Algorithms in Gurobi

GUROBI
OPTIMIZATION

What's Inside Gurobi Optimizer

- 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

- Presole
- Primal & dual simplex method
  - Numerically stable (most challenging part)
- Parallel barrier method with crossover
  - Can effectively exploit multiple cores
- Concurrent optimization ← ConcurrentSettings
  - 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

Presolve

- Goal
  - Reduce the problem size
- 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 $x + z$

Primal and Dual LP

- Primal Linear Program:

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

- 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:

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

Strong Duality Theorem:

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

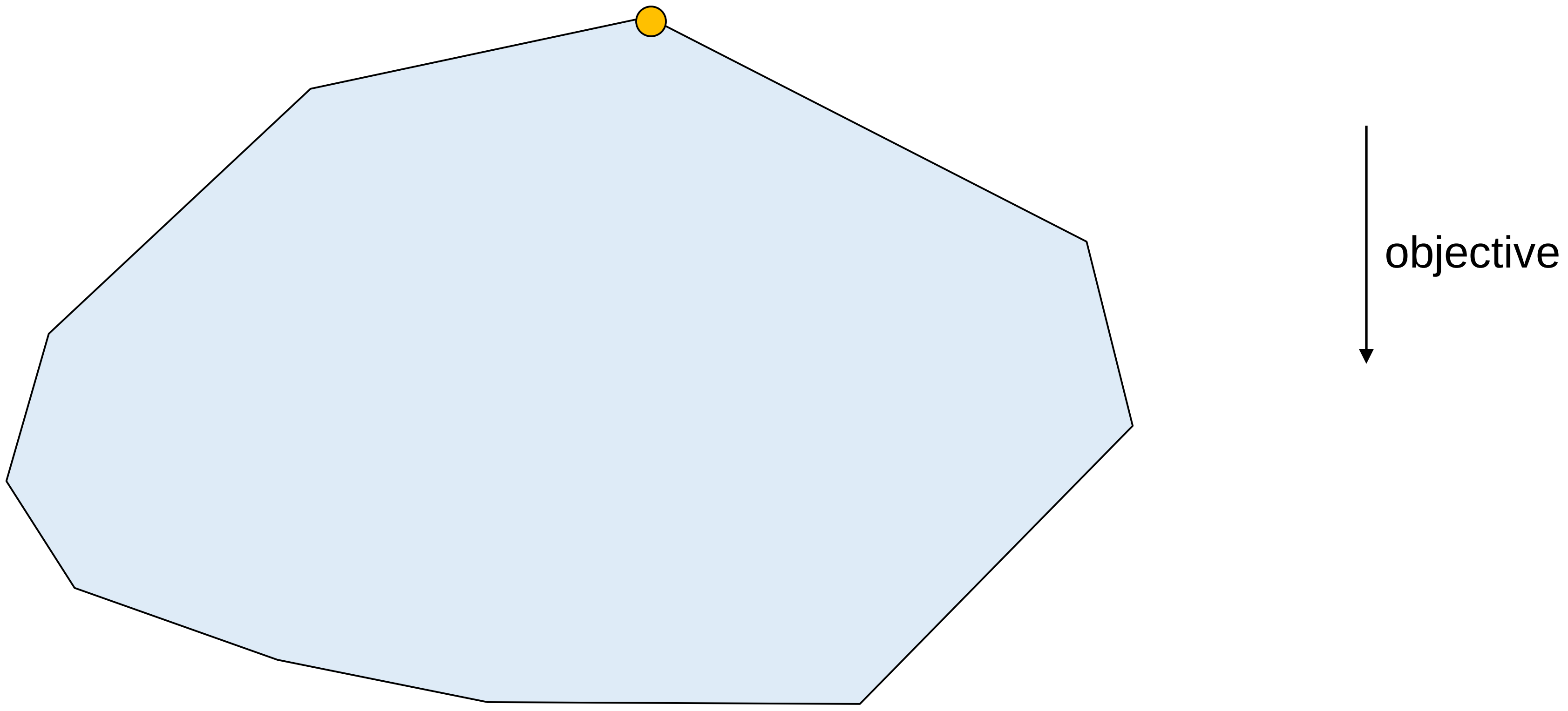
(if primal and dual are both feasible)

