Algorithms I – Basics

What's Inside Gurobi Optimizer

GUROBI OPTIMIZATION

- Algorithms for continuous optimization
- Algorithms for discrete optimization
- Automatic presolve for both LP and MIP
- Algorithms to analyze infeasible models
- Automatic parameter tuning tool
- Parallel and distributed parallel support
- Gurobi Compute Server
- Gurobi Instant Cloud
- Programming interfaces
- Gurobi modeling language based on Python
- Full-featured interactive shell

Gurobi LP Algorithms

Continuous: LP / QP / QCP

- Presolve
- Primal & dual simplex method
  - Numerically stable (most challenging part)
  - Easy to restart after a model modification
- Parallel barrier method with crossover
  - Can effectively exploit multiple cores
- Concurrent optimization
  - Run both simplex and barrier simultaneously
  - Solution is reported by first one to finish
  - Great use of multiple CPU cores
  - Best mix of speed and robustness
  - Deterministic and non-deterministic versions available

ConcurrentSettings

Presolve

- Goal
  - Reduce the problem size
- Example

$x + y + z \leq 5$ (1)
$u - x - z = 0$ (2)
.....
$0 \leq x, y, z \leq 1$ (3)
u is free (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 $x + z$

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:

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

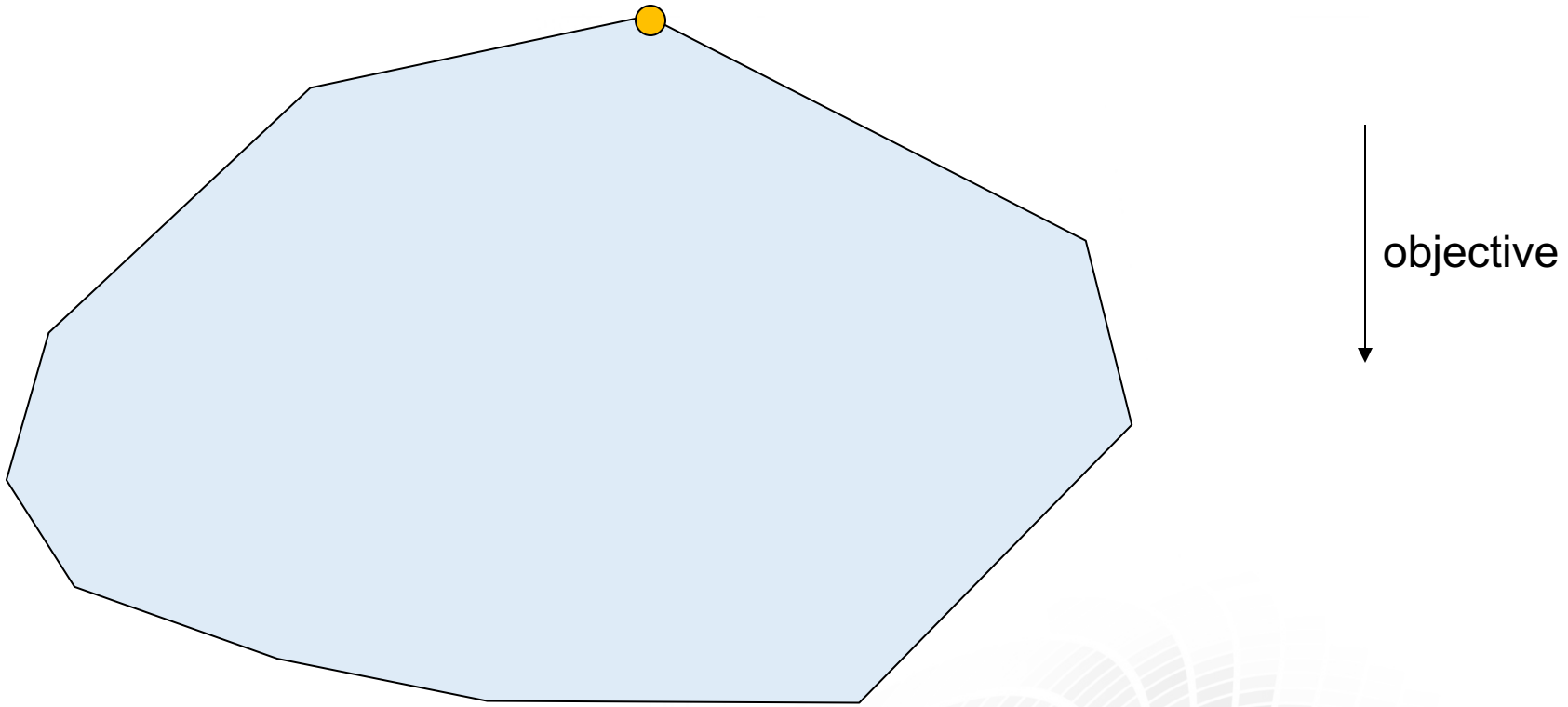
(if primal and dual are both feasible)

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: $x^T z = 0$

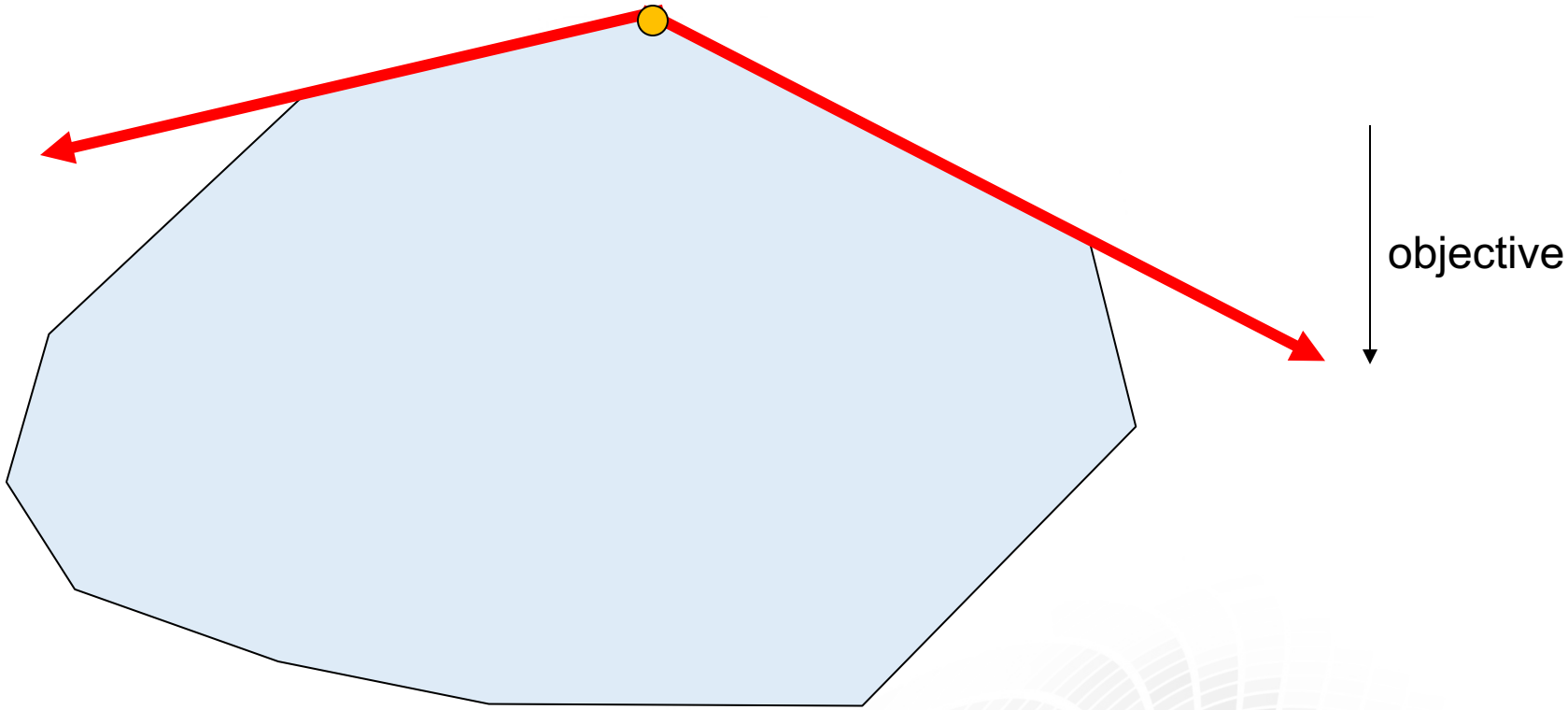
	Primal feas	Dual feas	Complementarity
Primal simplex	Maintain	Goal	Maintain
Dual simplex	Goal	Maintain	Maintain
Barrier	Goal	Goal	Goal

