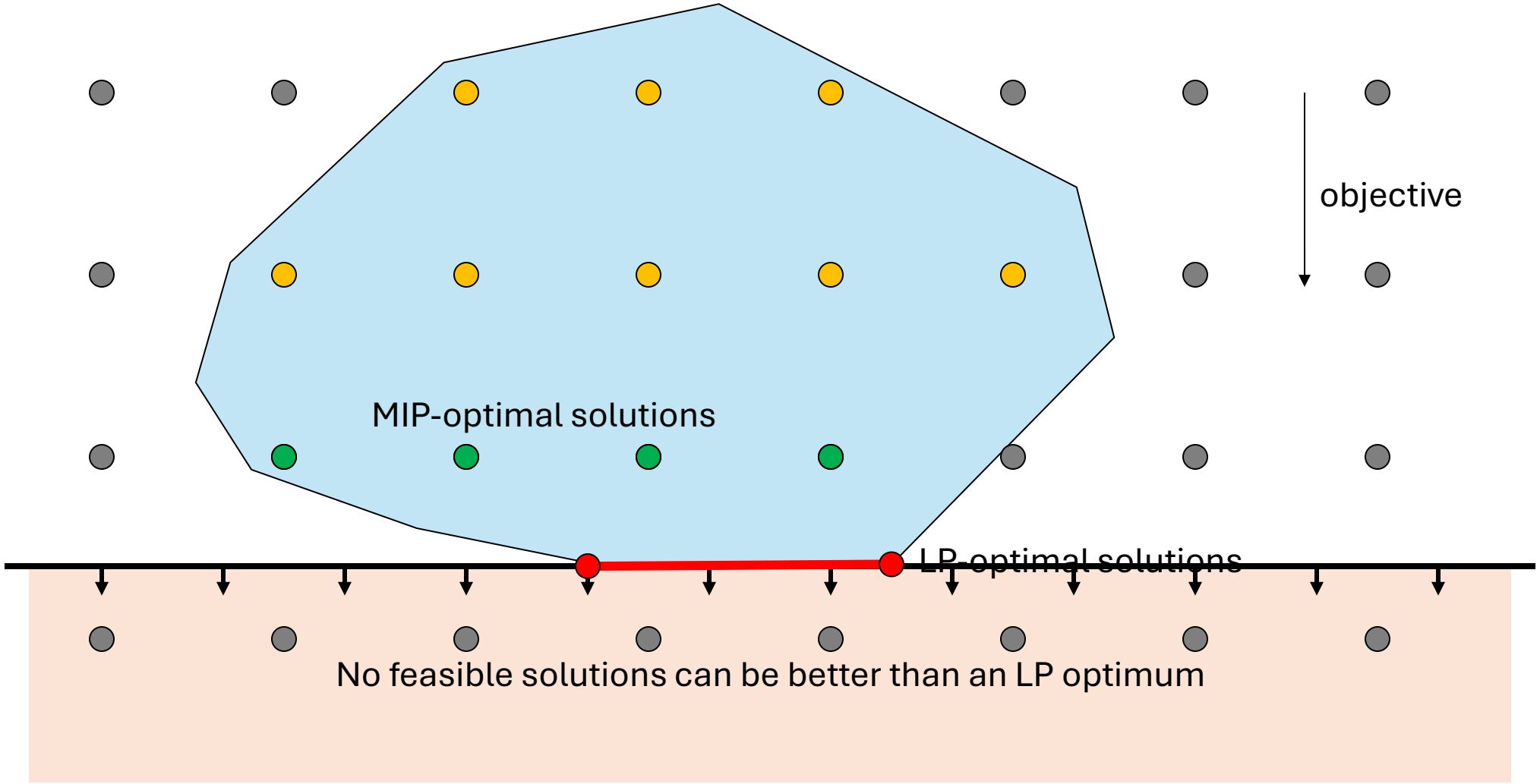
How to Solve Mixed Integer Programs in Practice



AI-generated image

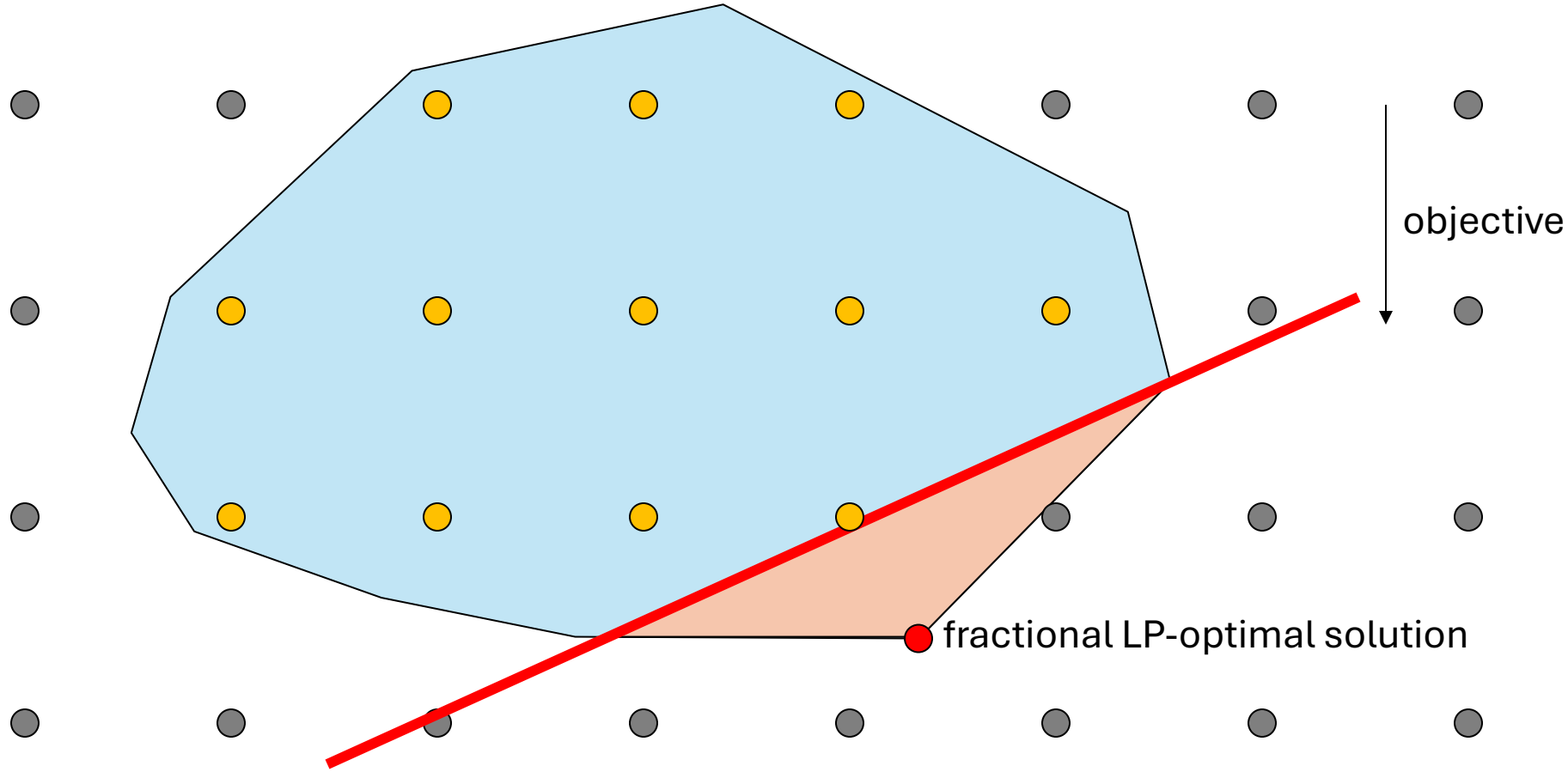
© Gurobi Optimization, LLC. All rights reserved.

MIP – LP Relaxation



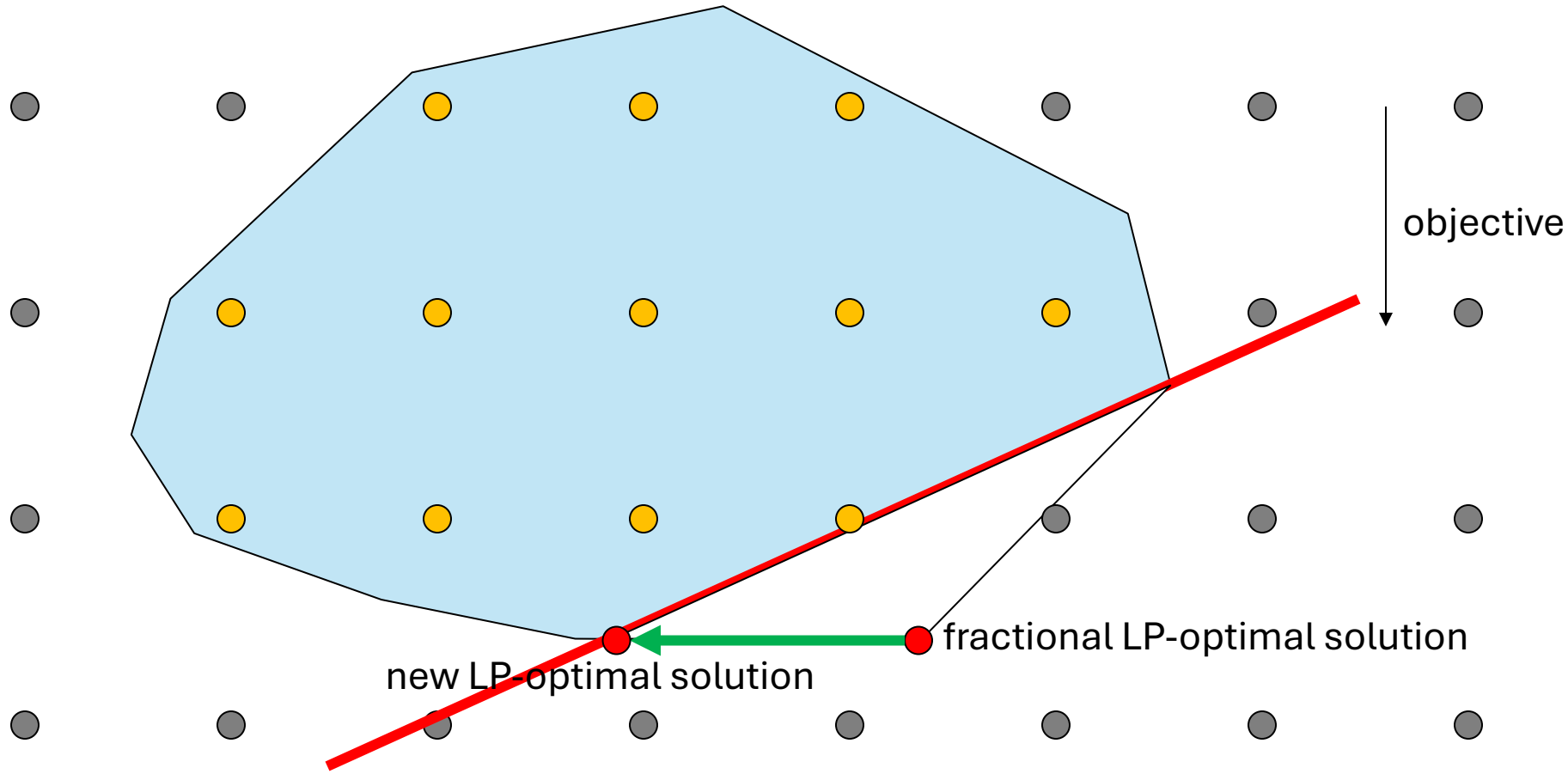
© Gurobi Optimization, LLC. All rights reserved.