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: $x^T z = 0$

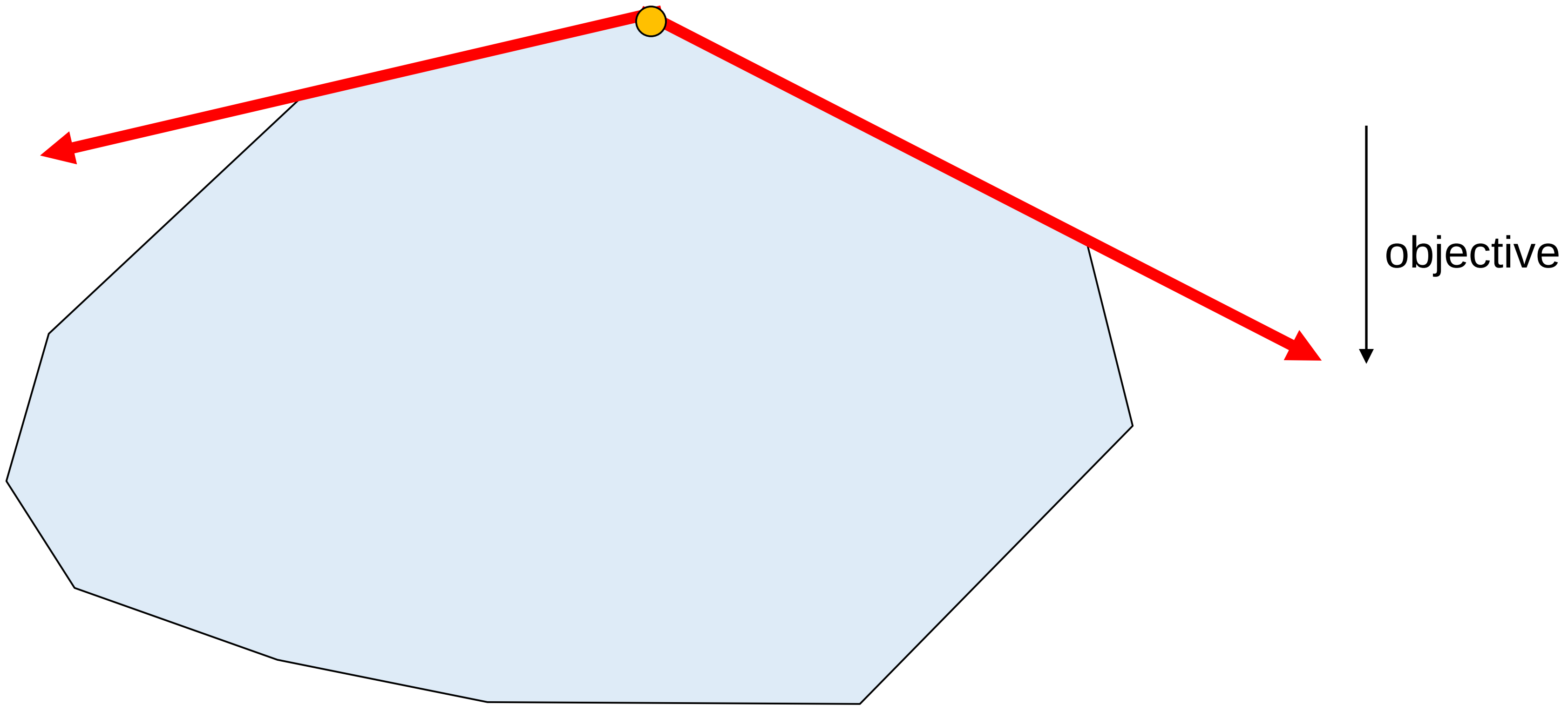
	Primal feas	Dual feas	Complementarity
Primal simplex	Maintain	Goal	Maintain
Dual simplex	Goal	Maintain	Maintain
Barrier	Goal	Goal	Goal

Simplex Algorithm



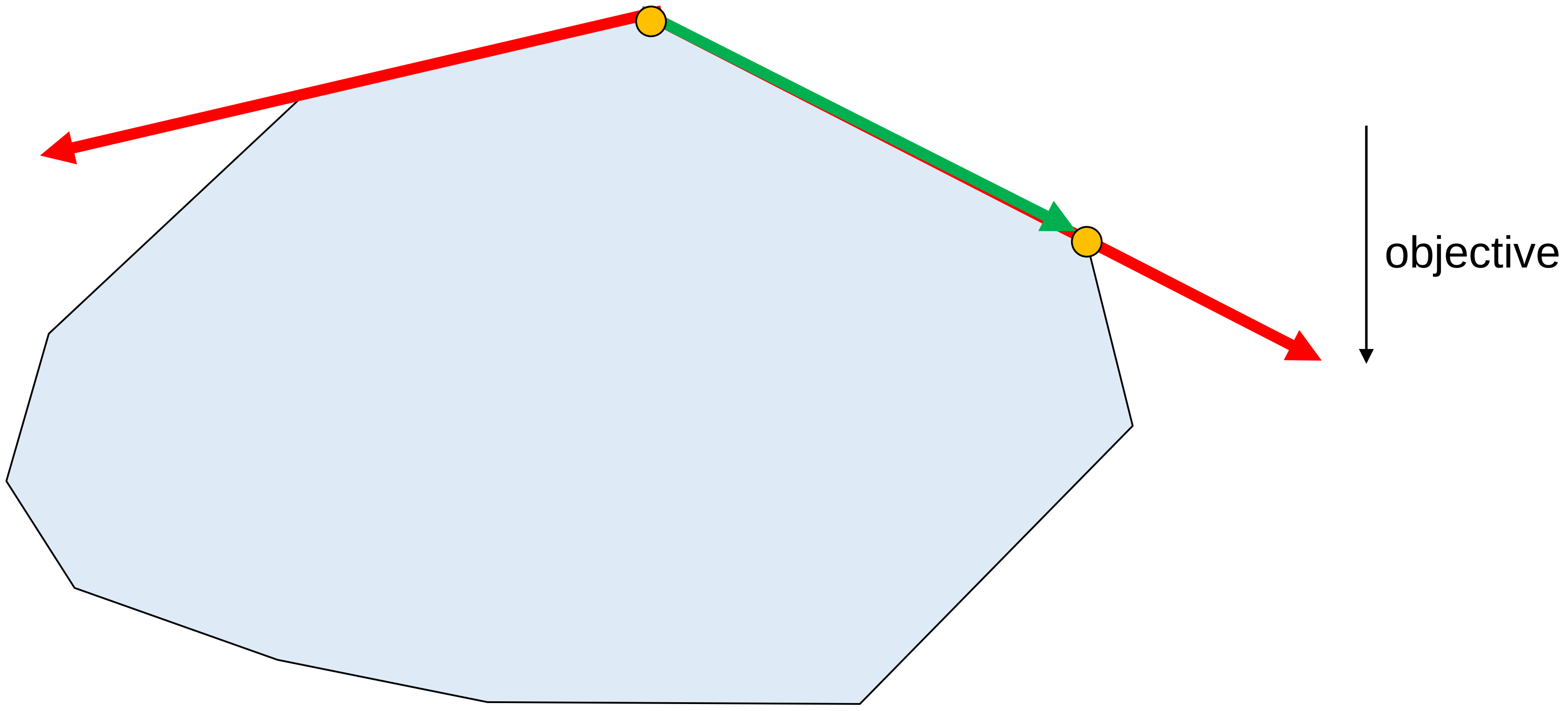
- **Phase 1**: find some feasible vertex solution