Simplex Algorithm



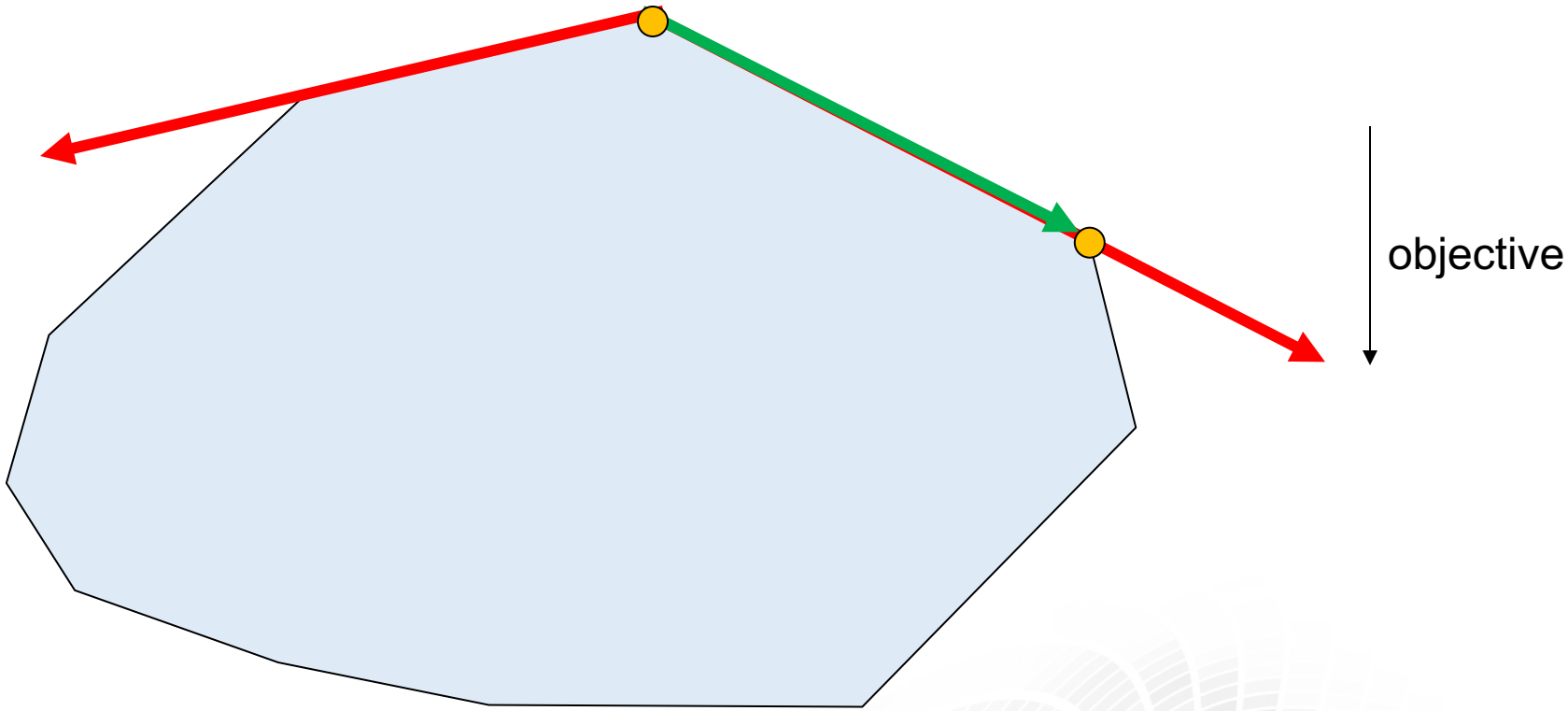
- **Phase 1**: find some feasible vertex solution

Simplex Algorithm



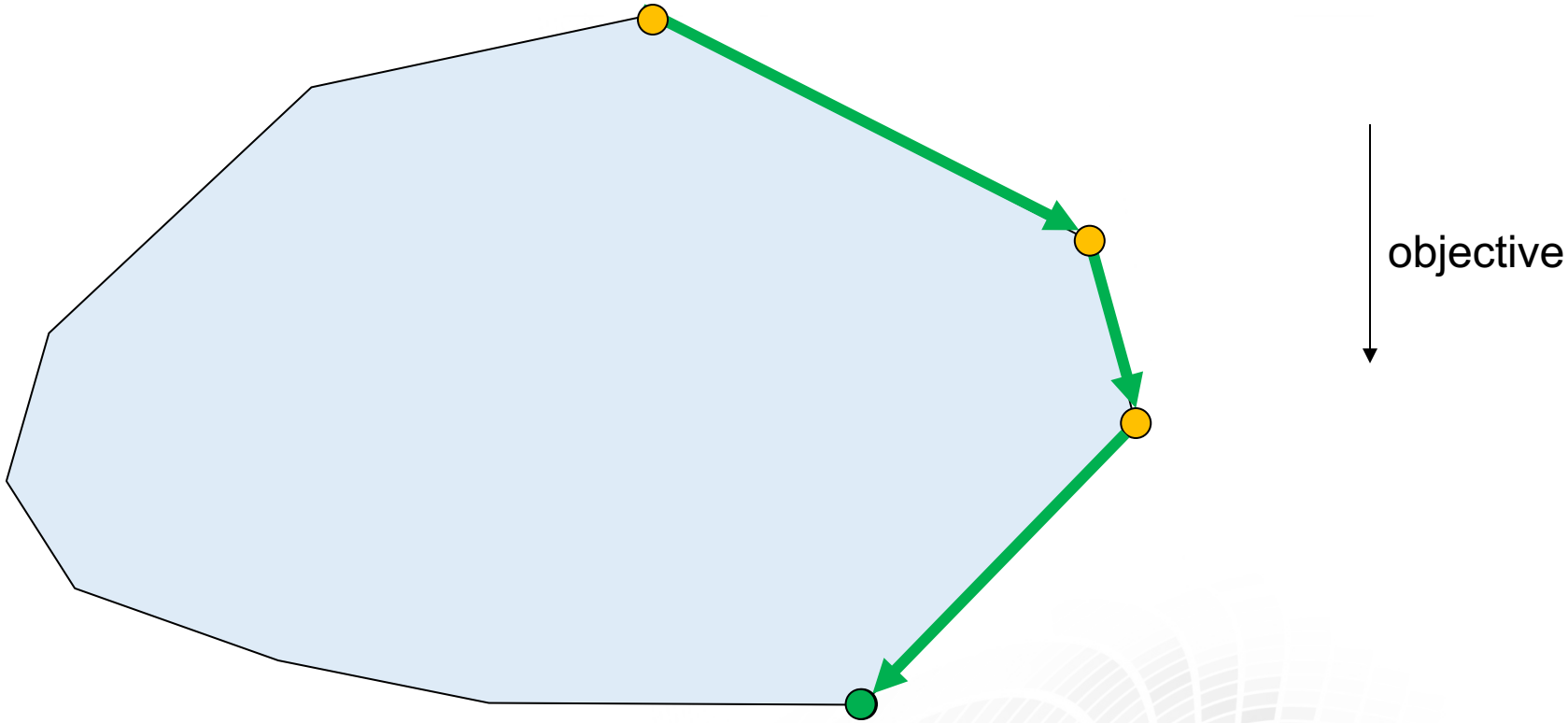
- **Pricing**: find directions in which objective improves and select one of them

Simplex Algorithm



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

Simplex Algorithm



- Iterate until no more improving direction is found

Simplex Log

```
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    1.7748600e+04   6.627132e+03   0.000000e+00      0s
    9643    1.1574611e+07   1.418653e+03   0.000000e+00      5s
   14440    1.1607748e+07   4.793500e+00   0.000000e+00     10s
   15213    1.1266396e+07   0.000000e+00   0.000000e+00     11s

Solved in 15213 iterations and 10.86 seconds
Optimal objective  1.126639605e+07
```

Simplex Algorithm – Linear Algebra

- Primal feasibility constraints

$$Ax = b$$

- Partition into basic and non-basic variables
  - Non-basic structural variables correspond to tight bounds
  - Non-basic slack variables correspond to tight constraints

$$Bx_B + Nx_N = b$$

- Solve for basic variables

$$x_B = B^{-1}\left(b - Nx_N\right)$$

- Solved by maintaining

$$B = LU$$

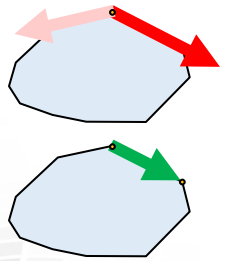
$$\begin{bmatrix} B & N \end{bmatrix} \cdot \begin{bmatrix} x_B \\ x_N \end{bmatrix} = \begin{bmatrix} b \end{bmatrix}$$

Primal Simplex Algorithm – Pivoting

$$Bx_B = b - Nx_N$$

$$\begin{aligned} B &= LU \\ LUx_B &= b - Nx_N \\ Lw &= b - Nx_N \\ Ux_B &= w \end{aligned}$$

- Simplex pivot:
  - Choose a non-basic variable to enter the basis (Pricing)
    - Pick one with a negative reduced cost
  - Push one variable out of the basis (Ratio test)
  - Update primal and dual variables, reduced costs, basis, basis factors, etc.
  - Main work in each iteration: 2 (+1 for pricing norms) linear system solves
- Apply simplex pivots until no more negative reduced cost variables exist (optimality)



Simplex Algorithm – Pricing

- Dual variables

$$y = \left(B^{-1}\right)^T c_B$$

- Reduced costs

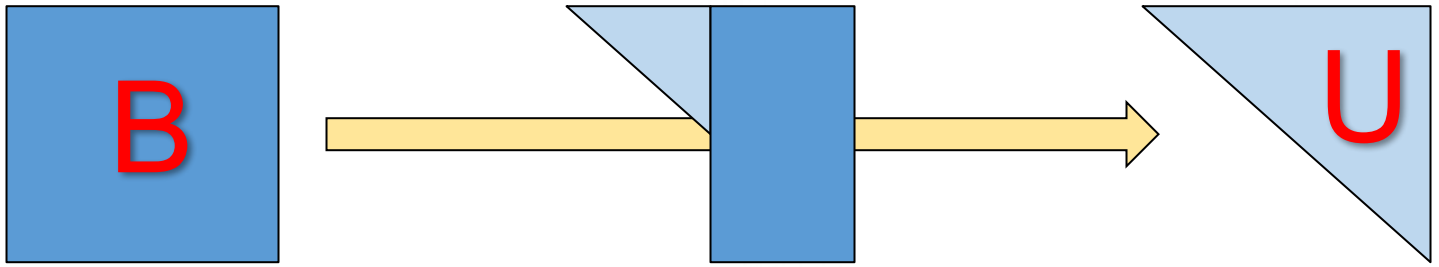
$$z = c - A^T y$$

- Reduced costs give pricing information
  - Change in objective per unit change in variable value
  - All reduced costs non-negative: proof of optimality
  - Multiple variables with negative reduced costs: pick one of them
    - Steepest edge pricing: geometrically sound interpretation of what a "unit change" in the variable value means