MIP – Cutting Planes



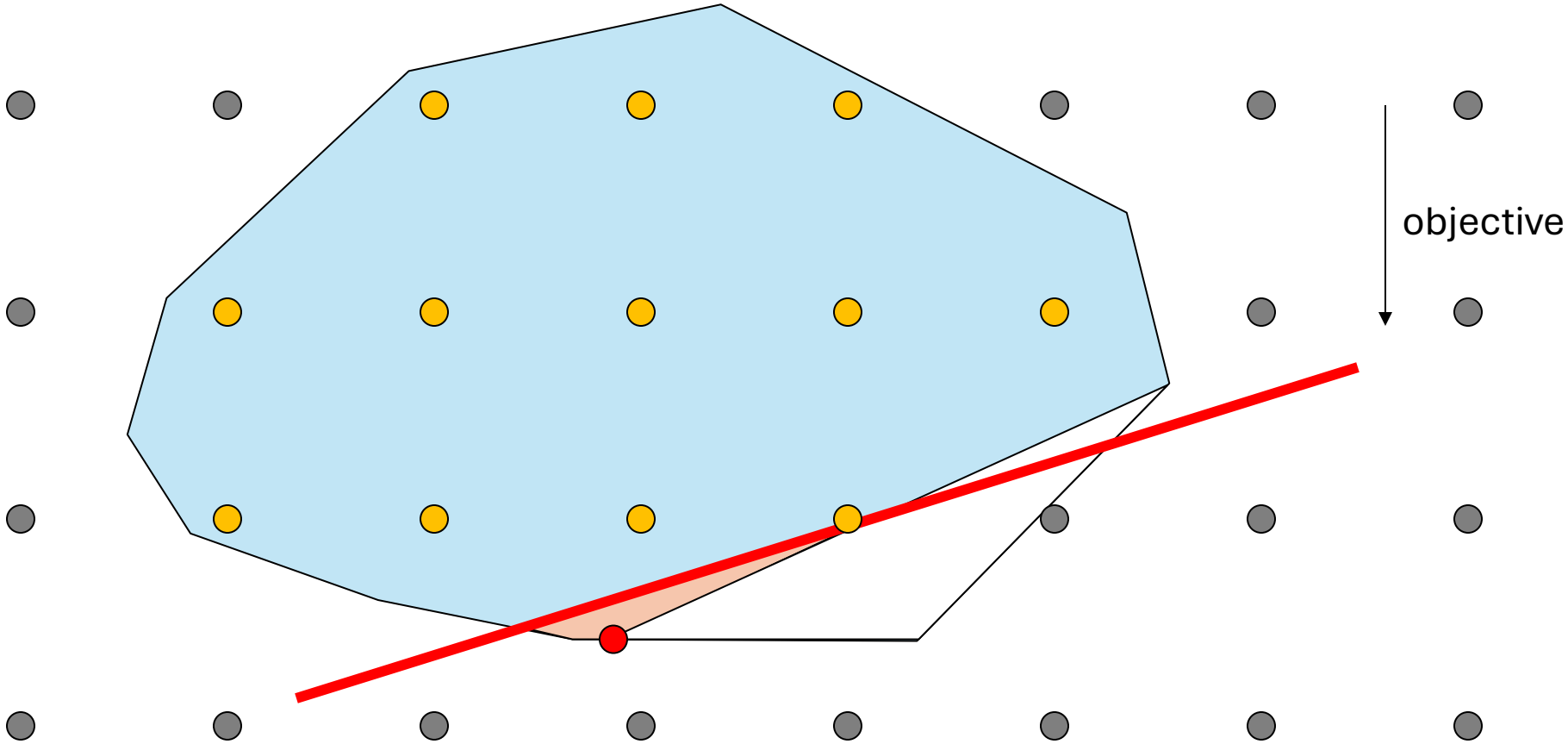
© Gurobi Optimization, LLC. All rights reserved.

MIP – Cutting Planes



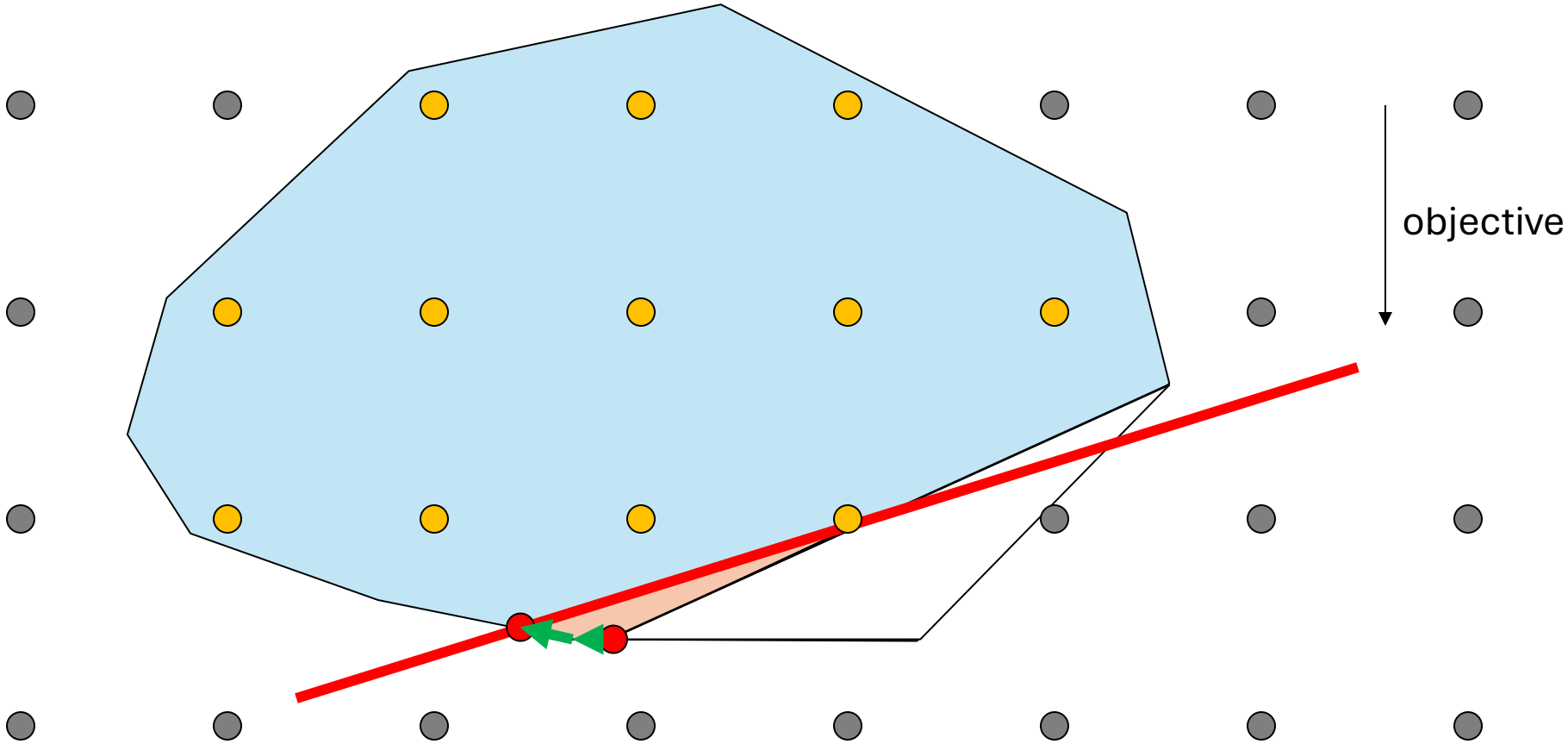
© Gurobi Optimization, LLC. All rights reserved.

MIP – Cutting Planes



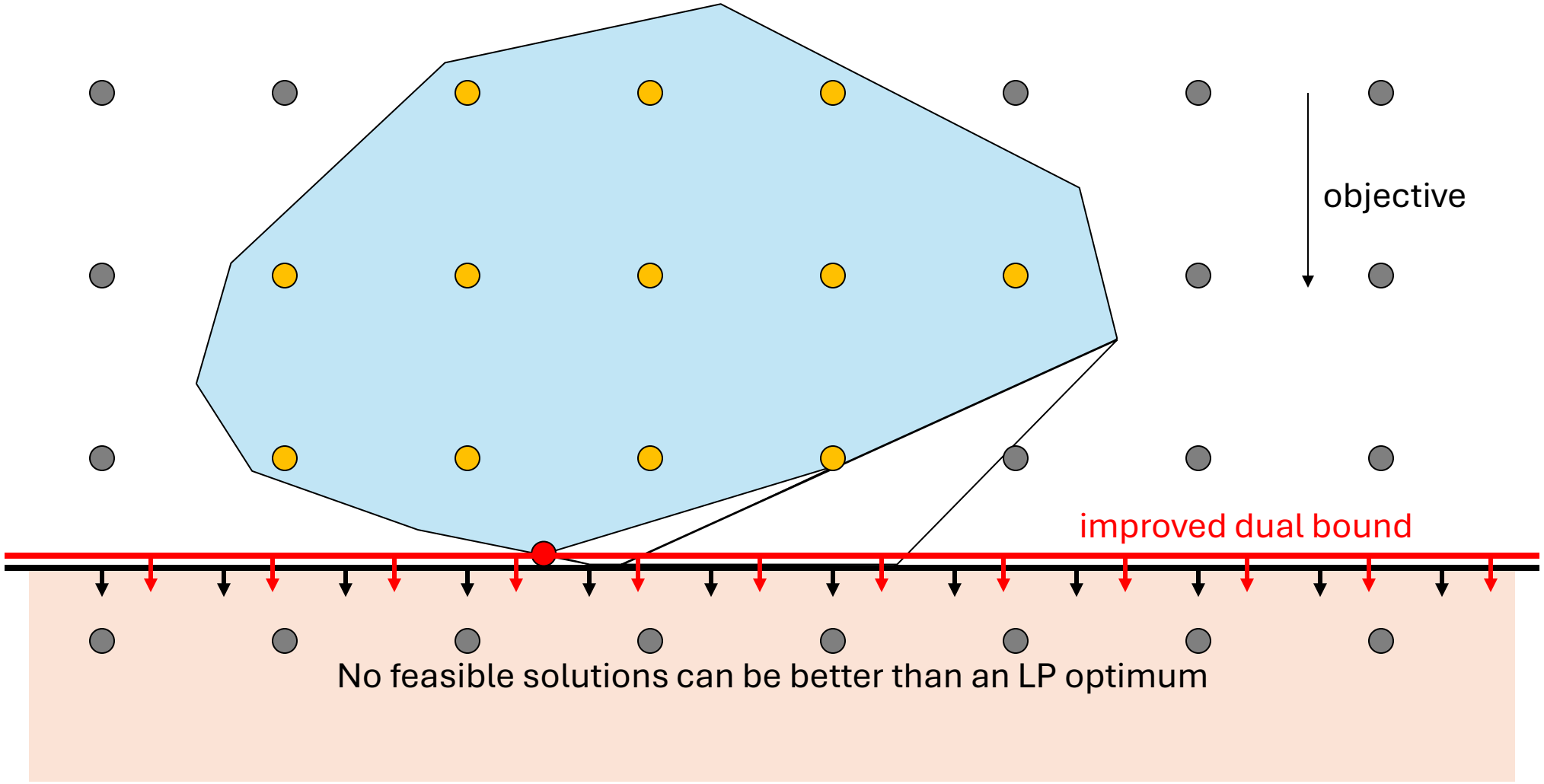
© Gurobi Optimization, LLC. All rights reserved.