Simplex Algorithm



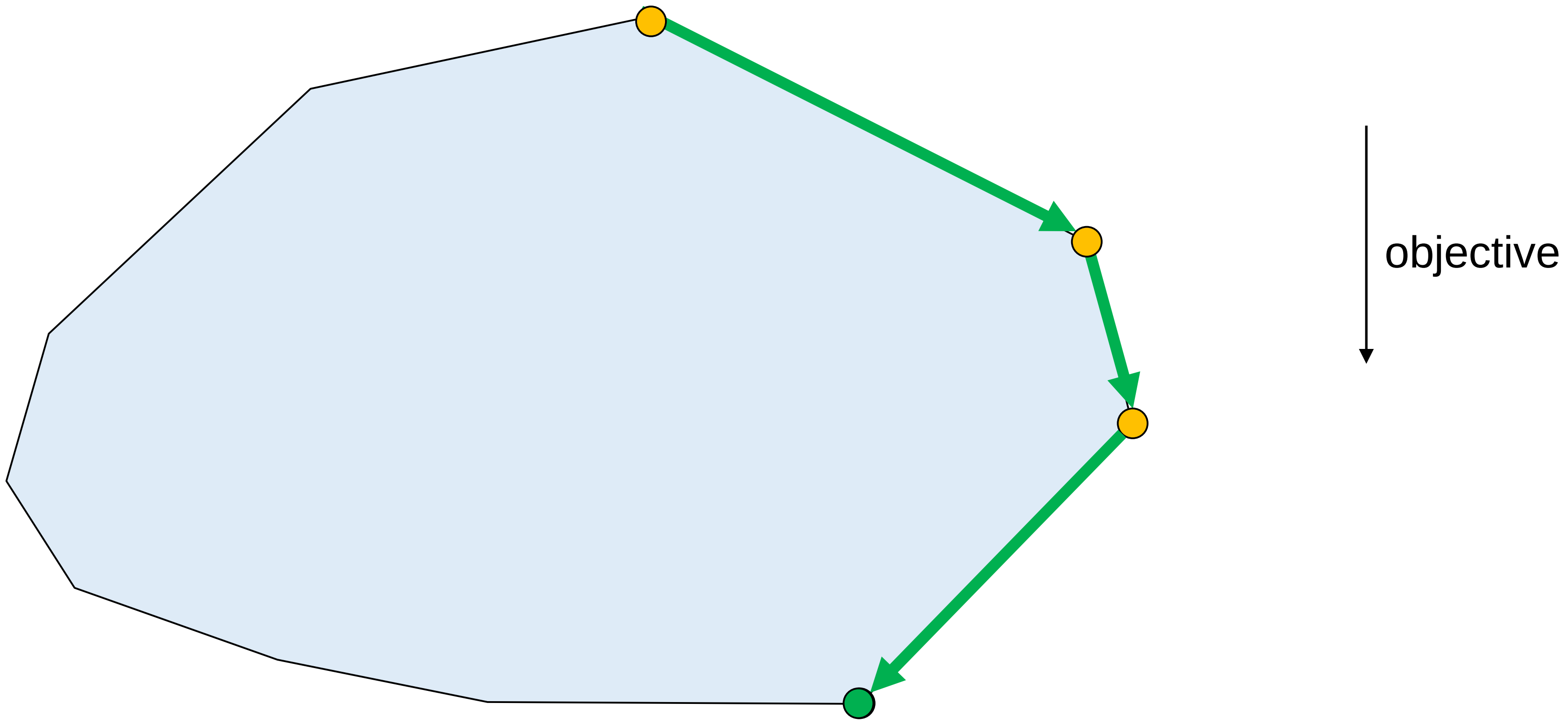
- **Pricing**: find directions in which objective improves and select one of them

Simplex Algorithm



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

Simplex Algorithm



- Iterate until no more improving direction is found

Simplex 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

$$Bx_B + Nx_N = b$$

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

- Solve for basic variables

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

- Solved by maintaining

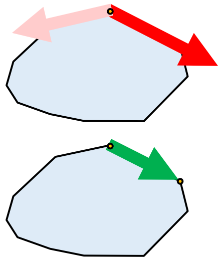
$$B = LU$$

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

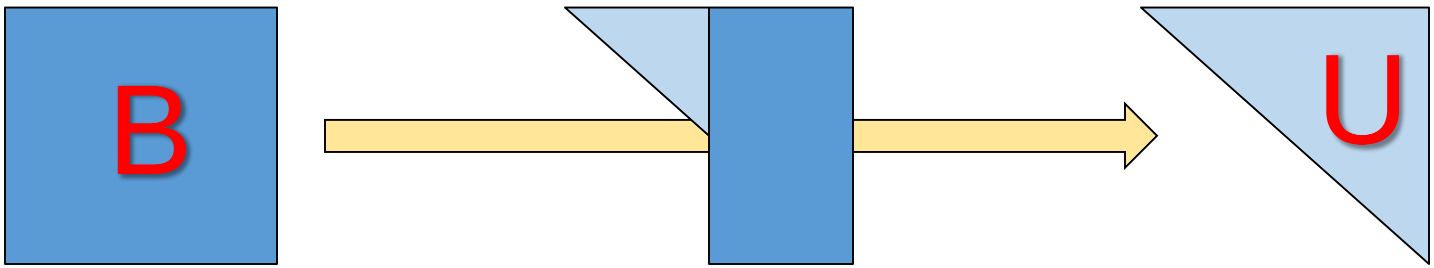
$$z = c - A^T y$$

- 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
  - Gauss elimination algorithm



pivot element

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

$$\begin{aligned} 3x_1 &- 3x_2 &+ 1x_3 &= 0 \\ & 8x_2 & &= 8 \\ & 1x_2 &- 0.0000333x_3 &= 0 \end{aligned}$$