SimplexPricing, NormAdjust

Simplex Numerics – B = LU

- LU factorization of basis matrix
  - Gaussian elimination algorithm



pivot element

$3x_1 - 3x_2 + 1x_3 = 0$

$6x_1 + 2x_2 + 2x_3 = 8$

$1x_1 + 0.3333x_3 = 0$

$x_2 = 1$

$x_3 = 30000$

$x_1 = -9999$

elimination

$3x_1 - 3x_2 + 1x_3 = 0$

$8x_2 = 8$

fill $1x_2 - 0.0000333x_3 = 0$

cancellation

almost cancellation

Similar: Presolve Aggregator Numerics

pivot element

$$
\begin{aligned}
3x_1 - 3x_2 + 1x_3 &= 0 \\
6x_1 + 2x_2 + 2x_3 &= 8 \\
1x_1 + 0.3333x_3 &= 0
\end{aligned}
\quad \Rightarrow \quad x_1 := x_2 - 1/3x_3
$$

$$
\Rightarrow \quad
\begin{aligned}
8x_2 &= 8 \\
1x_2 - 0.0000333x_3 &= 0
\end{aligned}
\quad \Rightarrow \quad x_3 := 30000\, x_2
$$

Interior-Point Method

- Basic algorithm (Karmarkar, Fiacco & McCormick, Dikin):
  - Modify KKT conditions:

$$\begin{aligned} Ax &= b \\ A^T y + z &= c \\ x^T z &= 0 \end{aligned} \quad \Rightarrow \quad \begin{aligned} Ax &= b \\ A^T y + z &= c \\ X\, z &= \mu\, e \end{aligned} \qquad (X = \mathrm{diag}(x))$$

- Linearize complementarity condition

$$\begin{pmatrix} -\theta & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} dx \\ dy \end{pmatrix} = \begin{pmatrix} r2 \\ r1 \end{pmatrix} \qquad \text{(augmented system)}$$

$\theta_j = z_j / x_j$
$x_j \cdot z_j = 0$ at optimality, so $\theta_j \rightarrow 0$ or ∞

- Further simplification: $A\,\theta^{-1}\,A^T\,dy = b$ (normal equations)
- Iterate, reducing μ in each iteration
- Provable convergence

BarConvTol, BarQCPConvTol

Computational Steps

- Setup steps:
  - Presolve (same for simplex)
  - Compute fill-reducing ordering ← BarOrder
- In each iteration:
  - Form $A\,\theta^{-1}\,A^T$
  - Factor $A\,\theta^{-1}\,A^T = L\,D\,L^T$ (Cholesky factorization)
  - Solve $L\,D\,L^T\,x = b$
  - A few Ax and A^Tx computations
  - A bunch of vector operations
- Post-processing steps:
  - Perform crossover to a basic solution ← Crossover, CrossoverBasis
    - Optional step, but usually required for LP relaxations in a MIP solve

Barrier Log

```
Barrier statistics:
 AA' NZ     : 2.836e+03
 Factor NZ  : 3.551e+03 (roughly 40 MBytes of memory)
 Factor Ops : 1.739e+05 (less than 1 second per iteration)
 Threads    : 4

                 Objective                Residual
Iter       Primal          Dual         Primal    Dual     Compl     Time
   0   1.30273209e+06  0.00000000e+00  5.90e+02 0.00e+00  7.32e+00    12s
   1   1.04326180e+05 -5.84079103e+02  4.84e+01 1.69e+00  5.95e-01    12s
   2   9.46325157e+03 -4.40392705e+02  2.92e+00 1.35e+00  5.46e-02    12s
   3   3.66683689e+03  9.27381244e+02  1.94e-01 5.35e-01  1.41e-02    12s
   4   3.37449982e+03  1.79938013e+03  1.29e-01 2.41e-01  7.64e-03    12s
   5   3.13244138e+03  1.90266941e+03  8.89e-02 2.07e-01  6.00e-03    12s
   6   2.71282610e+03  2.11401255e+03  3.20e-02 1.15e-01  2.96e-03    12s
   7   2.48856811e+03  2.18107490e+03  1.06e-02 7.26e-02  1.56e-03    12s
   8   2.35427593e+03  2.21183615e+03  3.20e-03 4.52e-02  7.36e-04    12s
   9   2.30239737e+03  2.22464753e+03  1.53e-03 2.38e-02  4.03e-04    12s
  10   2.25547118e+03  2.23096162e+03  3.00e-04 1.40e-02  1.30e-04    12s
  11   2.24052450e+03  2.23917612e+03  4.10e-06 6.33e-04  7.20e-06    12s
  12   2.23967243e+03  2.23966346e+03  2.01e-08 5.01e-06  4.82e-08    12s
  13   2.23966667e+03  2.23966666e+03  1.11e-10 1.14e-13  4.81e-11    13s

Barrier solved model in 13 iterations and 12.51 seconds
Optimal objective 2.23966667e+03
```

Crossover Log

```
Barrier solved model in 13 iterations and 12.51 seconds
Optimal objective 2.23966667e+03


Root crossover log...

      40 DPushes remaining with DInf 0.0000000e+00                13s
       0 DPushes remaining with DInf 7.8159701e-14                13s

    1176 PPushes remaining with PInf 0.0000000e+00                13s
       0 PPushes remaining with PInf 0.0000000e+00                13s

  Push phase complete: Pinf 0.0000000e+00, Dinf 1.2079227e-13     13s


Root simplex log...

Iteration    Objective       Primal Inf.    Dual Inf.      Time
    1219    2.2396667e+03   0.000000e+00   0.000000e+00     13s
    1219    2.2396667e+03   0.000000e+00   0.000000e+00     13s

Root relaxation: objective 2.239667e+03, 1219 iterations, 0.43 seconds
```

Essential Differences

- Simplex:
  - Thousand/millions of iterations on extremely sparse matrices
  - Each iteration extremely cheap
  - Few opportunities to exploit parallelism
  - Can be warm-started
- Barrier:
  - Dozens of expensive iterations
  - Much denser matrices
  - Lots of opportunities to exploit parallelism
  - Barrier warm-start still an open research topic

Warm Start

- Warm start
  - Solve an LP and get optimal solution and basis
  - Change it slightly
  - Resolve it from previous solution or basis
- Simplex can warm start well
- Barrier cannot
- Warm start is crucial for MIP
- Typical change in MIP solve: add row or tighten bound
  - Both maintain dual feasibility of a previously optimal solution
- Consequence:
  - Dual simplex algorithm is the dominant algorithm to solve LP relaxations within a MIP solve

LP Performance

- Performance results:
  - Gurobi 6.0, quad-core Xeon E3-1240
  - Simplex on 1 core, barrier on 4 cores
  - Models that take >1s

	GeoMean
Dual simplex	2.50
Primal simplex	5.27
Barrier	1.28
Concurrent	1.00
Det. concurrent	1.10

Concurrent Log

```
Root barrier log...

Ordering time: 0.22s

Barrier statistics:
 AA' NZ     : 2.149e+07
 Factor NZ  : 1.309e+08 (roughly 1.1 GBytes of memory)
 Factor Ops : 3.332e+11 (roughly 6 seconds per iteration)
 Threads    : 2

Barrier performed 0 iterations in 24.01 seconds
Barrier solve interrupted - model solved by another algorithm

Concurrent spin time: 0.26s

Root simplex log...

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    3.6591691e+09   0.000000e+00   0.000000e+00     24s

Solved with dual simplex
Solved in 11056 iterations and 24.17 seconds
Optimal objective  3.659169147e+09

Root relaxation: objective 3.659169e+09, 11056 iterations, 5.60 seconds
```

Gurobi MIP Algorithms

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

Presolve, PrePasses, AggFill, Aggregate, DualReductions, PreSparsify, ...

Method, NodeMethod

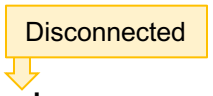
Cuts, CutPasses, CutAggPasses, GomoryPasses, CliqueCuts, CoverCuts, FlowCoverCuts, ...