MIP – Cutting Planes



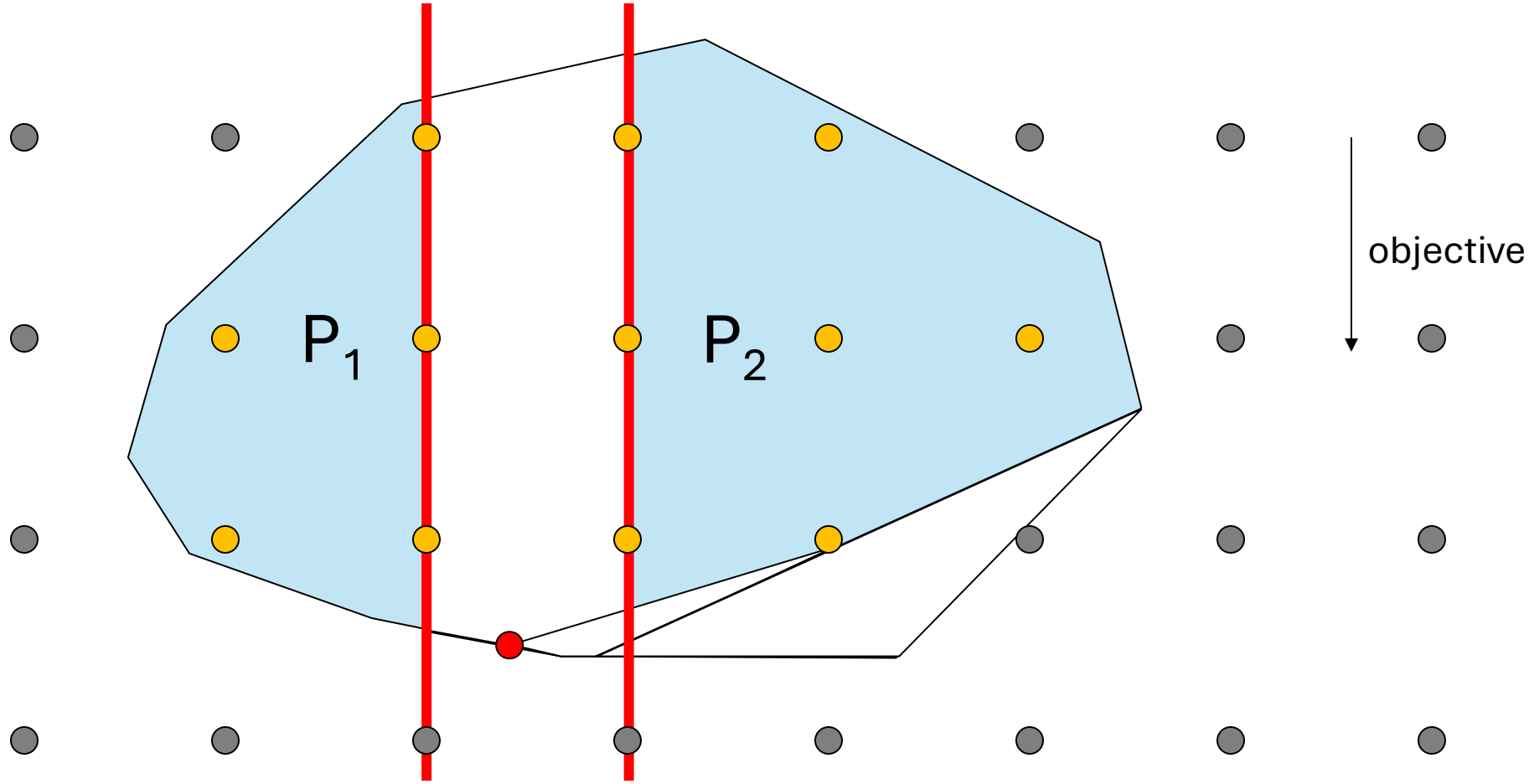
© Gurobi Optimization, LLC. All rights reserved.

MIP – Cutting Planes



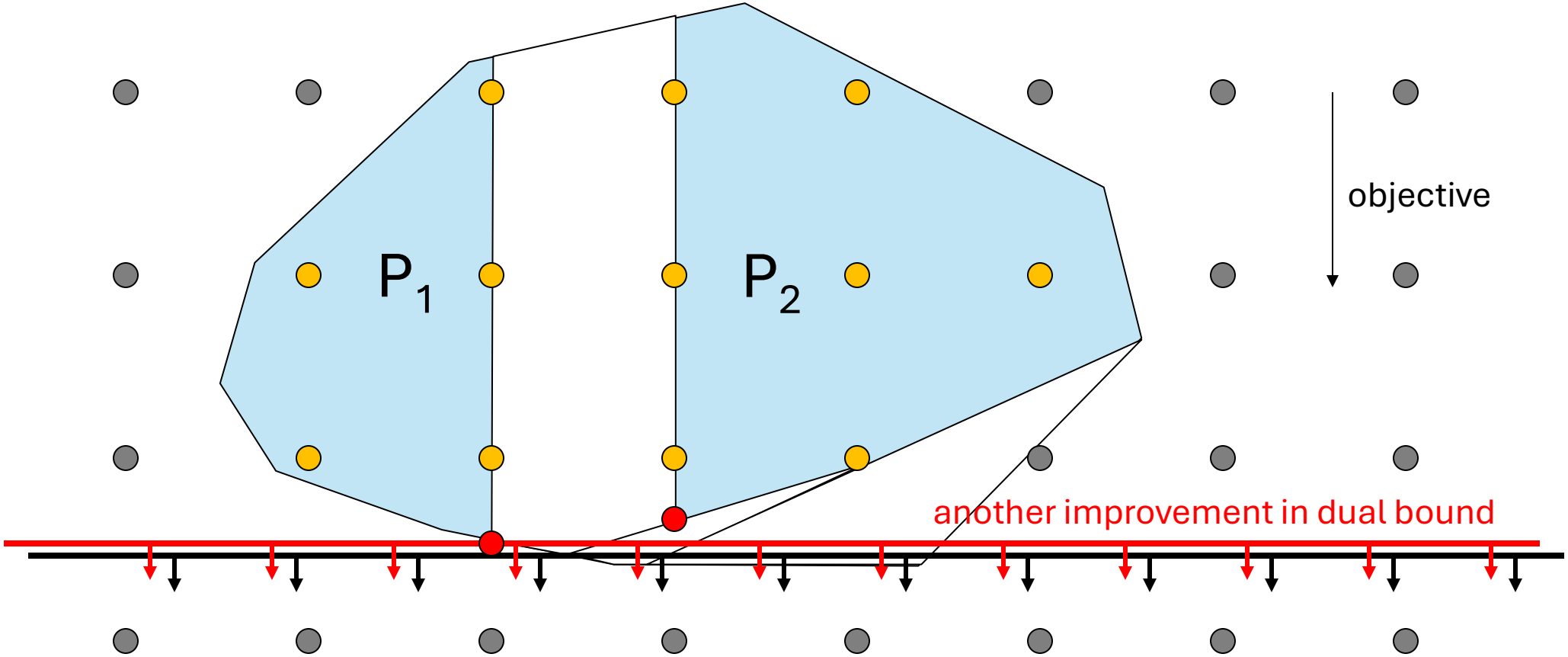
© Gurobi Optimization, LLC. All rights reserved.

MIP – Branching



© Gurobi Optimization, LLC. All rights reserved.

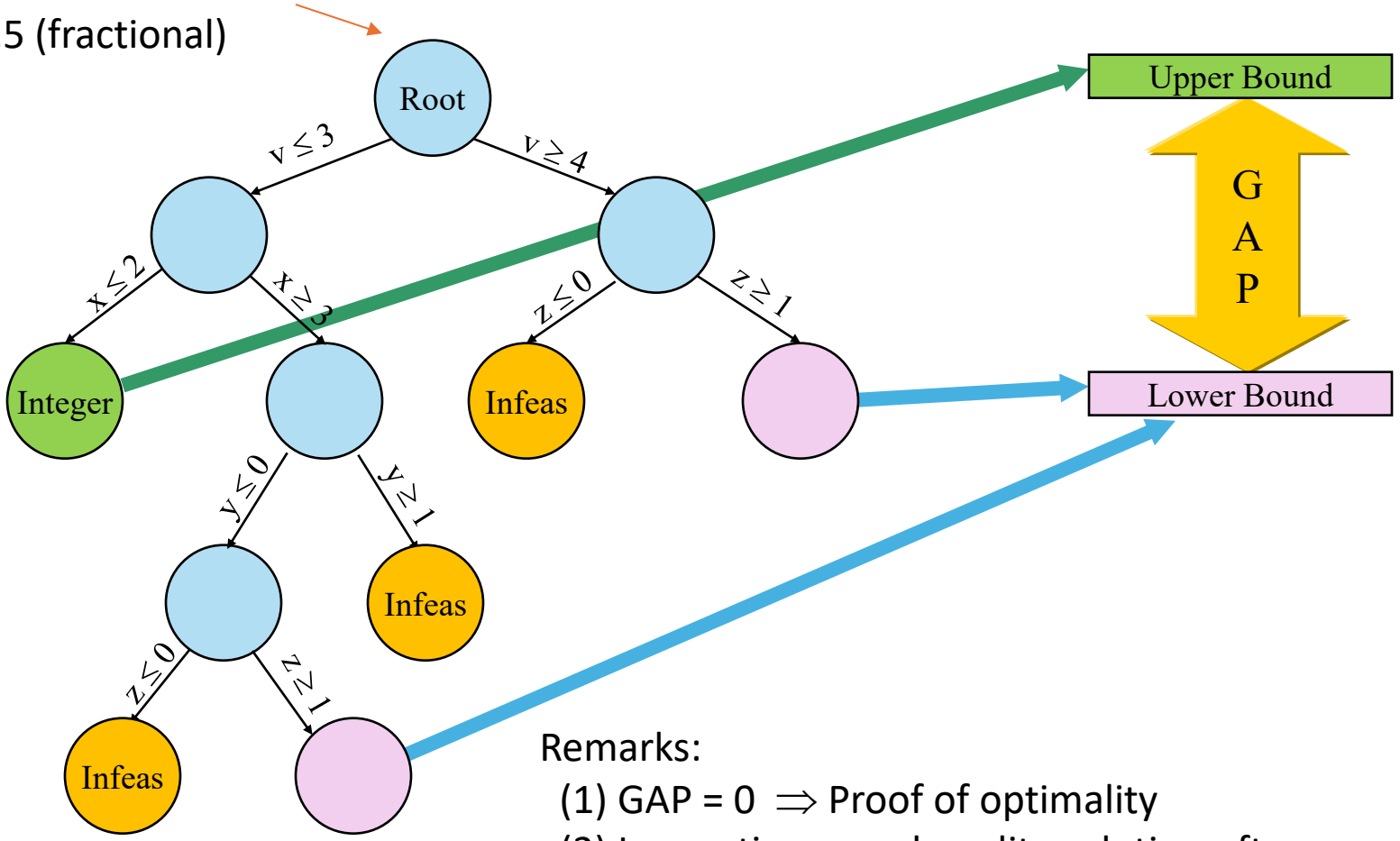
MIP – Branching



© Gurobi Optimization, LLC. All rights reserved.

LP based Branch-and-Bound

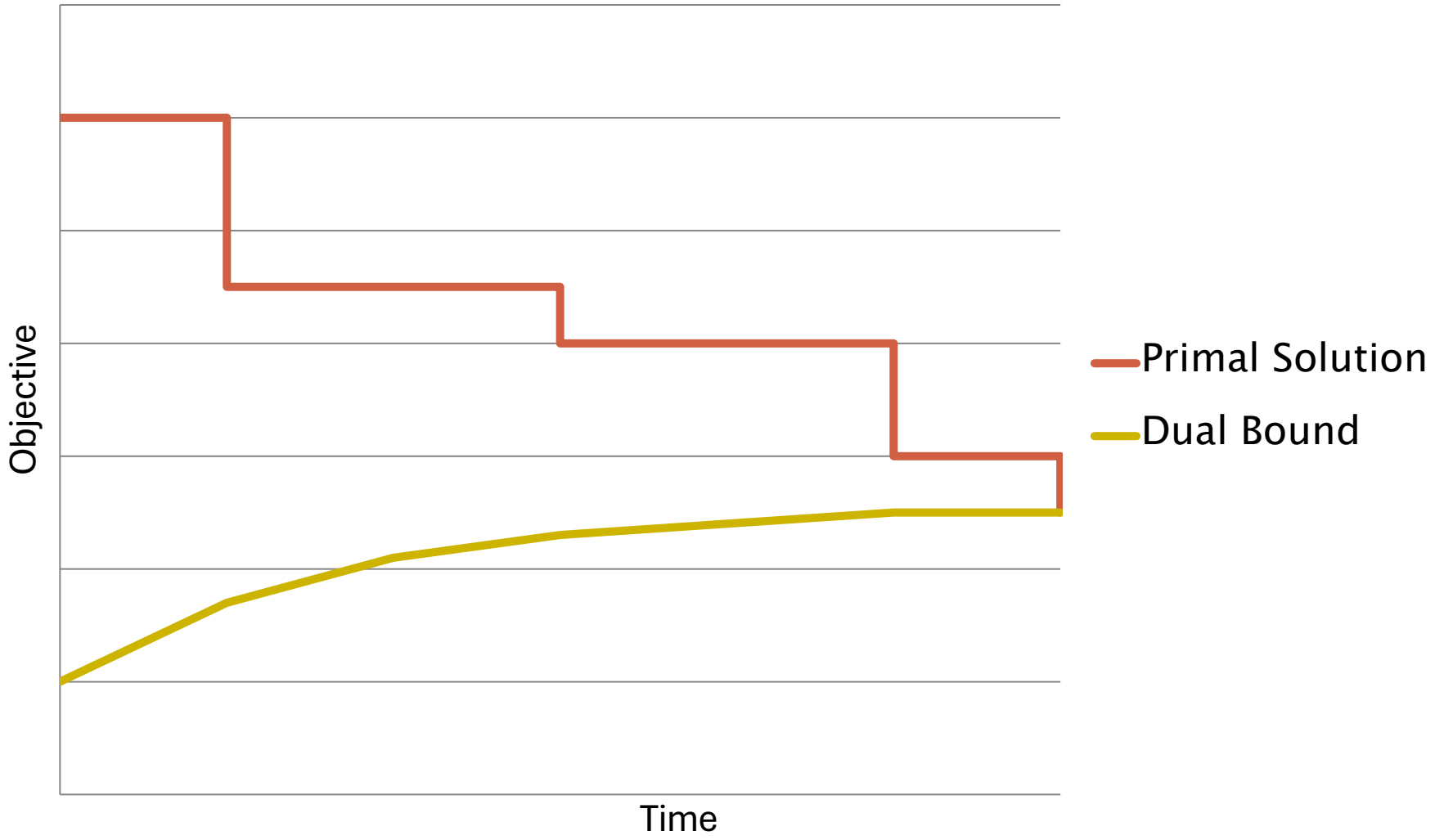
Solve LP relaxation:
v=3.5 (fractional)



Remarks:
(1) GAP = 0 ⇒ Proof of optimality
(2) In practice: good quality solution often enough

© Gurobi Optimization, LLC. All rights reserved.