elimination

fill

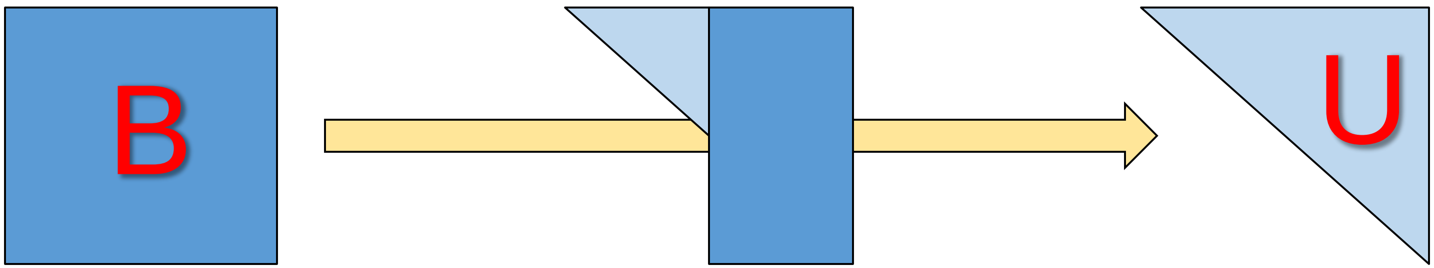
cancellation

almost cancellation

GUROBI OPTIMIZATION

Simplex Numerics – B = LU

- LU factorization of basis matrix
  - Gauss elimination algorithm



pivot element

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

elimination

fill

$$
\begin{aligned}
3x_1 &- 3x_2 &+ 1x_3 &= 0\\
 & 8x_2 & &= 8\\
 & 1x_2 &- 0.0000333x_3 &= 0
\end{aligned}
$$

$$
\begin{aligned}
x_2 &= 1\\
x_3 &= 30000\\
x_1 &= -9999
\end{aligned}
$$

GUROBI OPTIMIZATION

Presolve Aggregator Numerics

pivot element

$$3x_1 - 3x_2 + 1x_3 = 0 \quad \Rightarrow \quad x_1 := x_2 - 1/3x_3$$

$$6x_1 + 2x_2 + 2x_3 \leq 8$$

$$1x_1 + 0.3333x_3 = 0$$

$$\Rightarrow$$

$$8x_2 \leq 8$$

$$1x_2 - 0.0000333x_3 = 0 \quad \Rightarrow \quad x_3 := 30000\, x_2$$

Interior-Point Method

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

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

- 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 $\theta_j \to 0$ or ∞

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

GUROBI OPTIMIZATION

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

GUROBI OPTIMIZATION

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
  - How to warm-start barrier is still an unsolved research topic

LP Performance

- Performance results:
  - Gurobi 6.0, quad-core Xeon E3-1240
  - Dual 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

Gurobi MIP Algorithms

MIP Building Blocks

- Presolve — Presolve, PrePasses, AggFill, Aggregate, DualReductions, PreSparsify, ImproveStartTime, ...
  - Tighten formulation and reduce problem size
- Solve continuous relaxations — Method, NodeMethod
  - Ignoring integrality
  - Gives a bound on the optimal integral objective
- Cutting planes — Cuts, CutPasses, CutAggPasses, GomoryPasses, CliqueCuts, CoverCuts, FlowCoverCuts, ...
  - Cut off relaxation solutions
- Branching variable selection — VarBranch
  - Crucial for limiting search tree size
- Primal heuristics — Heuristics, MinRelNodes, PumpPasses, RINS, SubMIPNodes, ZeroObjNodes
  - Find integer feasible solutions