VarBranch

Heuristics, MinRelNodes, PumpPasses, RINS, SubMIPNodes, ZeroObjNodes

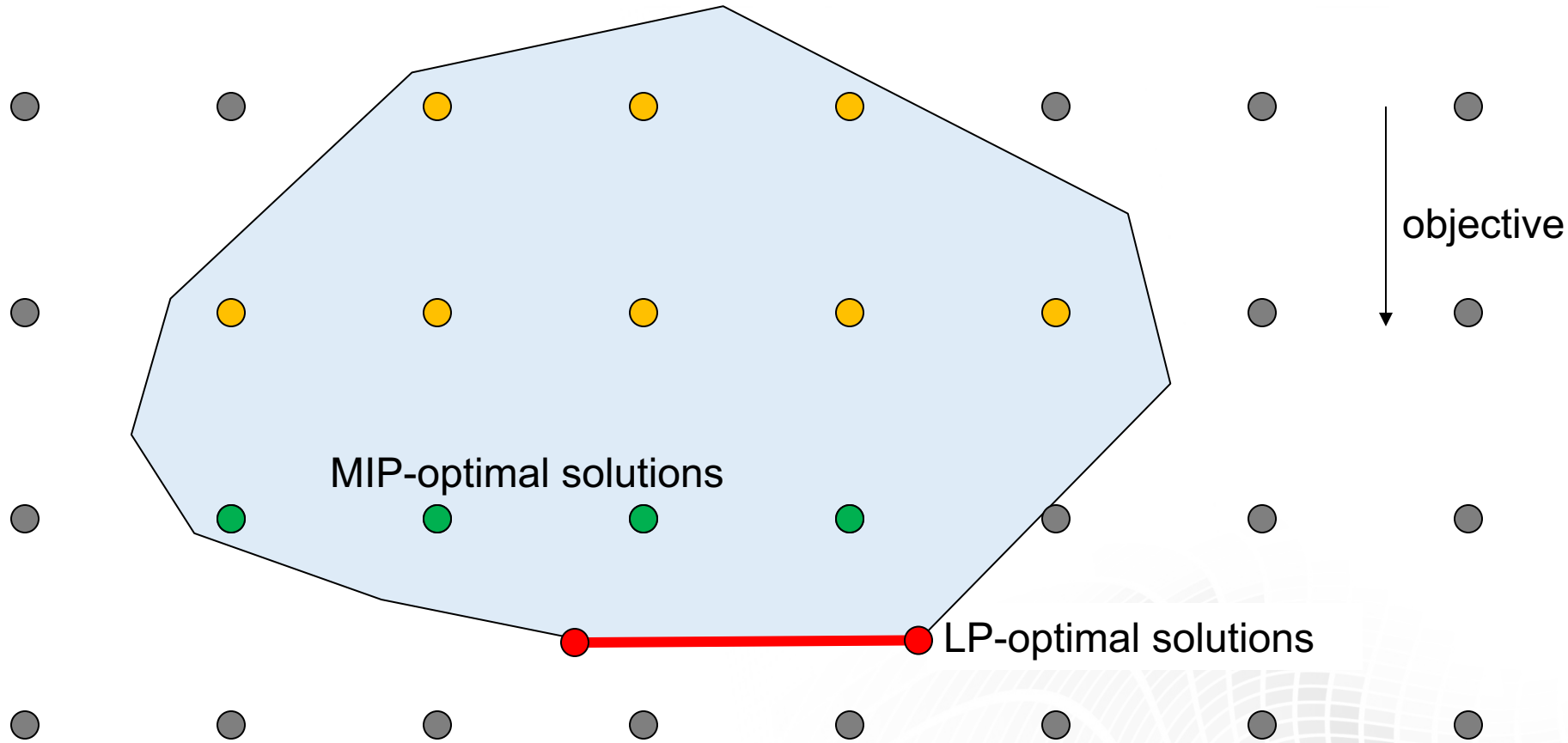
MIP Presolve

- Goals:
  - Reduce problem size
  - Strengthen LP relaxation
  - Identify problem sub-structures
    - Cliques, implied bounds, networks, disconnected components, ...

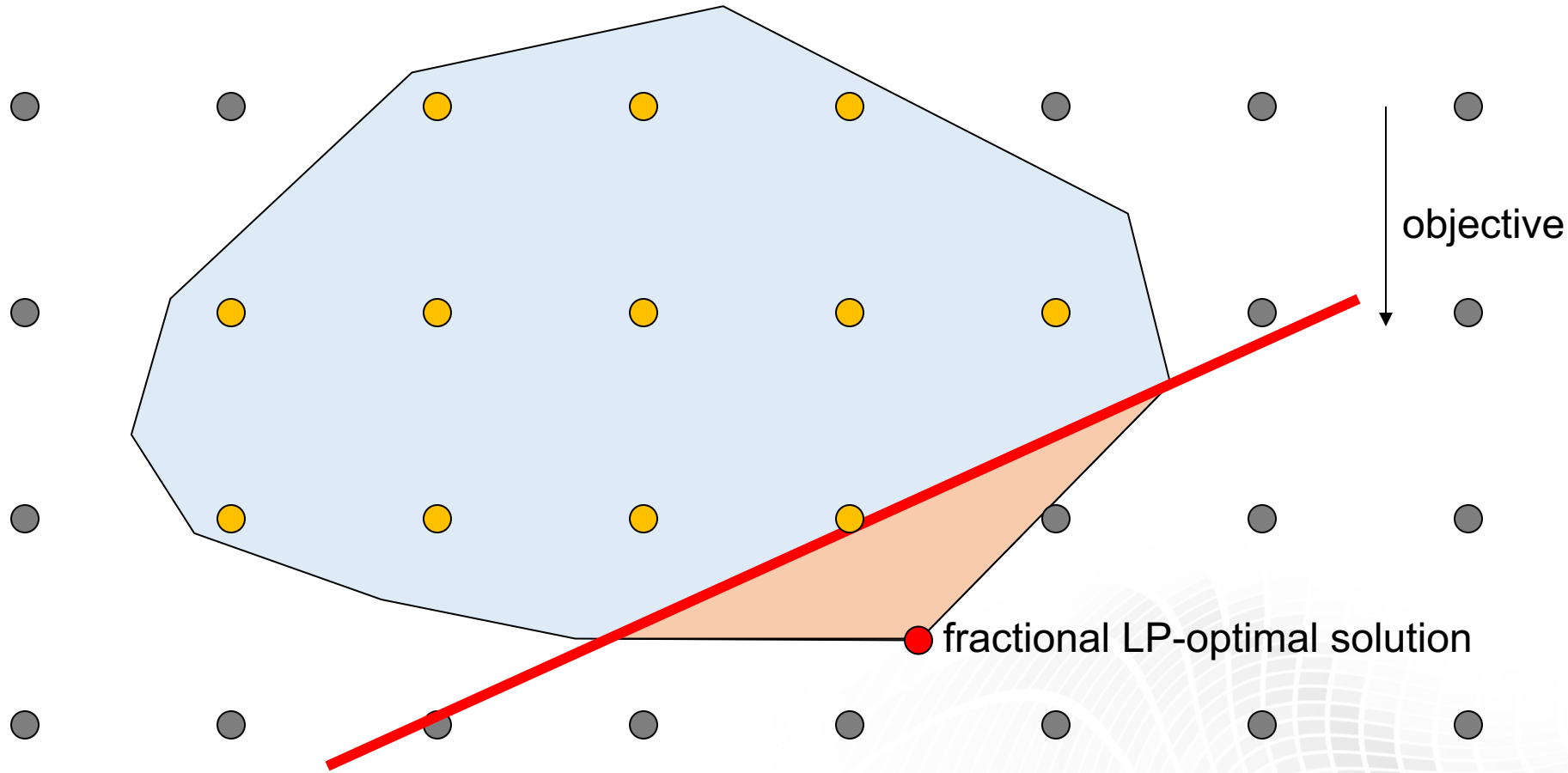


- Similar to LP presolve, but more powerful:
  - Exploit integrality
    - Round fractional bounds and right hand sides
    - Lifting/coefficient strengthening
    - Probing
  - Does not need to preserve duality
    - We only need to be able to uncrush a primal solution
    - Neither a dual solution nor a basis needs to be uncrushed