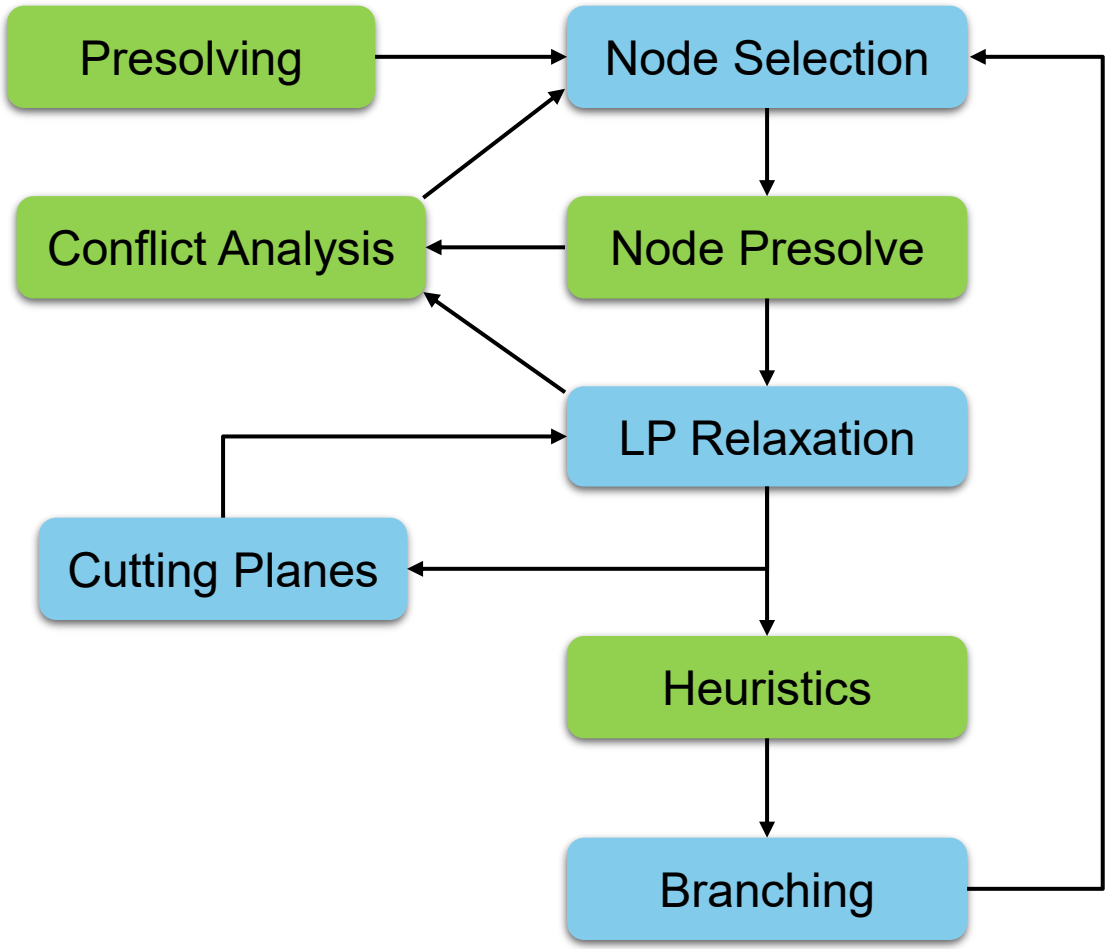
Solving a MIP Model



© Gurobi Optimization, LLC. All rights reserved.

MIP Building Blocks

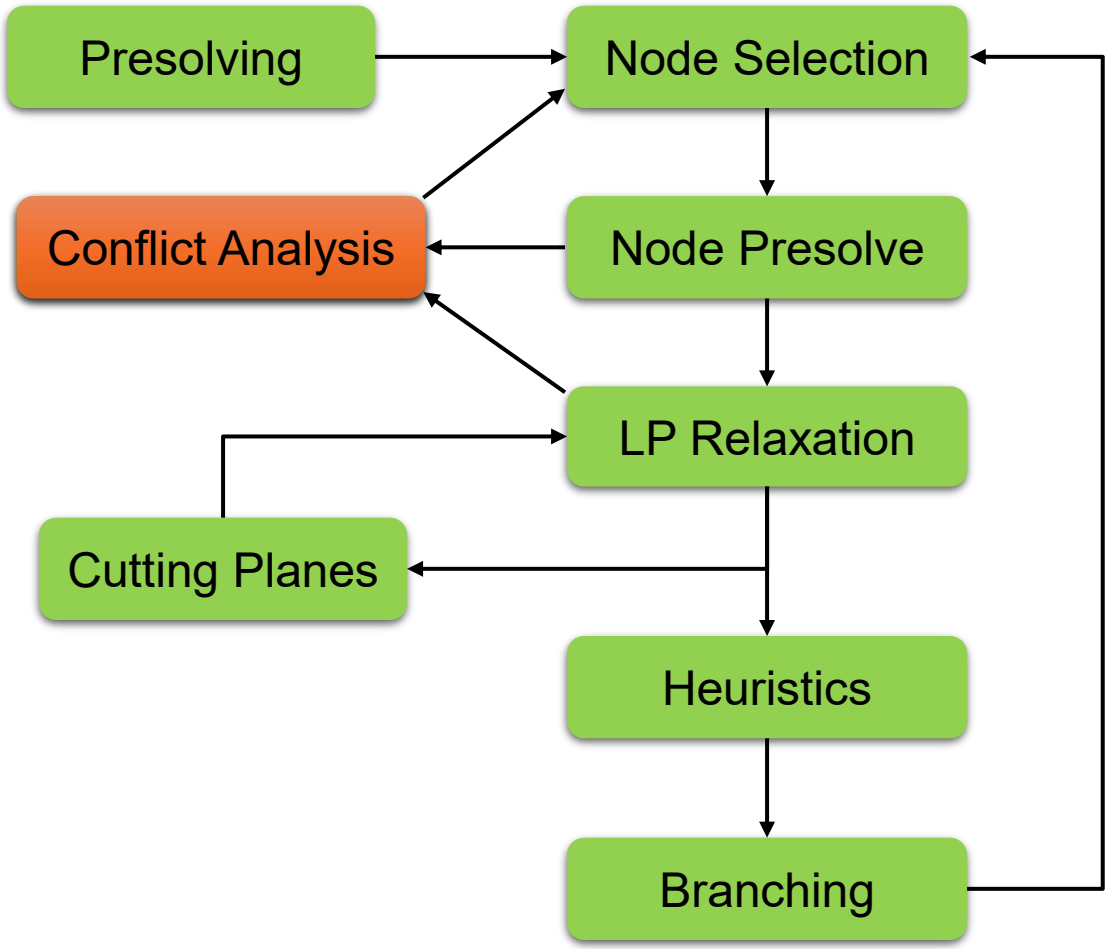
- Presolve
  - Tighten formulation and reduce problem size
- Node selection
  - Select next subproblem to process
- Node presolve
  - Additional presolve for subproblem
- Solve continuous relaxations
  - Ignoring integrality
  - Gives a bound on the optimal integral objective
- Conflict analysis
  - Learn from infeasible subproblems
- Cutting planes
  - Cut off relaxation solutions
- Primal heuristics
  - Find integer feasible solutions
- Branching variable selection
  - Crucial for limiting search tree size



© Gurobi Optimization, LLC. All rights reserved.

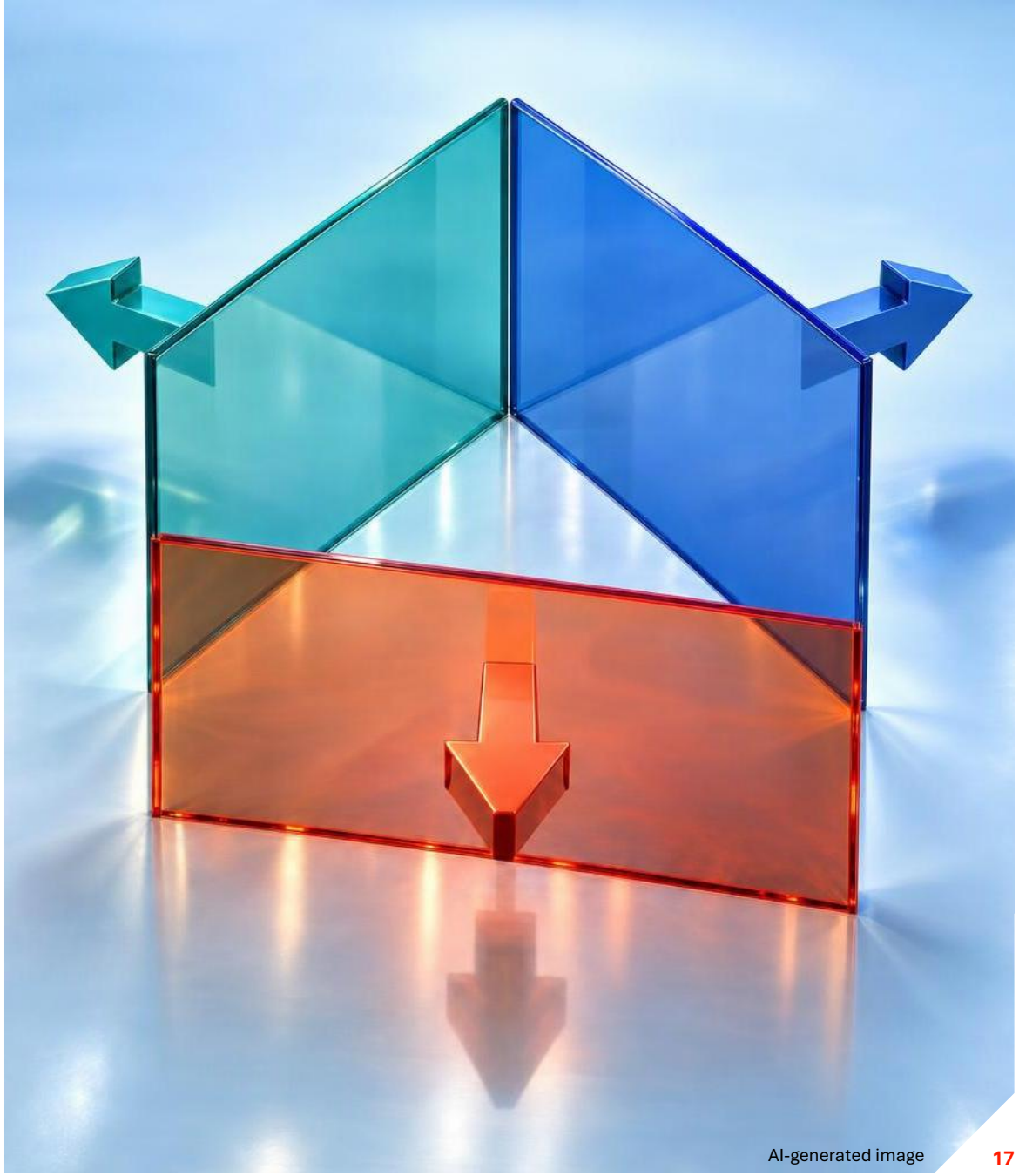
MIP Building Blocks

- Presolve
  - Tighten formulation and reduce problem size
- Node selection
  - Select next subproblem to process
- Node presolve
  - Additional presolve for subproblem
- Solve continuous relaxations
  - Ignoring integrality
  - Gives a bound on the optimal integral objective
- Conflict analysis
  - Learn from infeasible subproblems
- Cutting planes
  - Cut off relaxation solutions
- Primal heuristics
  - Find integer feasible solutions
- Branching variable selection
  - Crucial for limiting search tree size



© Gurobi Optimization, LLC. All rights reserved.