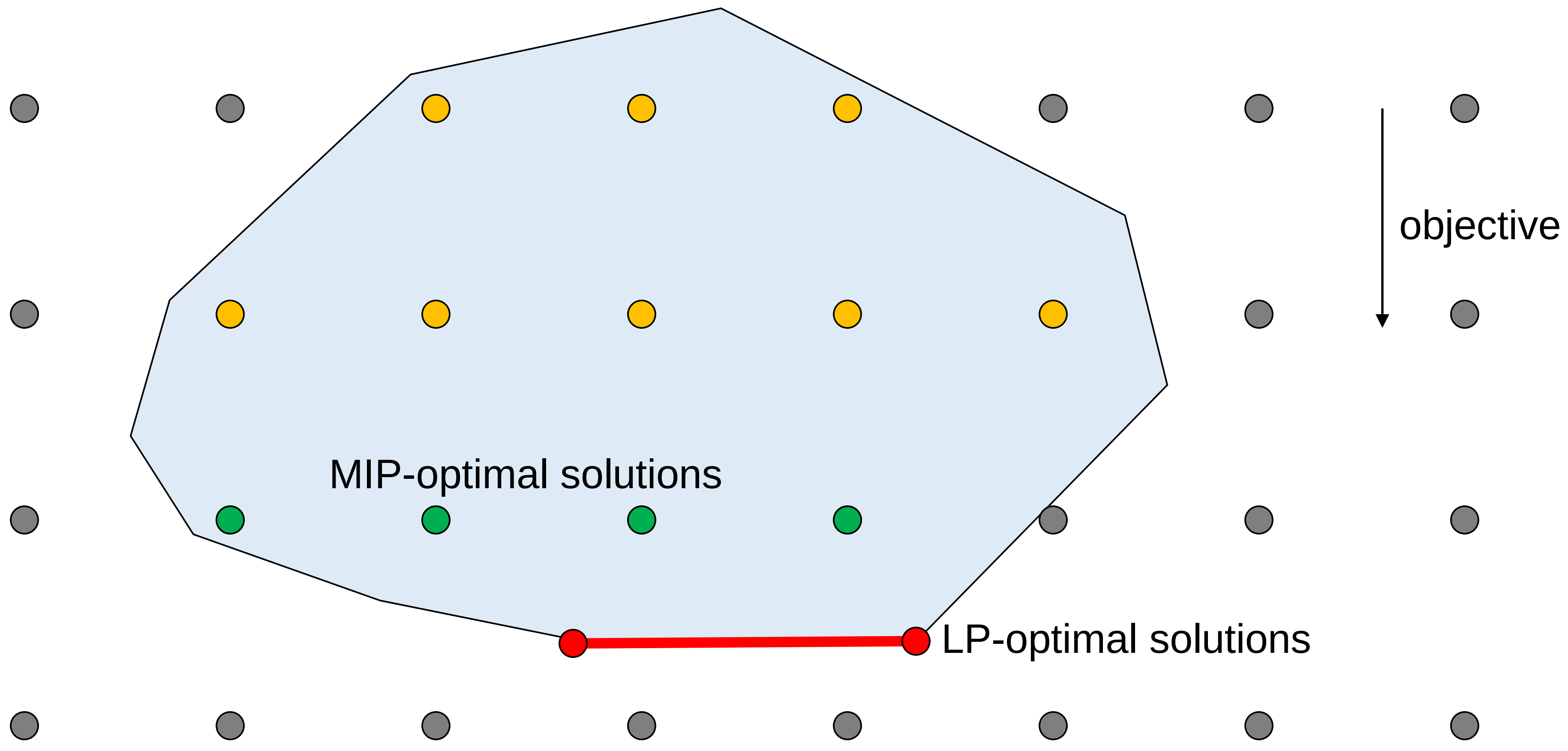
MIP Presolve

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

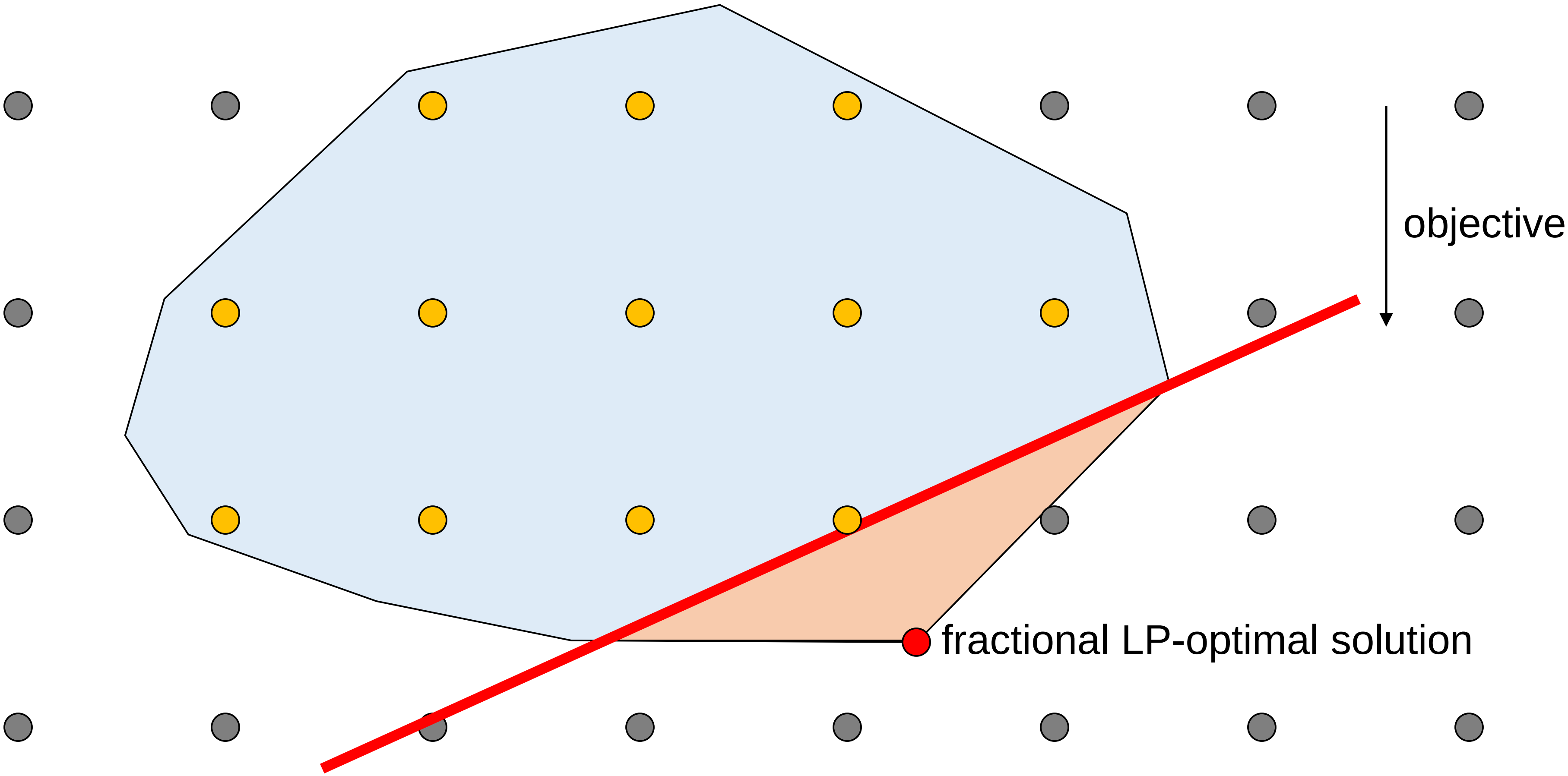
Disconnected