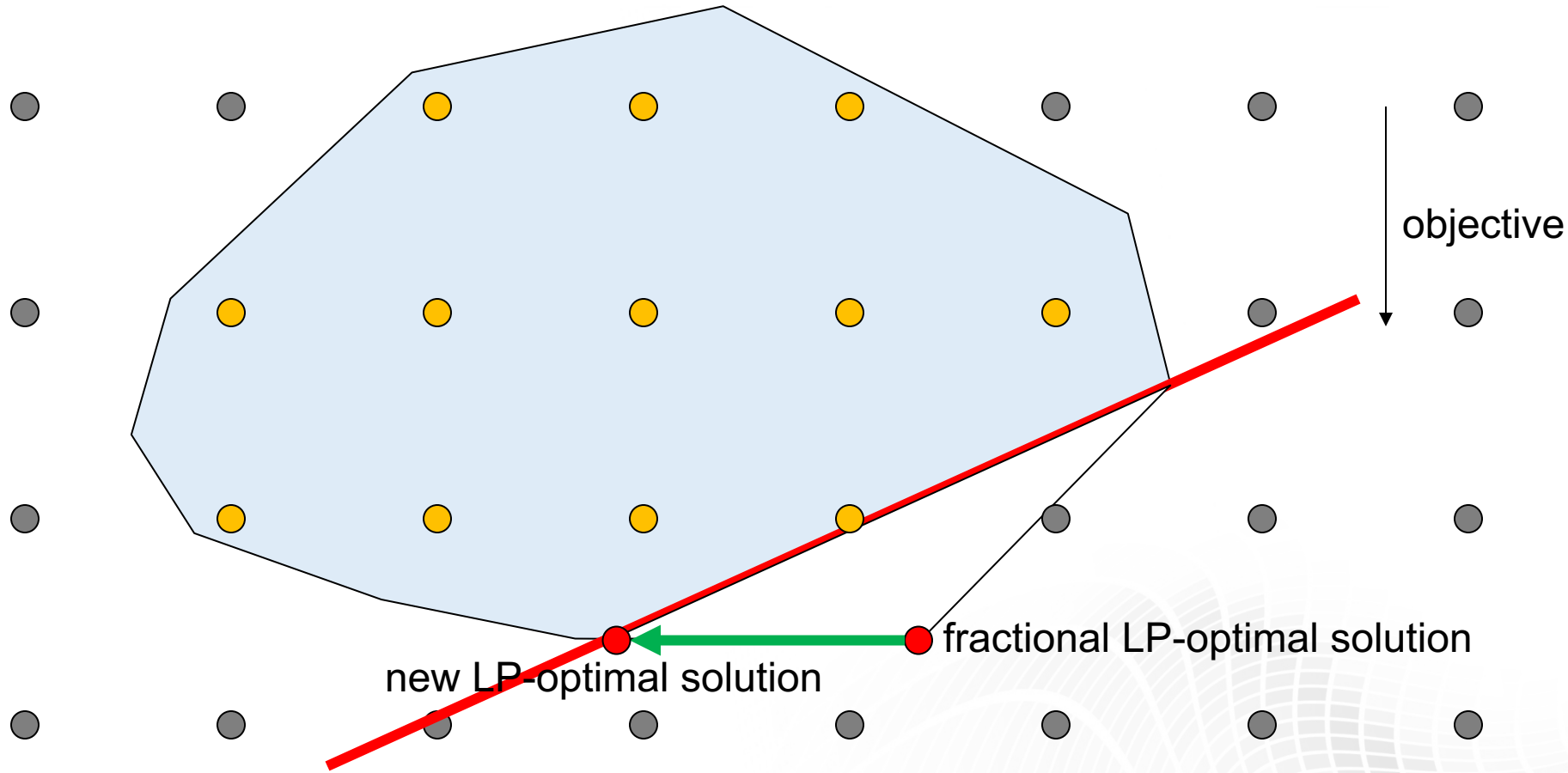
MIP – LP Relaxation



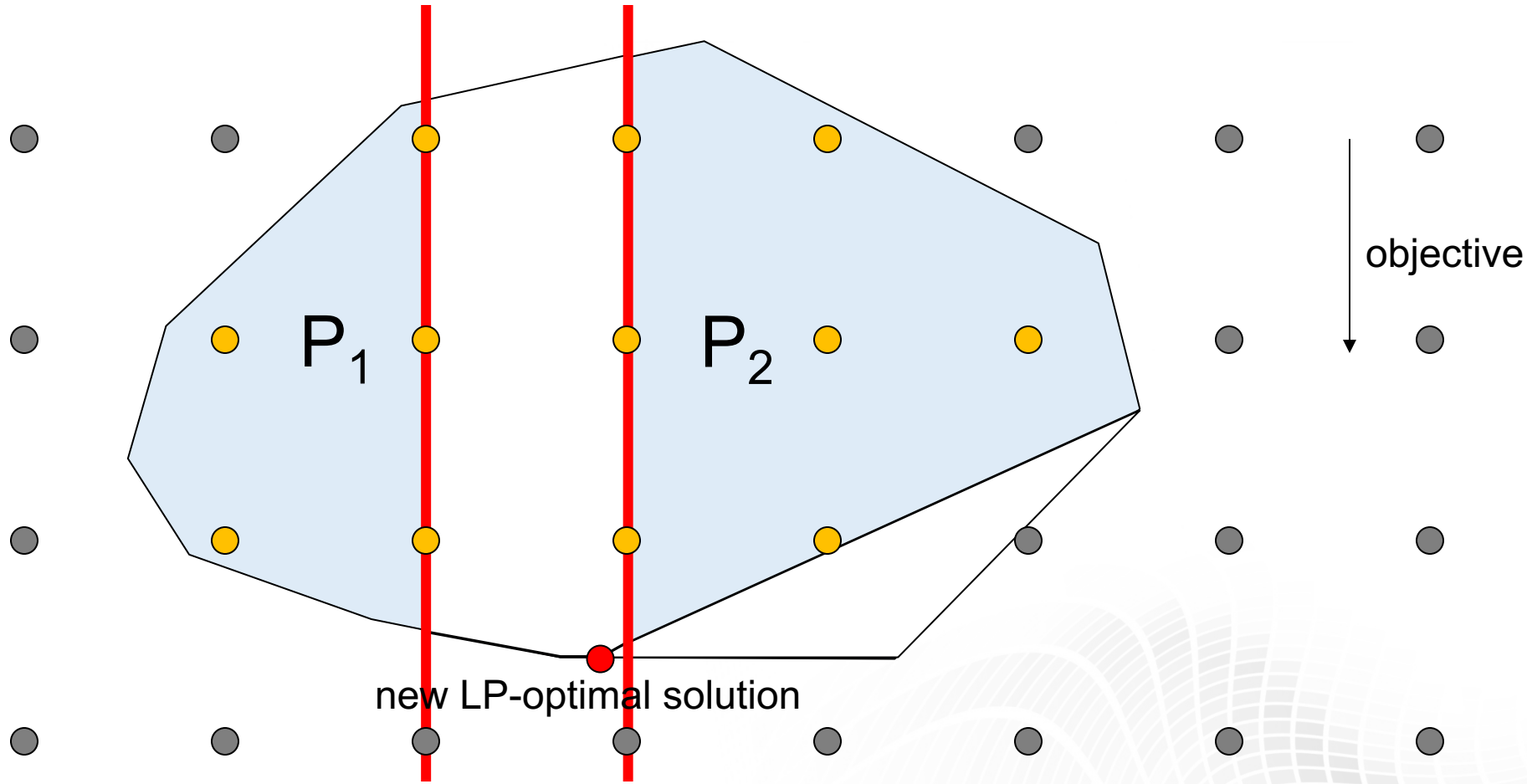
MIP – Cutting Planes



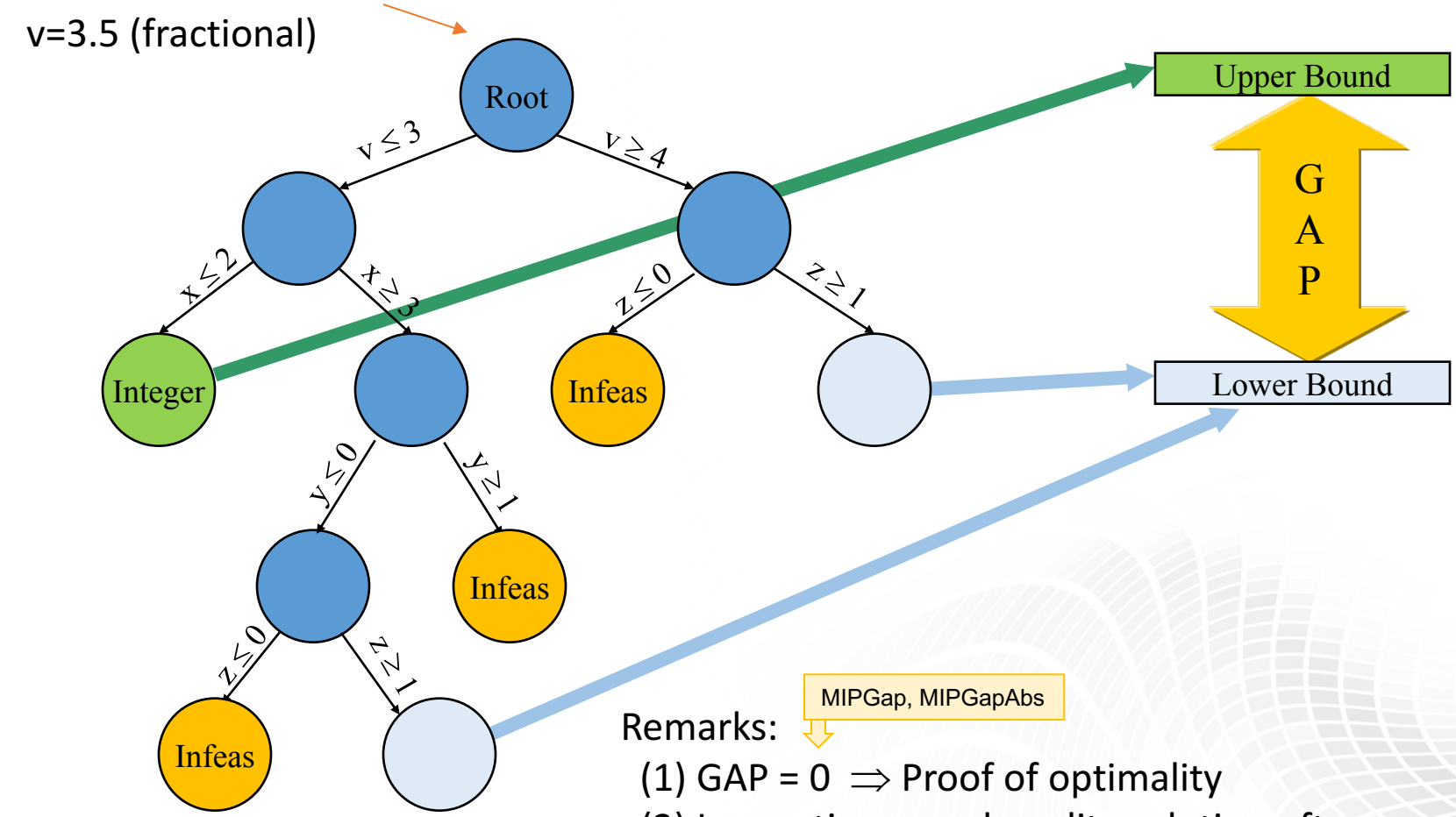
MIP – Cutting Planes