Conflict Analysis

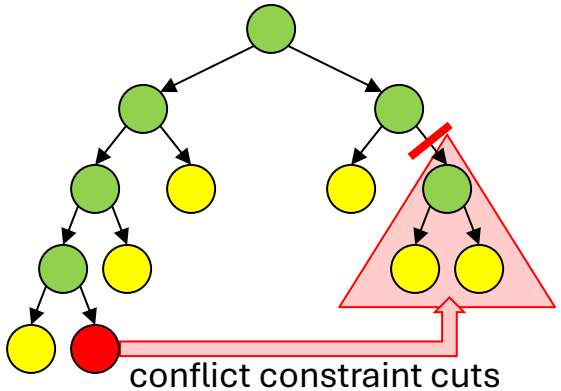


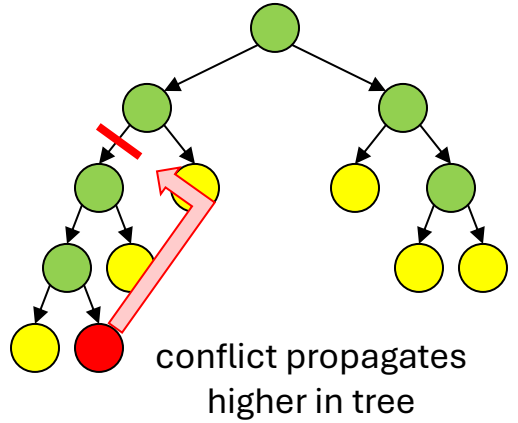
AI-generated image

© Gurobi Optimization, LLC. All rights reserved.

Conflict Analysis in MIP

- Some nodes in MIP search tree are infeasible
  - Good: node can be pruned
  - Even better: conflict analysis derives useful information
- Results of conflict analysis
  - Valid constraints cut off other parts of the search tree
  - Domain propagation
  - Cutting planes
- Non-chronological backtracking
- Statistics to guide branching decisions
  - "Hybrid Branching"





Sandholm and Shields (2006): “Nogood learning for mixed integer programming”
Achterberg (2007): “Conflict analysis in mixed integer programming“

© Gurobi Optimization, LLC. All rights reserved.

Conflict Analysis in SAT

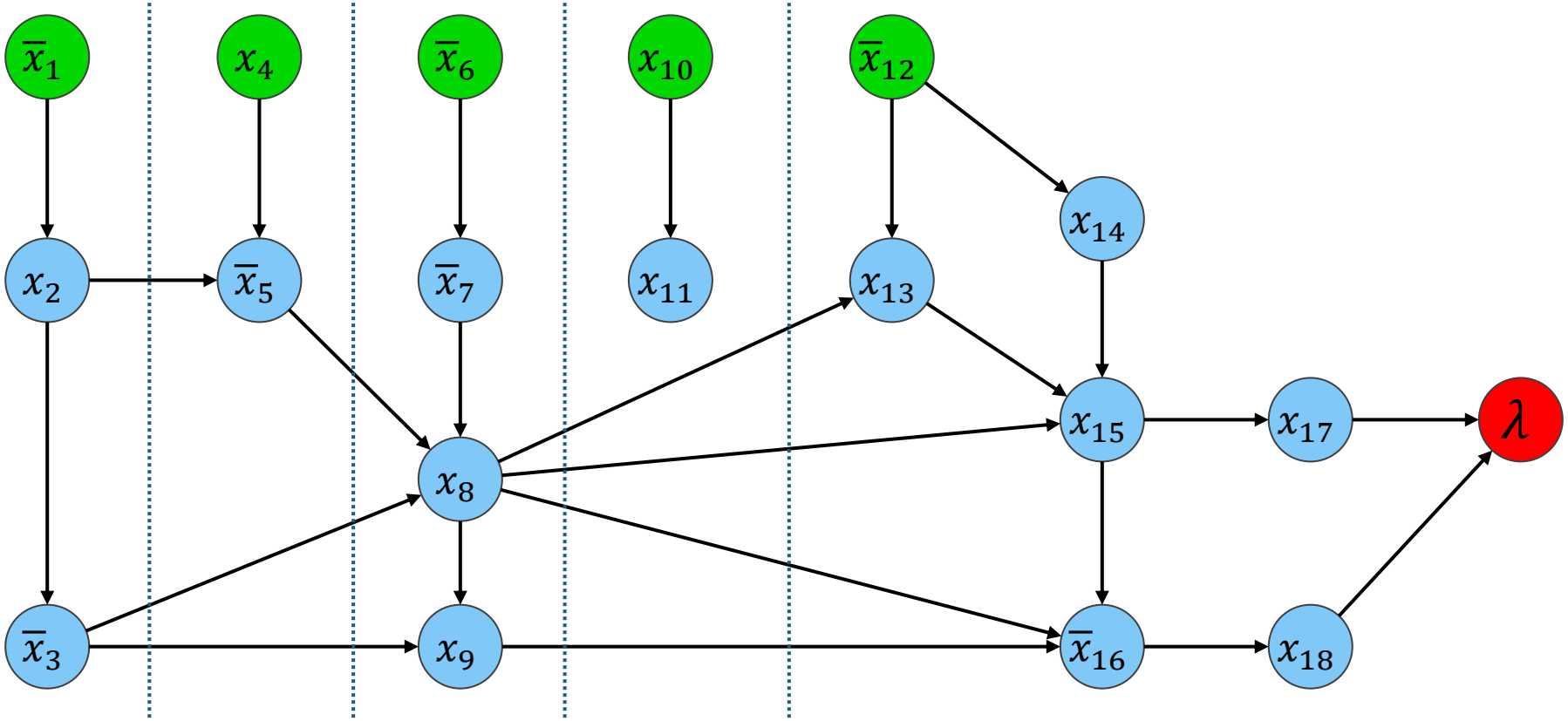
- SAT solvers apply branching, constraint propagation and backtracking
  - DPLL algorithm: Davis, Putnam, Logemann, Loveland (1960, 1961)
- Major breakthrough was invention of conflict analysis
  - CDCL: conflict driven clause learning
  - Marques-Silva and Sakallah (1996, 1999): “GRASP: A search algorithm for propositional satisfiability”

$C_1: x_1 \vee x_2$
$C_2: \bar{x}_2 \vee \bar{x}_3$
$C_3: \bar{x}_2 \vee \bar{x}_4 \vee \bar{x}_5$
$C_4: x_6 \vee \bar{x}_7$
$C_5: x_3 \vee x_5 \vee x_7 \vee x_8$
$C_6: x_3 \vee \bar{x}_8 \vee x_9$
$C_7: \bar{x}_{10} \vee x_{11}$
$C_8: \bar{x}_8 \vee x_{12} \vee x_{13}$
$C_9: x_{12} \vee x_{14}$
$C_{10}: \bar{x}_8 \vee \bar{x}_{13} \vee \bar{x}_{14} \vee x_{15}$
$C_{11}: \bar{x}_8 \vee \bar{x}_9 \vee \bar{x}_{15} \vee \bar{x}_{16}$
$C_{12}: \bar{x}_{15} \vee x_{17}$
$C_{13}: x_{16} \vee x_{18}$
$C_{14}: \bar{x}_{17} \vee \bar{x}_{18}$
$C_{15}: \bar{x}_{12} \vee x_{19}$
$C_{16}: x_7 \vee \bar{x}_{19} \vee x_{20}$
$C_{17}: x_{15} \vee \bar{x}_{20} \vee x_{21}$
$C_{18}: \bar{x}_8 \vee \bar{x}_{20} \vee \bar{x}_{21}$

Branching on $x_1 = 0, x_4 = 1, x_6 = 0, x_{10} = 1$, and $x_{12} = 0$ leads to conflict generated by C_{14}

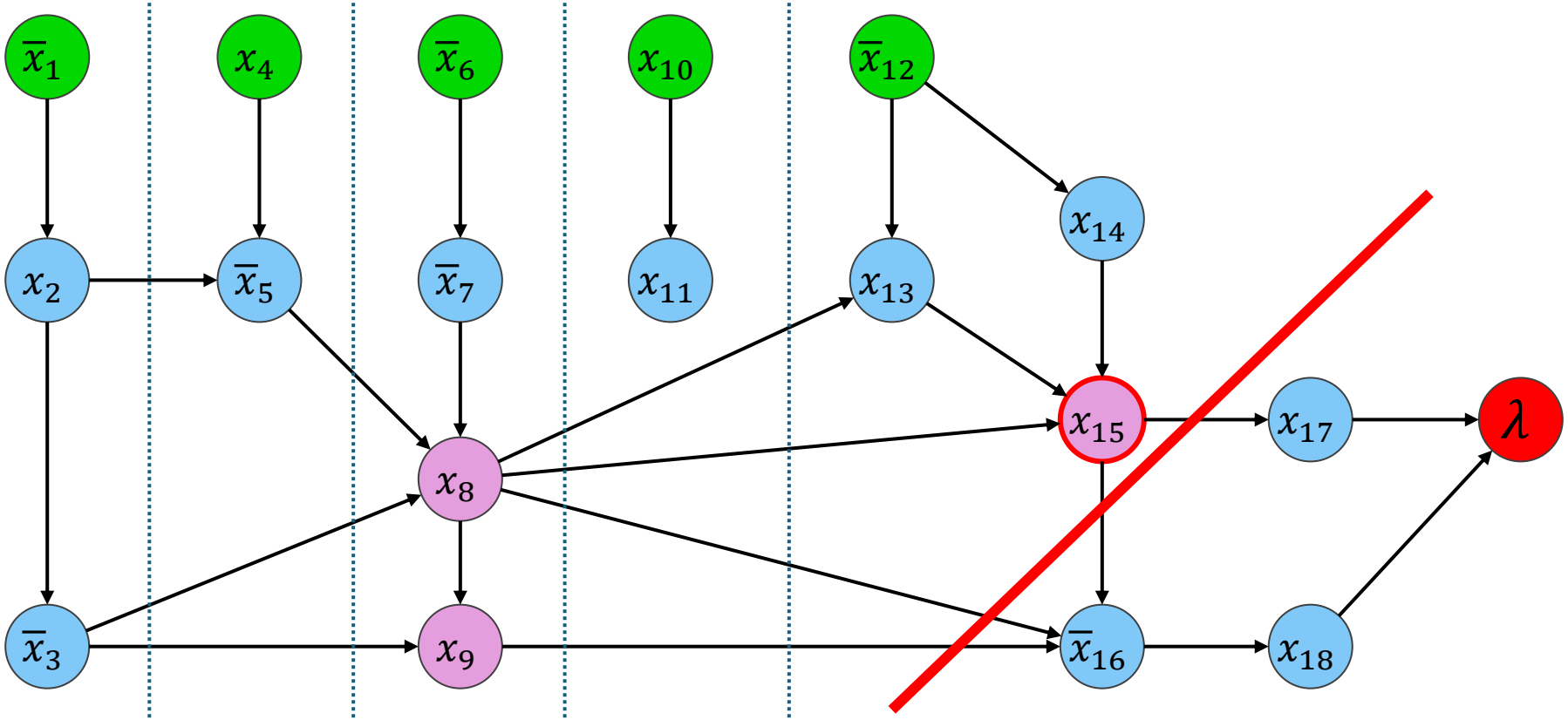
© Gurobi Optimization, LLC. All rights reserved.

Conflict Analysis in SAT



© Gurobi Optimization, LLC. All rights reserved.

Conflict Analysis in SAT

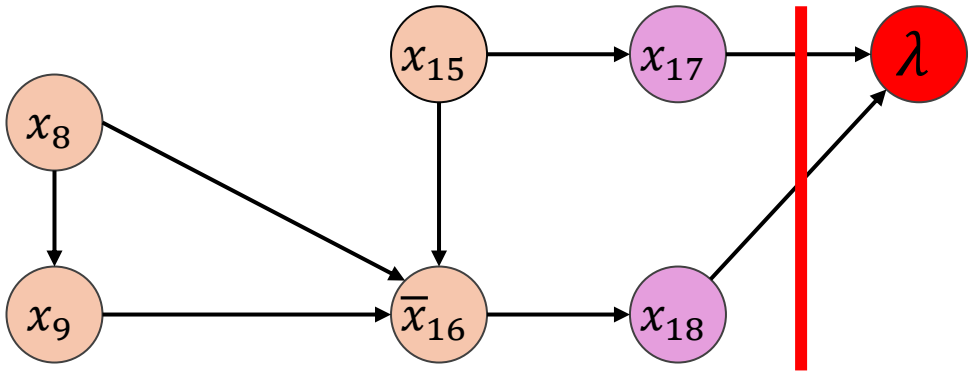


First unique implication point (FUIP) cut

$\bar{x}_8 \vee \bar{x}_9 \vee \bar{x}_{15}$

© Gurobi Optimization, LLC. All rights reserved.