- 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

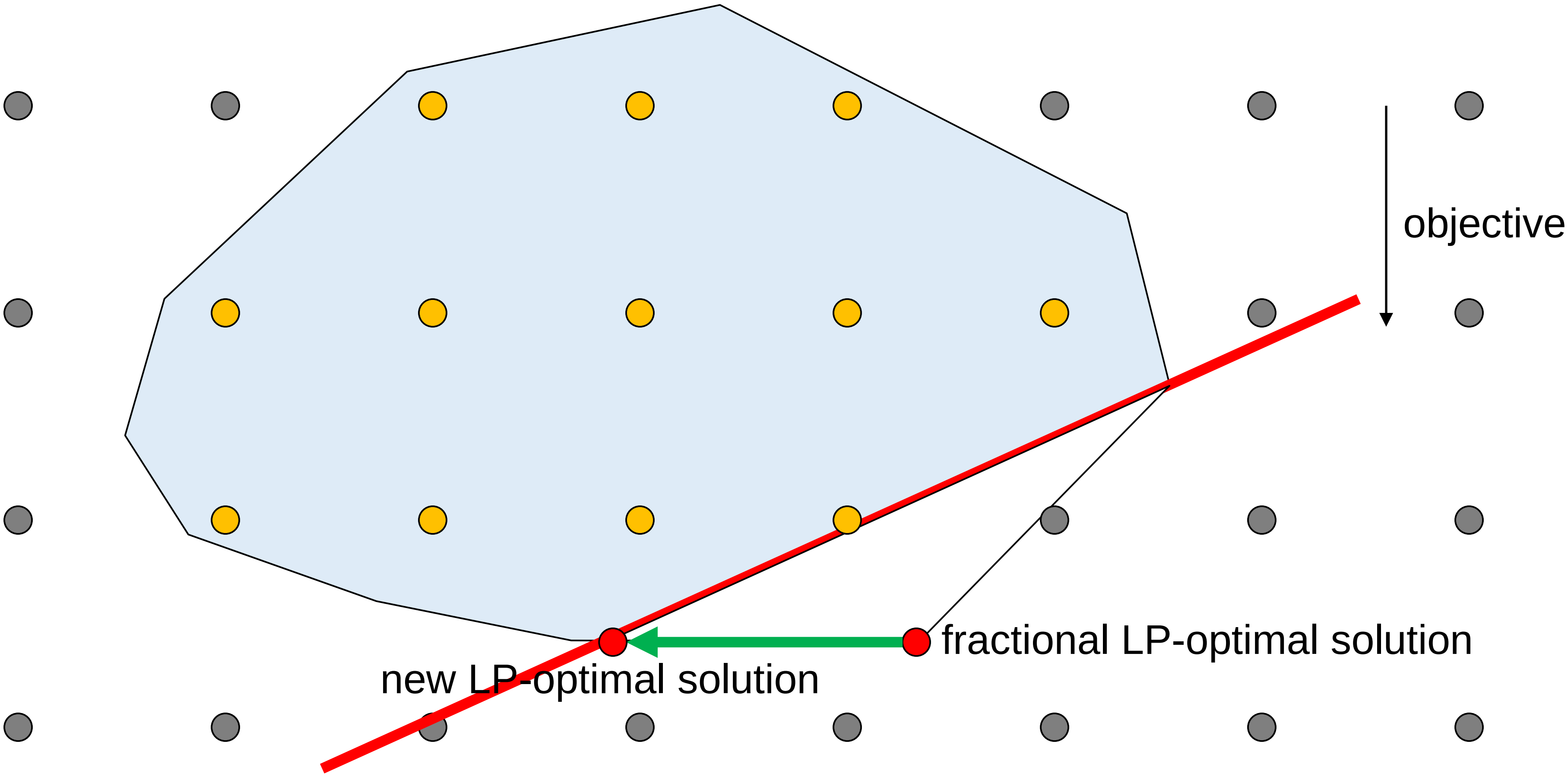
MIP – LP Relaxation



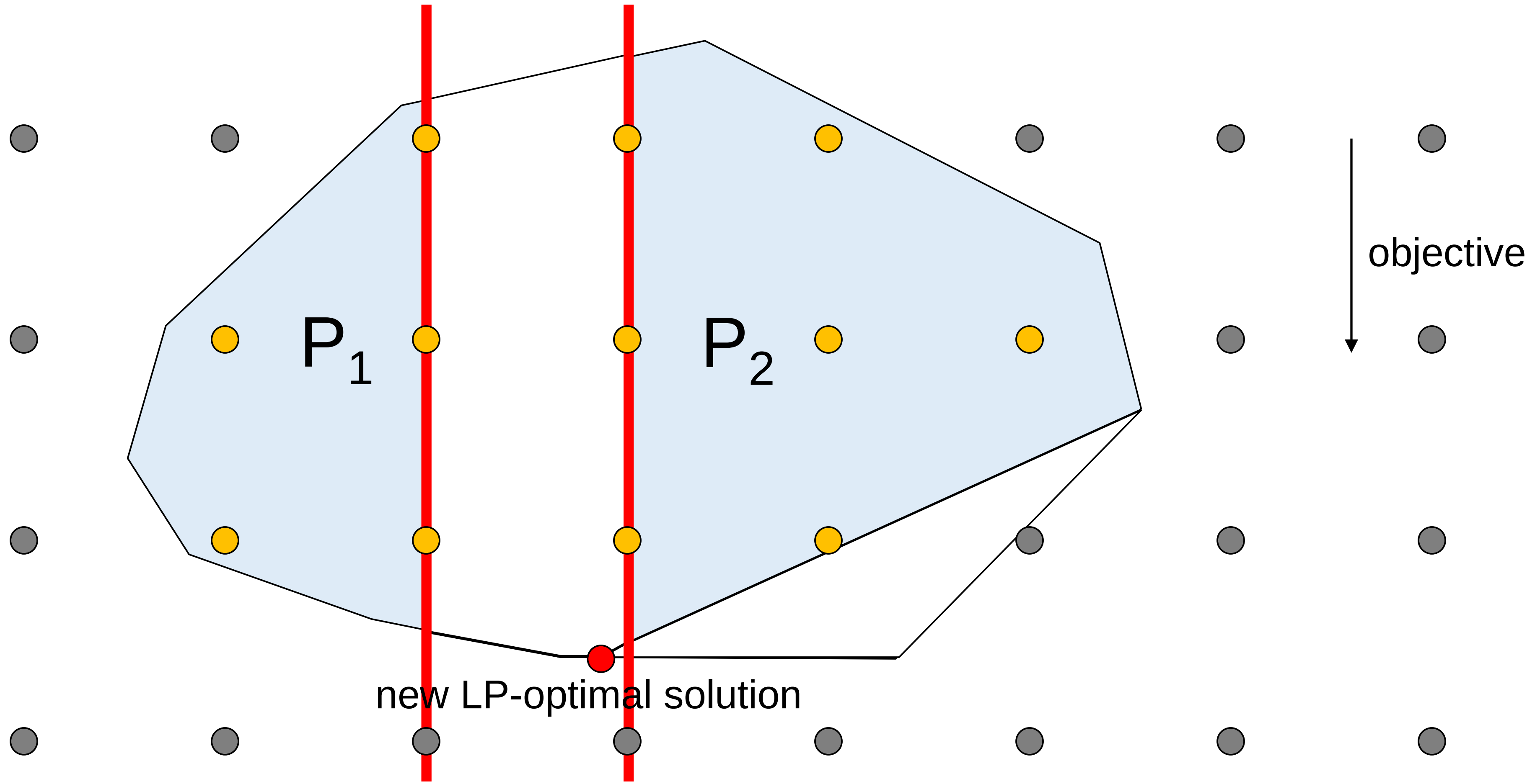