MIP – Branching



LP based Branch-and-Bound



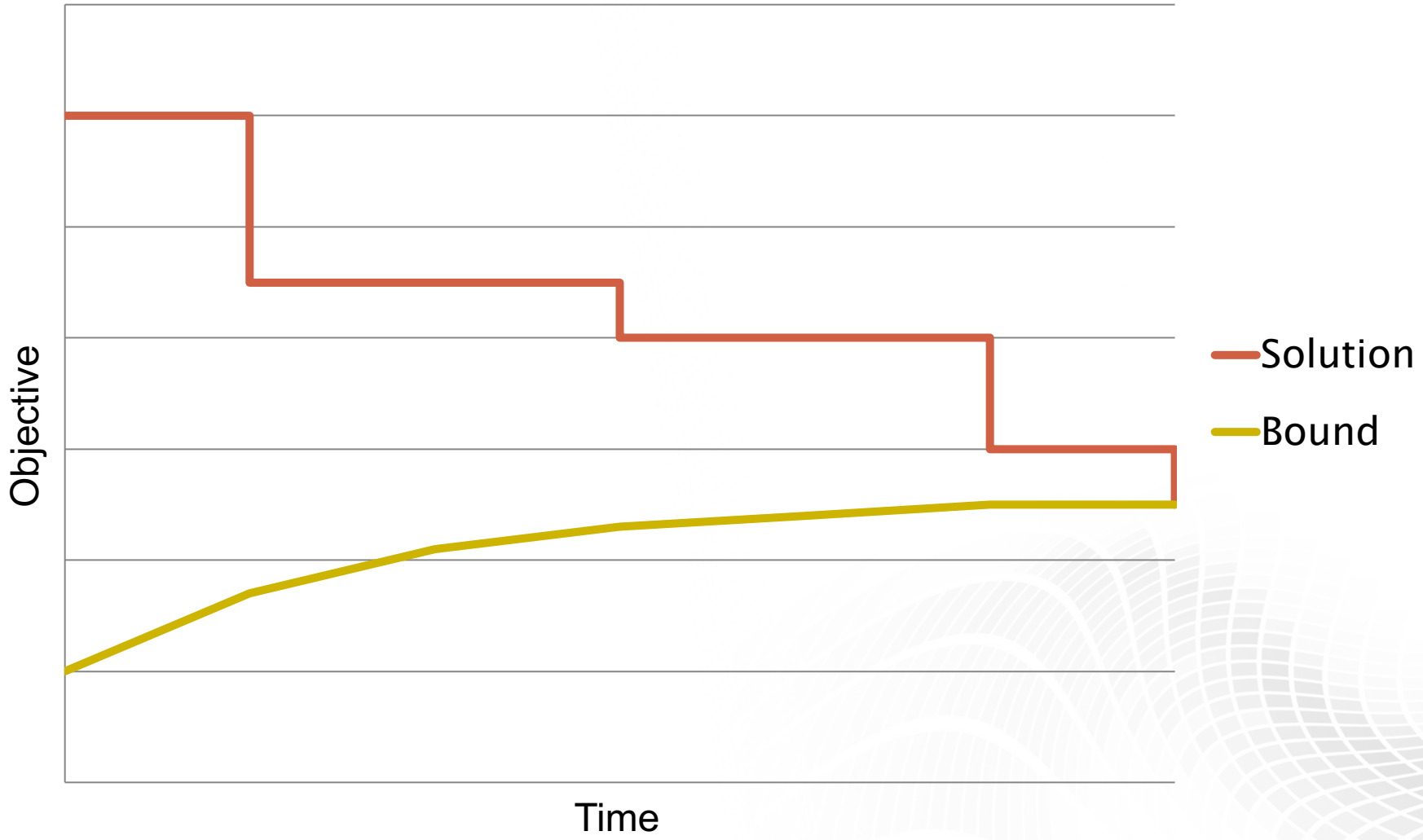
Remarks:

MIPGap, MIPGapAbs

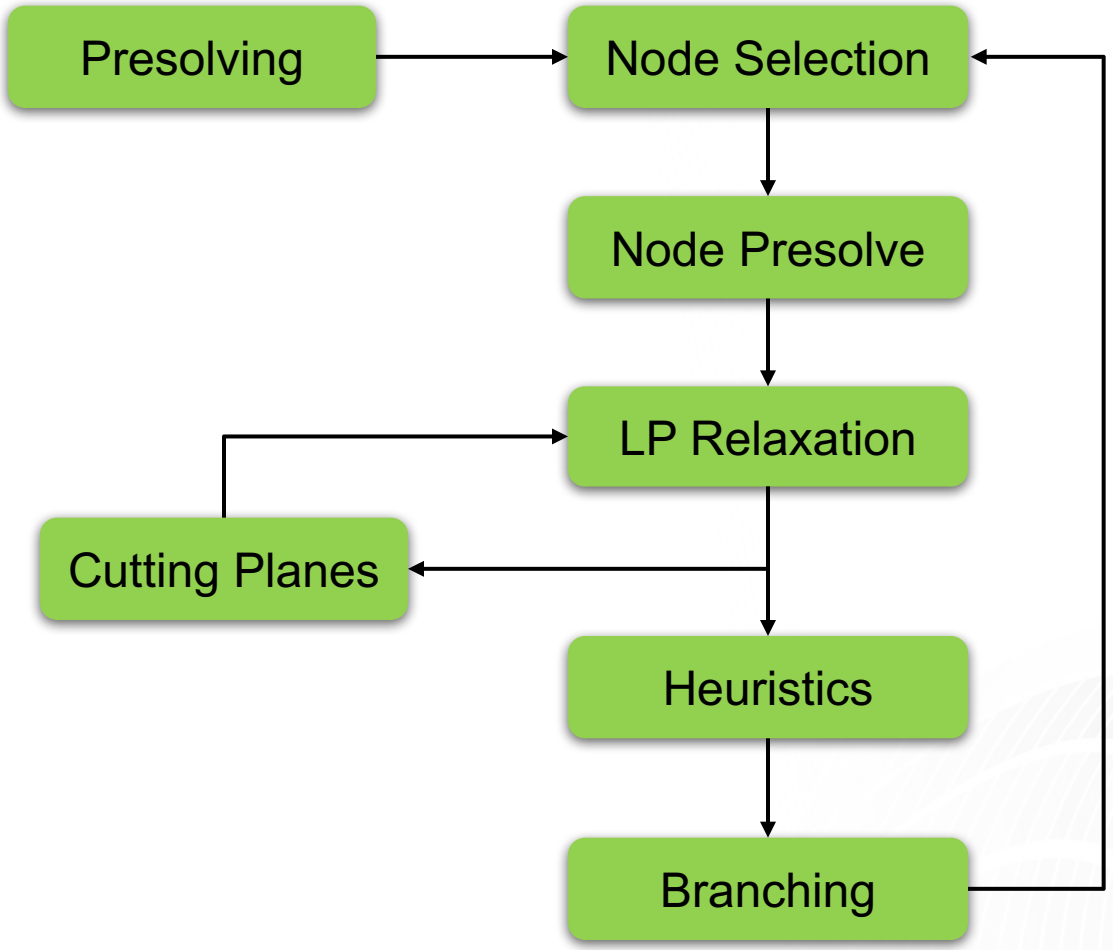
(1) GAP = 0 $\Rightarrow$ Proof of optimality