Conflict Analysis in SAT and Resolution

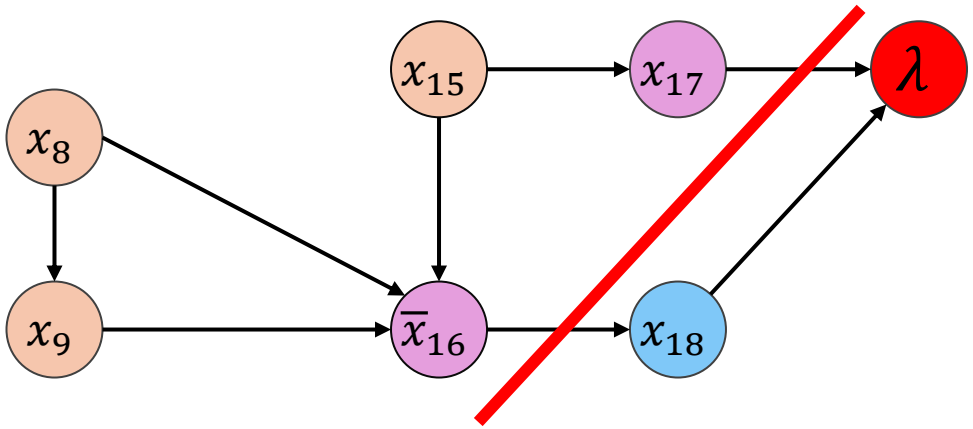


$C_1: x_1 \vee x_2$
$C_2: \bar{x}_2 \vee \bar{x}_3$
$C_3: \bar{x}_2 \vee \bar{x}_4 \vee \bar{x}_5$
$C_4: x_6 \vee \bar{x}_7$
$C_5: x_3 \vee x_5 \vee x_7 \vee x_8$
$C_6: x_3 \vee \bar{x}_8 \vee x_9$
$C_7: \bar{x}_{10} \vee x_{11}$
$C_8: \bar{x}_8 \vee x_{12} \vee x_{13}$
$C_9: x_{12} \vee x_{14}$
$C_{10}: \bar{x}_8 \vee \bar{x}_{13} \vee \bar{x}_{14} \vee x_{15}$
$C_{11}: \bar{x}_8 \vee \bar{x}_9 \vee \bar{x}_{15} \vee \bar{x}_{16}$
$C_{12}: \bar{x}_{15} \vee x_{17}$
$C_{13}: x_{16} \vee x_{18}$
$C_{14}: \bar{x}_{17} \vee \bar{x}_{18}$
$C_{15}: \bar{x}_{12} \vee x_{19}$
$C_{16}: x_7 \vee \bar{x}_{19} \vee x_{20}$
$C_{17}: x_{15} \vee \bar{x}_{20} \vee x_{21}$
$C_{18}: \bar{x}_8 \vee \bar{x}_{20} \vee \bar{x}_{21}$

$C_{14}: \bar{x}_{17} \vee \bar{x}_{18}$

© Gurobi Optimization, LLC. All rights reserved.

Conflict Analysis in SAT and Resolution



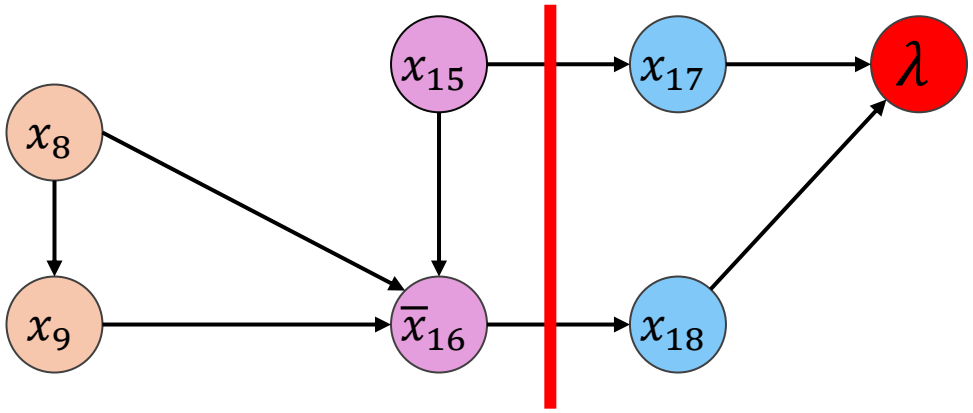
$C_1: x_1 \vee x_2$
$C_2: \bar{x}_2 \vee \bar{x}_3$
$C_3: \bar{x}_2 \vee \bar{x}_4 \vee \bar{x}_5$
$C_4: x_6 \vee \bar{x}_7$
$C_5: x_3 \vee x_5 \vee x_7 \vee x_8$
$C_6: x_3 \vee \bar{x}_8 \vee x_9$
$C_7: \bar{x}_{10} \vee x_{11}$
$C_8: \bar{x}_8 \vee x_{12} \vee x_{13}$
$C_9: x_{12} \vee x_{14}$
$C_{10}: \bar{x}_8 \vee \bar{x}_{13} \vee \bar{x}_{14} \vee x_{15}$
$C_{11}: \bar{x}_8 \vee \bar{x}_9 \vee \bar{x}_{15} \vee \bar{x}_{16}$
$C_{12}: \bar{x}_{15} \vee x_{17}$
$C_{13}: x_{16} \vee x_{18}$
$C_{14}: \bar{x}_{17} \vee \bar{x}_{18}$
$C_{15}: \bar{x}_{12} \vee x_{19}$
$C_{16}: x_7 \vee \bar{x}_{19} \vee x_{20}$
$C_{17}: x_{15} \vee \bar{x}_{20} \vee x_{21}$
$C_{18}: \bar{x}_8 \vee \bar{x}_{20} \vee \bar{x}_{21}$

$C_{14}: \bar{x}_{17} \vee \bar{x}_{18}$
$C_{13}: x_{16} \vee x_{18}$

$C_A: x_{16} \vee \bar{x}_{17}$

© Gurobi Optimization, LLC. All rights reserved.

Conflict Analysis in SAT and Resolution



$C_1: x_1 \vee x_2$
$C_2: \bar{x}_2 \vee \bar{x}_3$
$C_3: \bar{x}_2 \vee \bar{x}_4 \vee \bar{x}_5$
$C_4: x_6 \vee \bar{x}_7$
$C_5: x_3 \vee x_5 \vee x_7 \vee x_8$
$C_6: x_3 \vee \bar{x}_8 \vee x_9$

$C_7: \bar{x}_{10} \vee x_{11}$
$C_8: \bar{x}_8 \vee x_{12} \vee x_{13}$
$C_9: x_{12} \vee x_{14}$
$C_{10}: \bar{x}_8 \vee \bar{x}_{13} \vee \bar{x}_{14} \vee x_{15}$
$C_{11}: \bar{x}_8 \vee \bar{x}_9 \vee \bar{x}_{15} \vee \bar{x}_{16}$
$C_{12}: \bar{x}_{15} \vee x_{17}$

$C_{13}: x_{16} \vee x_{18}$
$C_{14}: \bar{x}_{17} \vee \bar{x}_{18}$
$C_{15}: \bar{x}_{12} \vee x_{19}$
$C_{16}: x_7 \vee \bar{x}_{19} \vee x_{20}$
$C_{17}: x_{15} \vee \bar{x}_{20} \vee x_{21}$
$C_{18}: \bar{x}_8 \vee \bar{x}_{20} \vee \bar{x}_{21}$

$C_{14}: \bar{x}_{17} \vee \bar{x}_{18}$
$C_{13}: x_{16} \vee x_{18}$

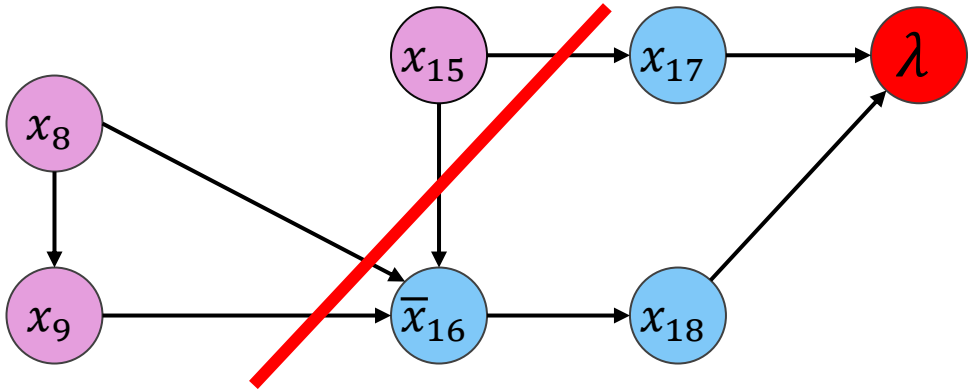
$C_A: x_{16} \vee \bar{x}_{17}$

$C_A: x_{16} \vee \bar{x}_{17}$
$C_{12}: \bar{x}_{15} \vee x_{17}$

$C_B: \bar{x}_{15} \vee x_{16}$

© Gurobi Optimization, LLC. All rights reserved.

Conflict Analysis in SAT and Resolution



$C_1: x_1 \vee x_2$
$C_2: \bar{x}_2 \vee \bar{x}_3$
$C_3: \bar{x}_2 \vee \bar{x}_4 \vee \bar{x}_5$
$C_4: x_6 \vee \bar{x}_7$
$C_5: x_3 \vee x_5 \vee x_7 \vee x_8$
$C_6: x_3 \vee \bar{x}_8 \vee x_9$
$C_7: \bar{x}_{10} \vee x_{11}$
$C_8: \bar{x}_8 \vee x_{12} \vee x_{13}$
$C_9: x_{12} \vee x_{14}$
$C_{10}: \bar{x}_8 \vee \bar{x}_{13} \vee \bar{x}_{14} \vee x_{15}$
$C_{11}: \bar{x}_8 \vee \bar{x}_9 \vee \bar{x}_{15} \vee \bar{x}_{16}$
$C_{12}: \bar{x}_{15} \vee x_{17}$
$C_{13}: x_{16} \vee x_{18}$
$C_{14}: \bar{x}_{17} \vee \bar{x}_{18}$
$C_{15}: \bar{x}_{12} \vee x_{19}$
$C_{16}: x_7 \vee \bar{x}_{19} \vee x_{20}$
$C_{17}: x_{15} \vee \bar{x}_{20} \vee x_{21}$
$C_{18}: \bar{x}_8 \vee \bar{x}_{20} \vee \bar{x}_{21}$

$C_{14}: \bar{x}_{17} \vee \bar{x}_{18}$
$C_{13}: x_{16} \vee x_{18}$

$C_A: x_{16} \vee \bar{x}_{17}$

$C_A: x_{16} \vee \bar{x}_{17}$
$C_{12}: \bar{x}_{15} \vee x_{17}$

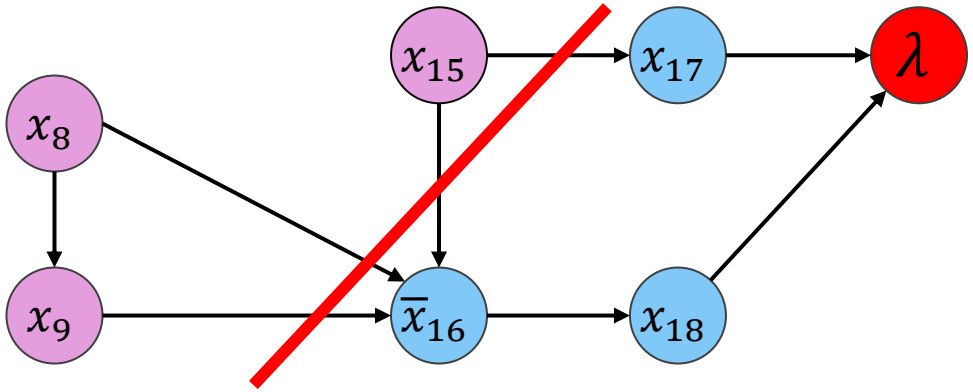
$C_B: \bar{x}_{15} \vee x_{16}$

$C_B: \bar{x}_{15} \vee x_{16}$
$C_{11}: \bar{x}_8 \vee \bar{x}_9 \vee \bar{x}_{15} \vee \bar{x}_{16}$

$C_C: \bar{x}_8 \vee \bar{x}_9 \vee \bar{x}_{15}$

© Gurobi Optimization, LLC. All rights reserved.