MIP – Cutting Planes



MIP – Cutting Planes



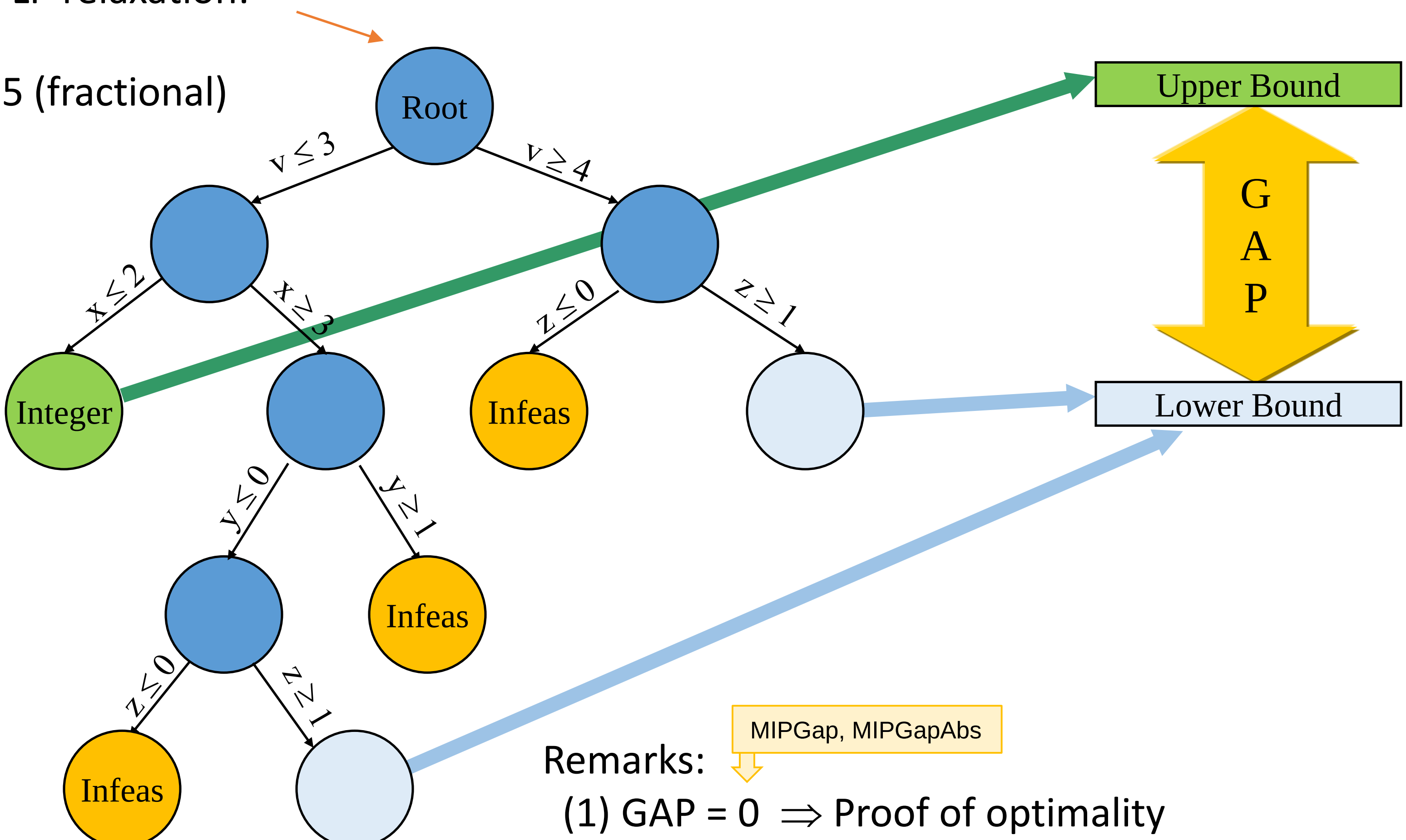
MIP – Branching



LP based Branch-and-Bound

Solve LP relaxation:

v=3.5 (fractional)