(2) In practice: good quality solution often enough

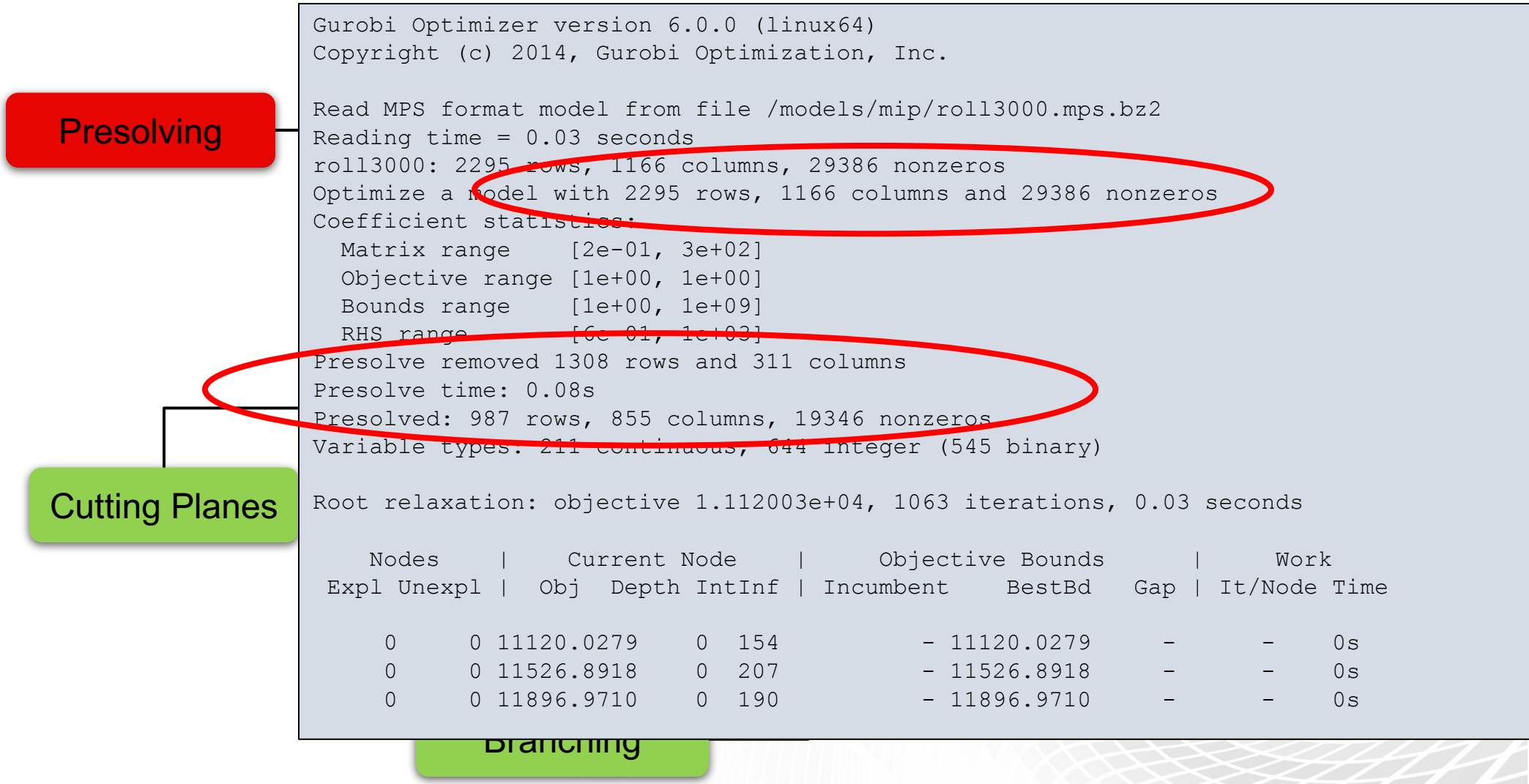
Solving a MIP Model



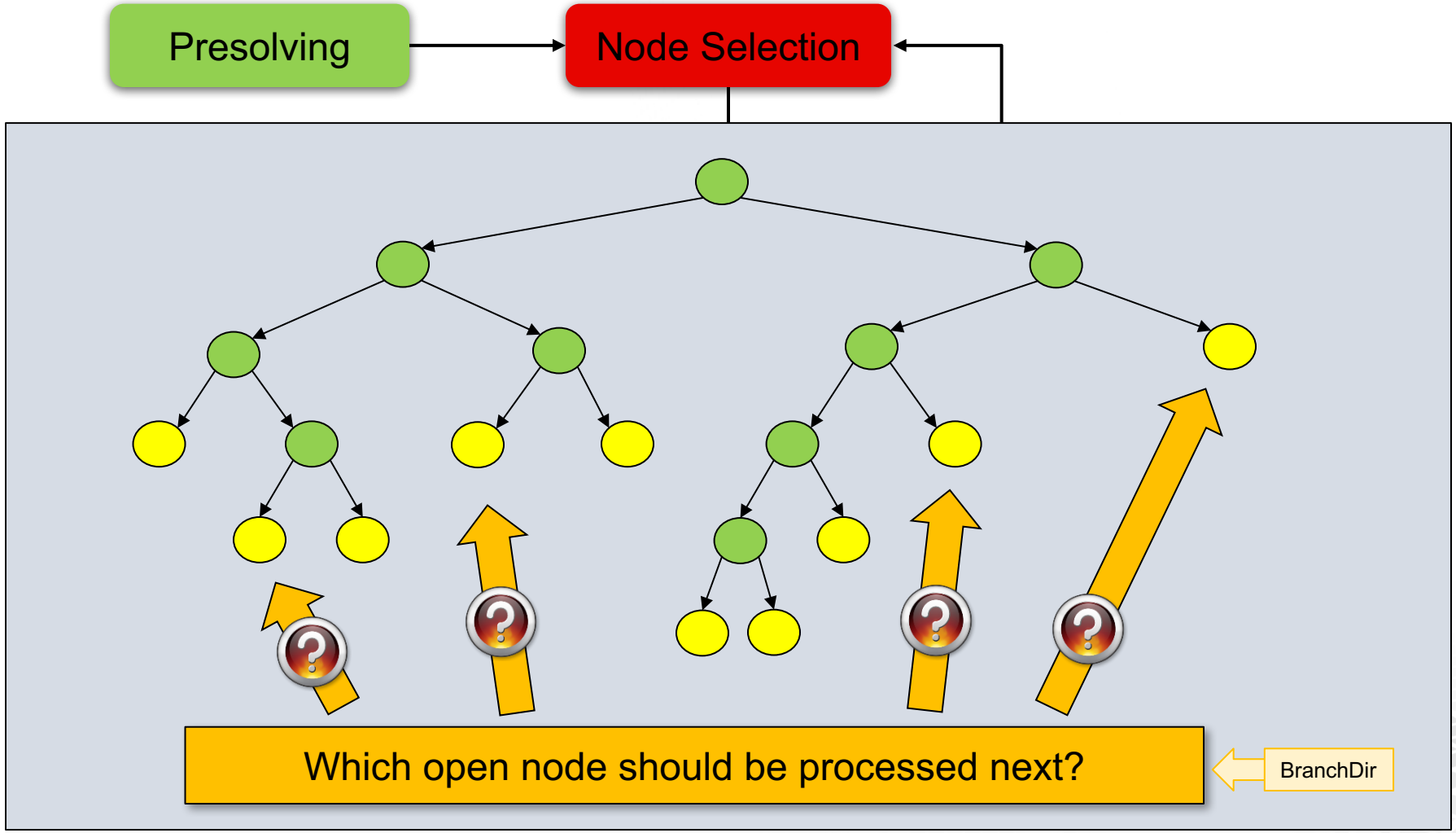
Branch-and-Cut



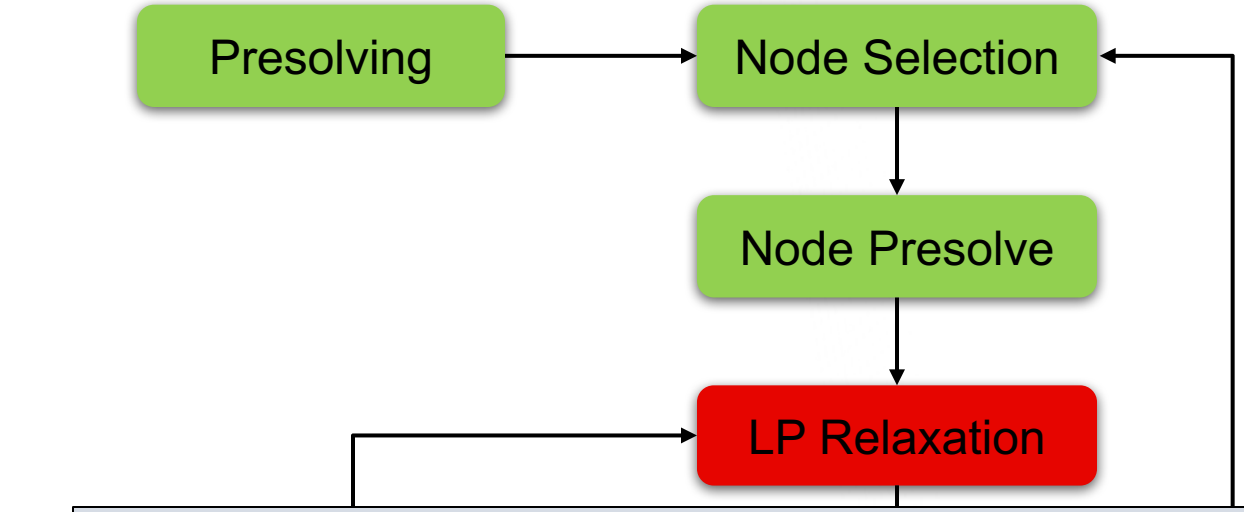
Branch-and-Cut



Branch-and-Cut



Branch-and-Cut



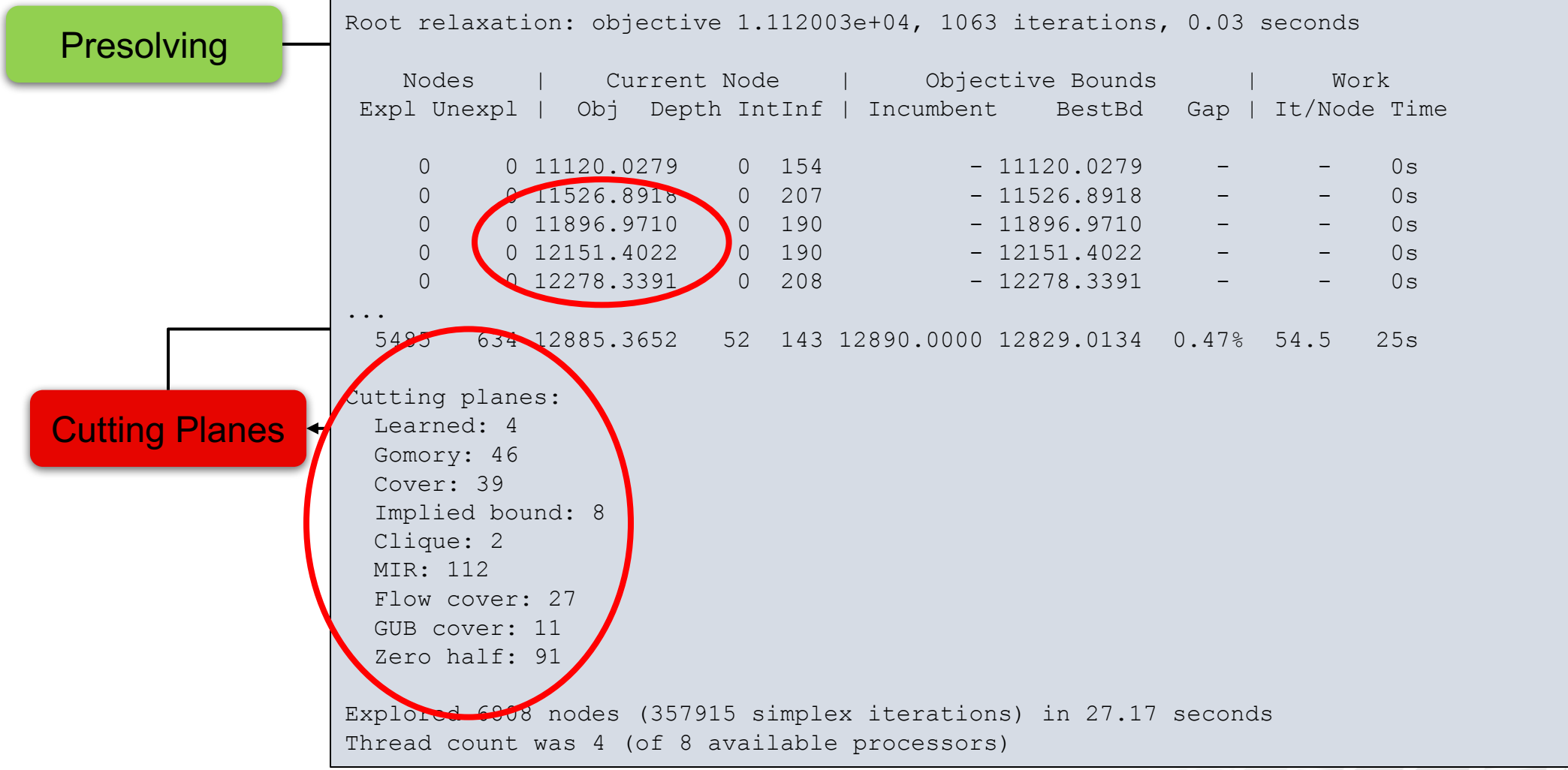

```
Presolved: 987 rows, 855 columns, 19346 nonzeros
Variable types: 211 continuous, 644 integer (545 binary)

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%   37.2    5s
```