Expressed in Linear Constraints



$C_1: x_1 \vee x_2$
$C_2: \bar{x}_2 \vee \bar{x}_3$
$C_3: \bar{x}_2 \vee \bar{x}_4 \vee \bar{x}_5$
$C_4: x_6 \vee \bar{x}_7$
$C_5: x_3 \vee x_5 \vee x_7 \vee x_8$
$C_6: x_3 \vee \bar{x}_8 \vee x_9$

$C_7: \bar{x}_{10} \vee x_{11}$
$C_8: \bar{x}_8 \vee x_{12} \vee x_{13}$
$C_9: x_{12} \vee x_{14}$
$C_{10}: \bar{x}_8 \vee \bar{x}_{13} \vee \bar{x}_{14} \vee x_{15}$
$C_{11}: \bar{x}_8 \vee \bar{x}_9 \vee \bar{x}_{15} \vee \bar{x}_{16}$
$C_{12}: \bar{x}_{15} \vee x_{17}$

$C_{13}: x_{16} \vee x_{18}$
$C_{14}: \bar{x}_{17} \vee \bar{x}_{18}$
$C_{15}: \bar{x}_{12} \vee x_{19}$
$C_{16}: x_7 \vee \bar{x}_{19} \vee x_{20}$
$C_{17}: x_{15} \vee \bar{x}_{20} \vee x_{21}$
$C_{18}: \bar{x}_8 \vee \bar{x}_{20} \vee \bar{x}_{21}$

$$\frac{\begin{array}{l} C_{14}: +x_{17} + x_{18} \leq 1 \\ C_{13}: -x_{16} - x_{18} \leq -1 \end{array}}{C_A: \; -x_{16} + x_{17} \leq 0}$$

$$\frac{\begin{array}{l} C_A: \; -x_{16} + x_{17} \leq 0 \\ C_{12}: +x_{15} - x_{17} \leq 0 \end{array}}{C_B: \; +x_{15} - x_{16} \leq 0}$$

$$\frac{\begin{array}{ll} C_B: & +x_{15} - x_{16} \leq 0 \\ C_{11}: & +x_8 + x_9 + x_{15} + x_{16} \leq 3 \end{array}}{\begin{array}{ll} & +x_8 + x_9 + 2x_{15} \leq 3 \end{array}}$$

$$C_C: \; +x_8 + x_9 + x_{15} \leq 2$$

Knapsack cover cut

© Gurobi Optimization, LLC. All rights reserved.

Non-Binary Variables

- Mixed integer programming also includes general integer and continuous variables
- Knapsack cover cuts would not apply for constraints involving such variables
- Use mixed integer rounding (MIR) cuts instead
  - One of the most powerful cutting planes in MIP
- Derives simple knapsack cover cuts that we need for pure binary constraints:

aggregated constraint:	$x_8 + x_9 + 2x_{15} \leq 3$
complement variables:	$-(1 - x_8) - (1 - x_9) - 2(1 - x_{15}) \leq -1$
	$-\bar{x}_8 - \bar{x}_9 - 2\bar{x}_{15} \leq -1$
divide by max coefficient:	$-\frac{1}{2}\bar{x}_8 - \frac{1}{2}\bar{x}_9 - \bar{x}_{15} \leq -\frac{1}{2}$
apply MIR formula:	$-\bar{x}_8 - \bar{x}_9 - \bar{x}_{15} \leq -1$
back to original variables:	$x_8 + x_9 + x_{15} \leq 2$

MIR Cuts

$$\sum_{j \in I} (a_j + f_j) x_j + \sum_{j \in C} d_j x_j \leq a_0 + f_0, \quad x \geq 0$$

with integral vectors a and fractional part $0 \leq f < 1$, integer variables I, continuous variables C

implies cutting plane:

$$\sum_{j \in I} \left(a_j + \frac{\max\{f_j - f_0, 0\}}{1 - f_0} \right) x_j + \sum_{j \in C} \frac{\min\{d_j, 0\}}{1 - f_0} x_j \leq a_0$$

© Gurobi Optimization, LLC. All rights reserved.

Conflict Analysis: SAT vs. MIP

SAT

- Fix variable as branching decision
- Propagate clauses to find further fixings
  - For each propagation, record „reason clause“
- If infeasibility discovered:
  - Backwards traversal of implication graph, starting with violated clause
  - Apply resolution with „reason clause“
    - SAT clause resolution of current conflict clause and „reason clause“
  - Repeat until first unique implication point
- Add resulting conflict clause to clause pool
  - Use for clause propagation

MIP

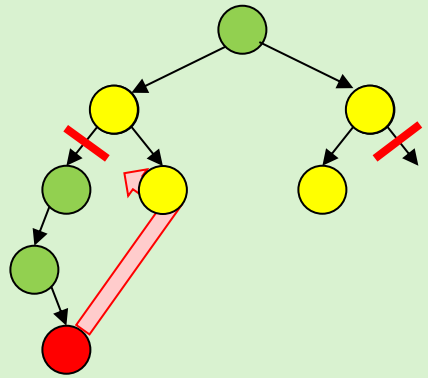
- Tighten bound as branching decision
- Propagate constraints to find further bound strengthenings
  - For each propagation, record „reason constraint“
- If infeasibility discovered:
  - Backwards traversal of implication graph, starting with violated constraint
  - Apply resolution with „reason constraint“
    - Add „reason constraint“ to current conflict constraint to cancel propagated variable
    - Apply MIR formula on resulting constraint
  - Repeat until first unique implication point
- Add resulting conflict constraint to constraint pool
  - Use for constraint propagation and cutting planes

© Gurobi Optimization, LLC. All rights reserved.

Using Conflict Constraints

SAT

- SAT uses depth-first search tree exploration



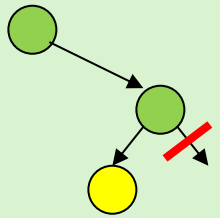
  - No need to create sibling nodes
  - Just dive down a path until a conflict is hit
    - Store conflict clause
    - Backtrack
    - Continue

© Gurobi Optimization, LLC. All rights reserved.

Using Conflict Constraints

SAT

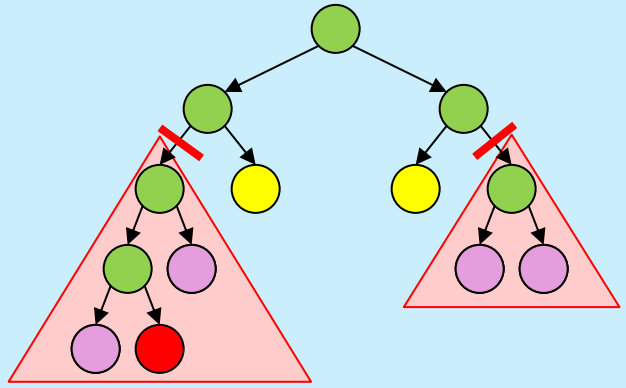
- SAT uses depth-first search tree exploration



  - No need to create sibling nodes
  - Just dive down a path until a conflict is hit
    - Store conflict clause
    - Backtrack
    - Continue

MIP

- MIP uses best-first search with limited diving



  - Multiple open leaf nodes exist
  - Some node results in a conflict
    - Store conflict constraint
    - Affected nodes are pruned by propagating conflict constraint while processing nodes

© Gurobi Optimization, LLC. All rights reserved.

Conflict Constraint Pool Management

- Number of conflict constraints can grow rapidly over time
  - Expensive to propagate huge number of conflict constraints
  - Memory consumption can grow significantly
- Need to discard useless conflict constraints to limit growth of pool
- Depth-first search is easy
  - Most recent ones are useful
  - If conflict constraint did not trigger for a while, discard it
- Best-first search with diving is more complicated
  - During dive: similar to depth-first-search
  - After jumping to different region
    - Recent conflict constraint may not be useful
    - Useful conflict constraints may have already been discarded
- Parallelization in MIP
  - Different threads work on different parts of search tree
  - Keep conflict constraints local to each thread
    - Do not share with other threads
    - No joint pool with thread locking required
    - No determinism issues

© Gurobi Optimization, LLC. All rights reserved.

Conflicts from Infeasible LP Relaxations

- Most frequent reason for pruning a node in MIP is not domain propagation but LP relaxation
  - LP relaxation is infeasible
  - LP relaxation has dual bound that is not better than incumbent value („cutoff bound“)
- Infeasibility of LP is proven by providing a dual ray

Farkas Lemma: $Ax \leq b, l \leq x \leq u$ is infeasible $\Leftrightarrow$ $\exists y, r_l, r_u \geq 0\colon\ y^T A - r_l + r_u = 0\ \wedge\ y^T b - r_l^T l + r_u^T u < 0$

- A dual ray for an infeasible LP is found naturally by the dual simplex algorithm
- The constraint

$$y^T A x \leq y^T b$$

  is globally valid and infeasible for the current (local) bounds $l \leq x \leq u$

- Same could be done for „cutoff bound“ case, but usually the cutoff proof is dense and not that useful

Achterberg (2007): „Conflict analysis in mixed integer programming“

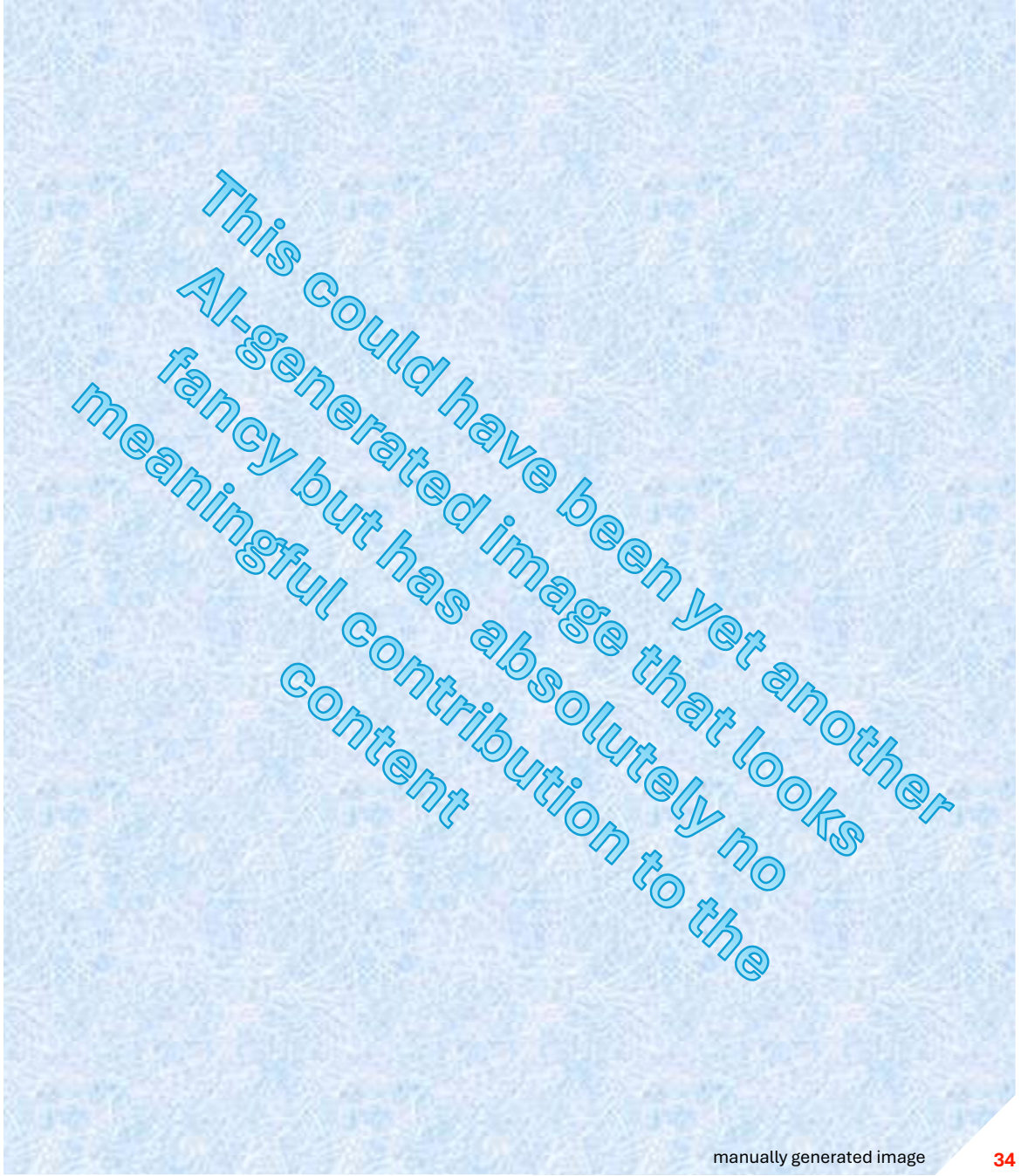
© Gurobi Optimization, LLC. All rights reserved.