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
...
     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
```

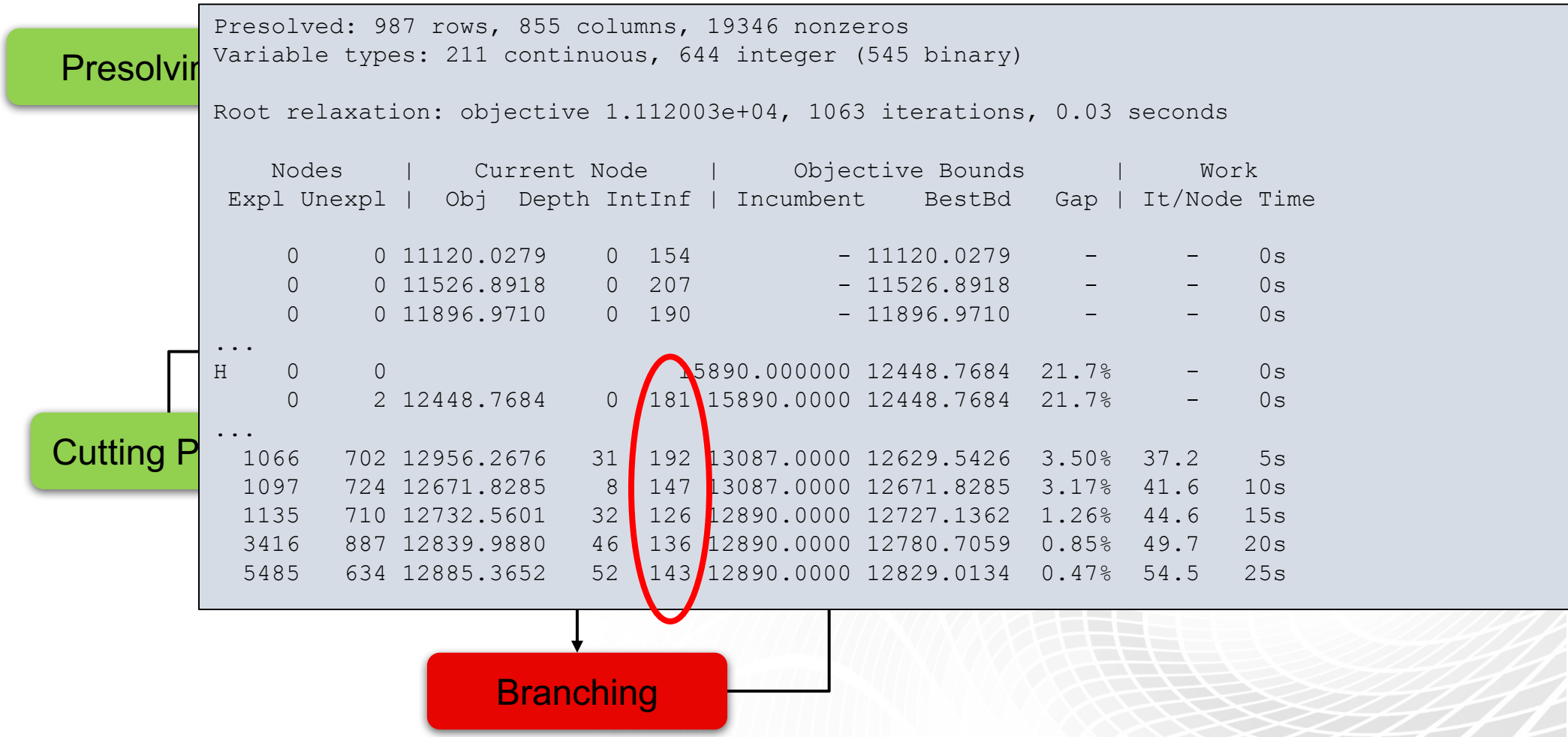
Presolving

Cutting Pla

Heuristics

Branching

Branch-and-Cut



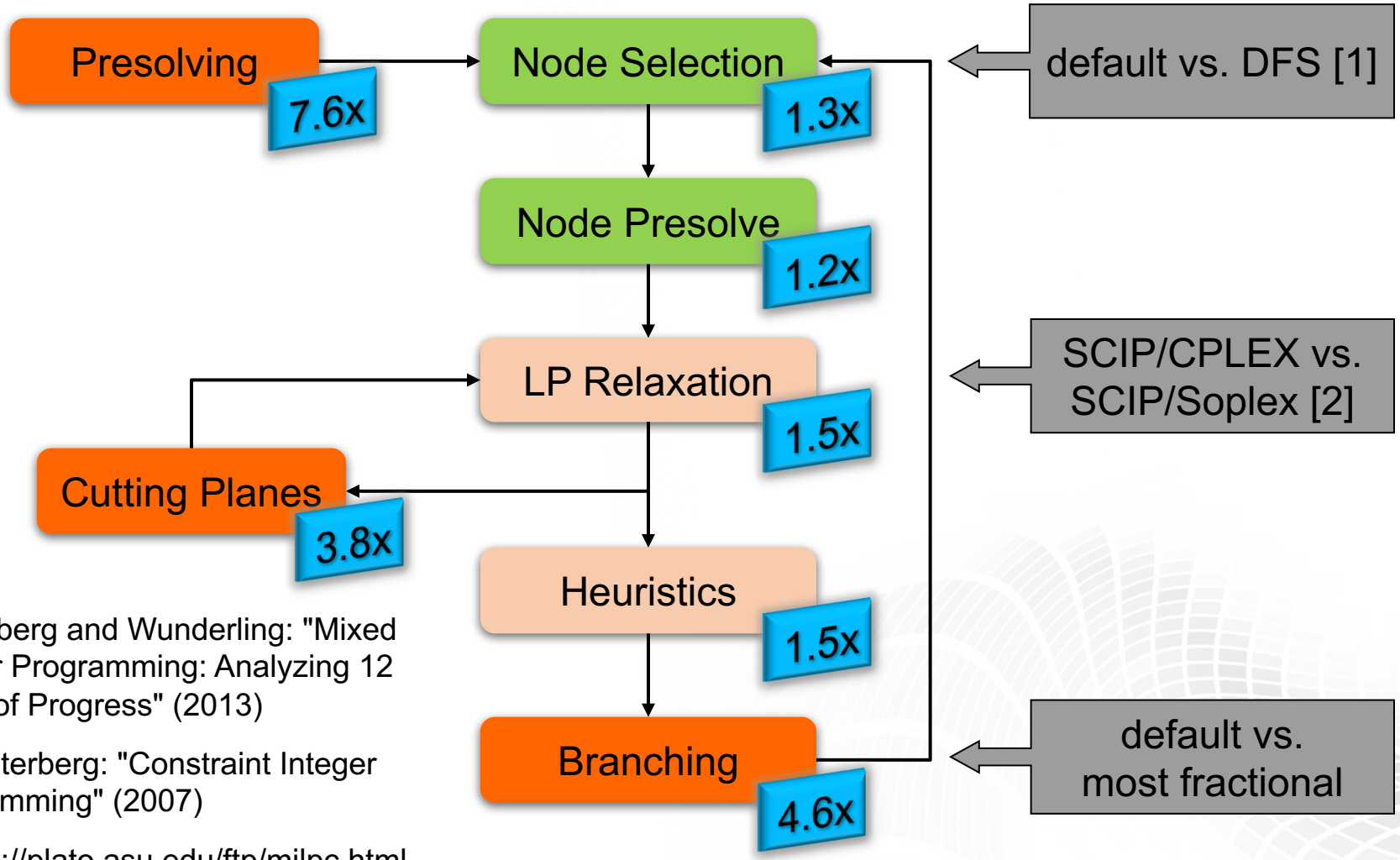

```
Presolved: 987 rows, 855 columns, 19346 nonzeros
Variable types: 211 continuous, 644 integer (545 binary)

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    0     0                       15890.000000 12448.7684  21.7%      -    0s
     0     2 12448.7684    0  181 15890.0000 12448.7684  21.7%      -    0s
...
  1066   702 12956.2676   31  192 13087.0000 12629.5426  3.50%  37.2    5s
  1097   724 12671.8285    8  147 13087.0000 12671.8285  3.17%  41.6   10s
  1135   710 12732.5601   32  126 12890.0000 12727.1362  1.26%  44.6   15s
  3416   887 12839.9880   46  136 12890.0000 12780.7059  0.85%  49.7   20s
  5485   634 12885.3652   52  143 12890.0000 12829.0134  0.47%  54.5   25s
```

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

Thank You