Further Use of Conflict Constraints in MIP

- Add violated conflict constraints directly to the LP relaxation
- Separate cutting planes on top of conflict constraints
  - Knapsack covers, MIR cuts, and other cuts for the single conflict constraint
  - Use conflict constraints in aggregation heuristics for MIR, zero-half and other cut separators
- Could use conflict constraints to influence branching variable selection
- Could use conflict constraints to guide primal heuristics
- Could use conflict constraints to decide to trigger a restart

© Gurobi Optimization, LLC. All rights reserved.

Restarts



manually generated image

© Gurobi Optimization, LLC. All rights reserved.

Restarts in SAT

- Gomes, Selman, Kautz (1998): „ Boosting combinatorial search through randomization”
  - Another major breakthrough in SAT solving
  - For many SAT instances, a randomized SAT solver has a heavy-tailed runtime distribution
  - Rapid restart strategy is preferred strategy to obtain minimal expected runtime
    - Several orders of magnitude speedup observed on test cases
  - Increases chance to find a „backdoor“ to the SAT instance
    - See Williams, Gomes and Selman (2003): „ On the connections between backdoors, restarts, and heavytailedness in combinatorial search”
- Combines very well with conflict analysis
  - Conflict clauses encode every failure that was encountered during the search
  - All knowledge about problem instance is stored in conflicts
  - Nothing gets lost when restarting
  - Branching strategy after restart can take conflict clauses into account

© Gurobi Optimization, LLC. All rights reserved.

Restarts in MIP

- Much more difficult than in SAT
  - Root node solve is very expensive
    - LP relaxation solve
    - Cutting planes
    - Heuristics
    - Strong branching
  - Best-first search yields many open leaf nodes
    - Only pruned nodes are encoded by conflicts
    - Work done in ancestors of open and pruned leaf nodes is lost
  - No conflict constraints for nodes pruned by LP relaxation‘s dual bound

- Opportunities
  - Apply more presolve after having found good incumbent solution
  - Reuse cutting planes found in previous runs
  - Reuse implied bounds and cliques derived in previous runs
  - May learn something about branching strategy
  - Like in SAT: increases chance to find backdoor

© Gurobi Optimization, LLC. All rights reserved.

Restart Types in MIP

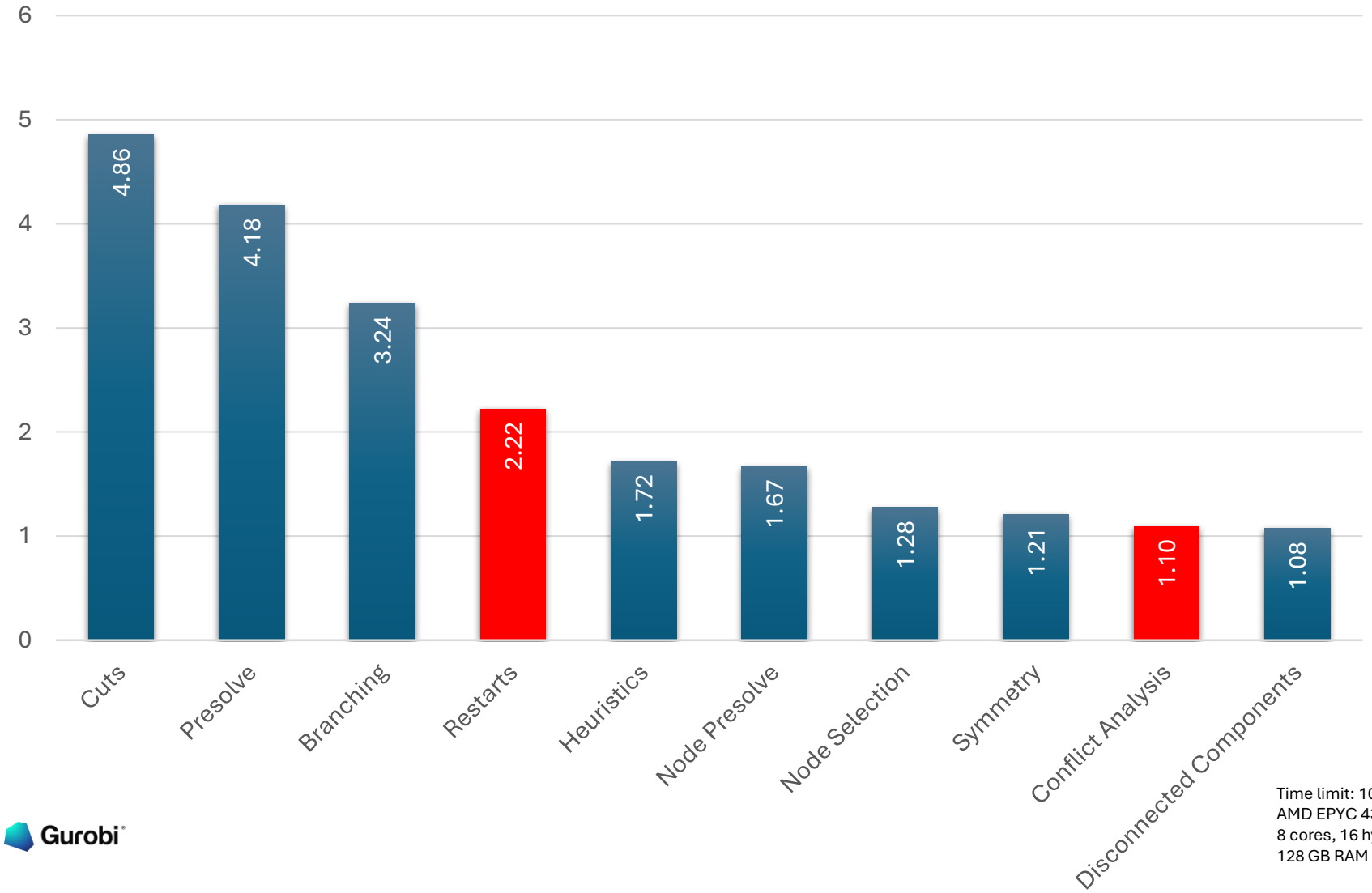
Presolve restart	Root restart	Tree restart
○ Presolve significantly reduced size	○ Found many bound changes and fixings	○ Search tree seems to get too large

○ Restart presolve to clean up data structures
  ○ Get rid of all discarded variables and constraints
  ○ Reset presolve work limits to smaller problem size
	○ Restart to apply presolve again
	  ○ Will usually find even more problem reductions
		○ Restart to hope for better luck in branching strategy
		○ May adjust strategy (e.g., branching)

○ Overhead: small
  ○ Does not lose any information that we collected
  ○ Apply it always if size got reduced enough
	○ Overhead: moderate
	  ○ Need to solve initial LP relaxation again
	  ○ Apply it almost always if enough fixings had been found
		○ Overhead: big
		  ○ Lose lots of information stored in partial search tree
		  ○ Apply it only very carefully and rarely

© Gurobi Optimization, LLC. All rights reserved.

Performance Impact of Solver Components



- Performance degradation for disabling a certain component in Gurobi 13
  - Branching: use random branching
  - Node selection: use depth-first-search

Time limit: 10000 sec.
AMD EPYC 4364P CPU @ 4.50GHz
8 cores, 16 hyper-threads
128 GB RAM

Test set has 10416 models:
- 355 discarded due to inconsistent answers
- 59 discarded due to assertion failures
- 2906 discarded that none of the versions can solve
- speed-up measured on >100s bracket: 3765 models

Gurobi

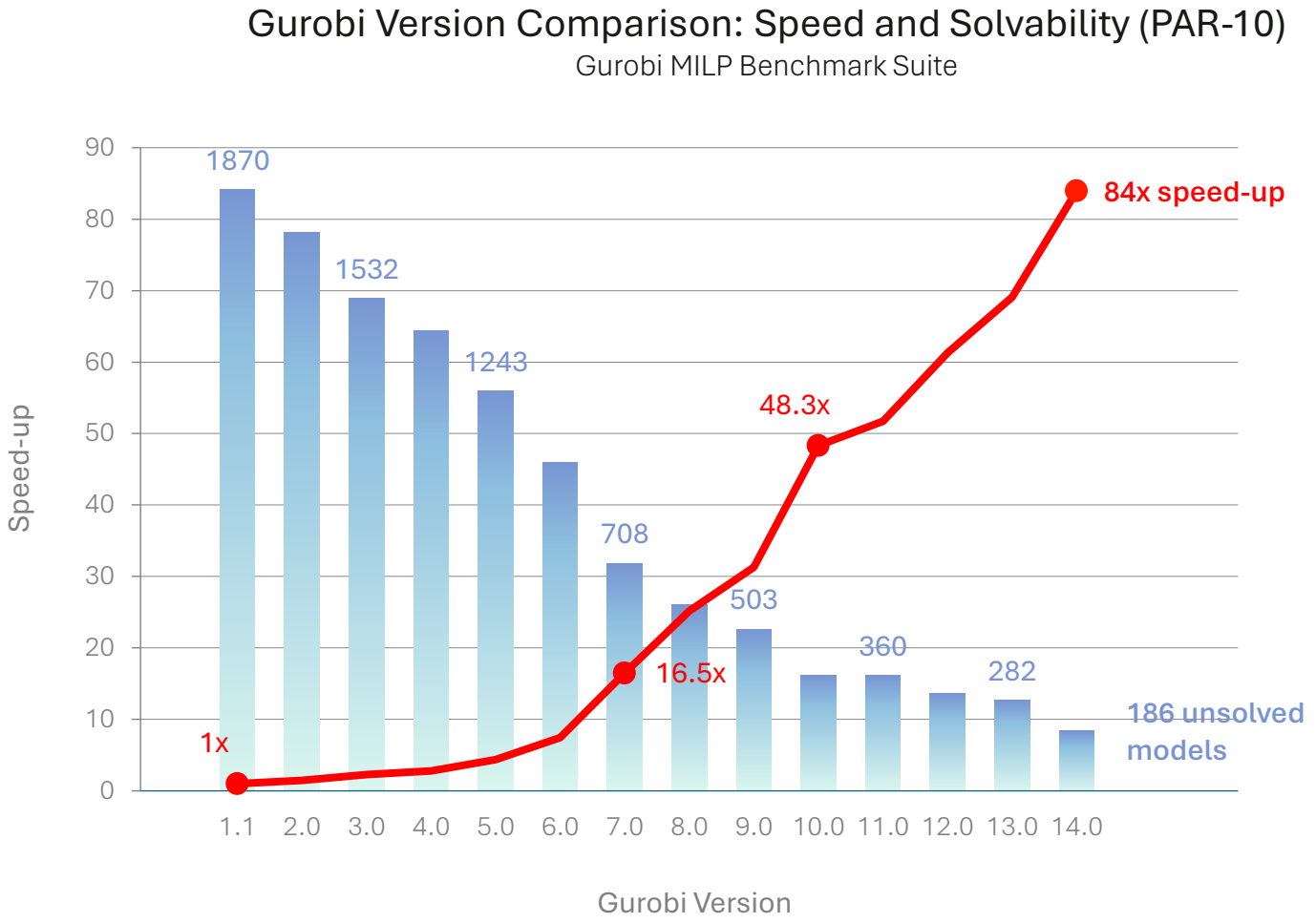
© Gurobi Optimization, LLC. All rights reserved.

MILP

Performance Evolution

In Gurobi's MILP benchmark suite, the latest version delivers:

- An **84x** speed-up over version 1.1 in geometric mean (PAR-10) of runtimes
- Only **186** models remain **unsolved** after 10,000 seconds with the latest version. The test set consists of all models that can be solved by at least one version.



Time limit: 10000 sec.
AMD EPYC 4364P CPU @ 4.50GHz
8 cores, 16 hyper-threads
128 GB RAM

Test set has 9391 models:

- 1044 discarded due to inconsistent answers
- 2355 discarded that none of the versions can solve
- Speed-up measured on >100s bracket: 3370 models

Gurobi

© Gurobi Optimization, LLC. All rights reserved.