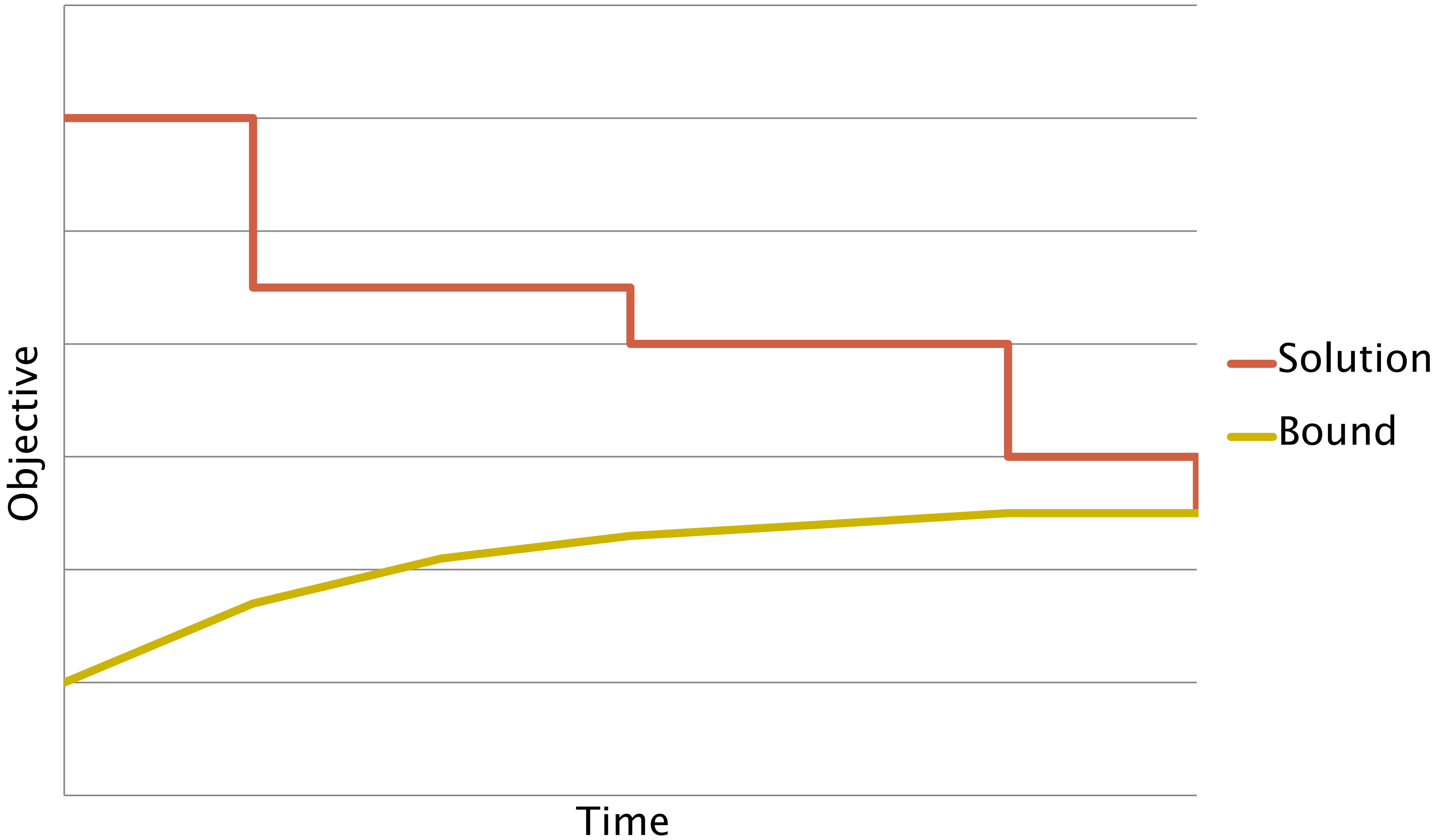
MIPGap, MIPGapAbs

Remarks:

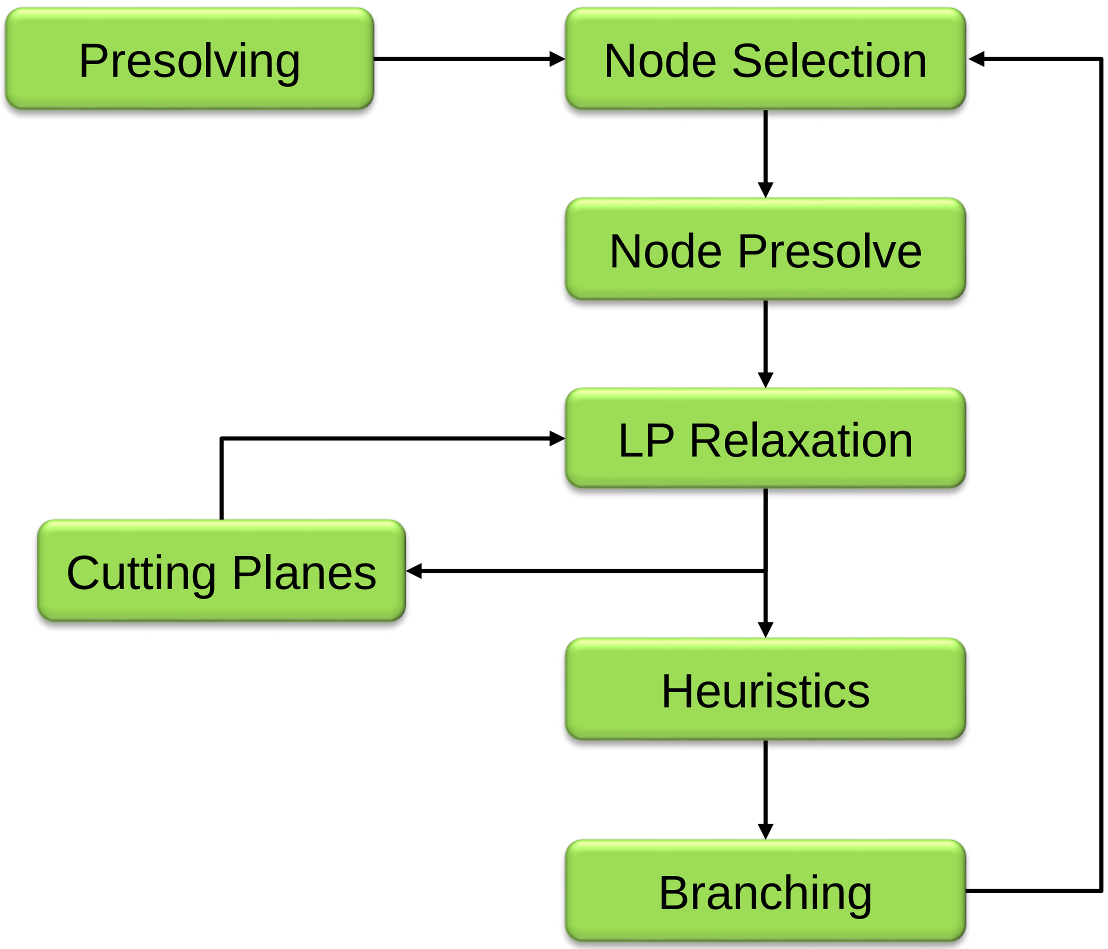
(1) GAP = 0 $\Rightarrow$ Proof of optimality

(2) In practice: good quality solution often enough

Solving a MIP Model



Branch-and-Cut



Branch-and-Cut

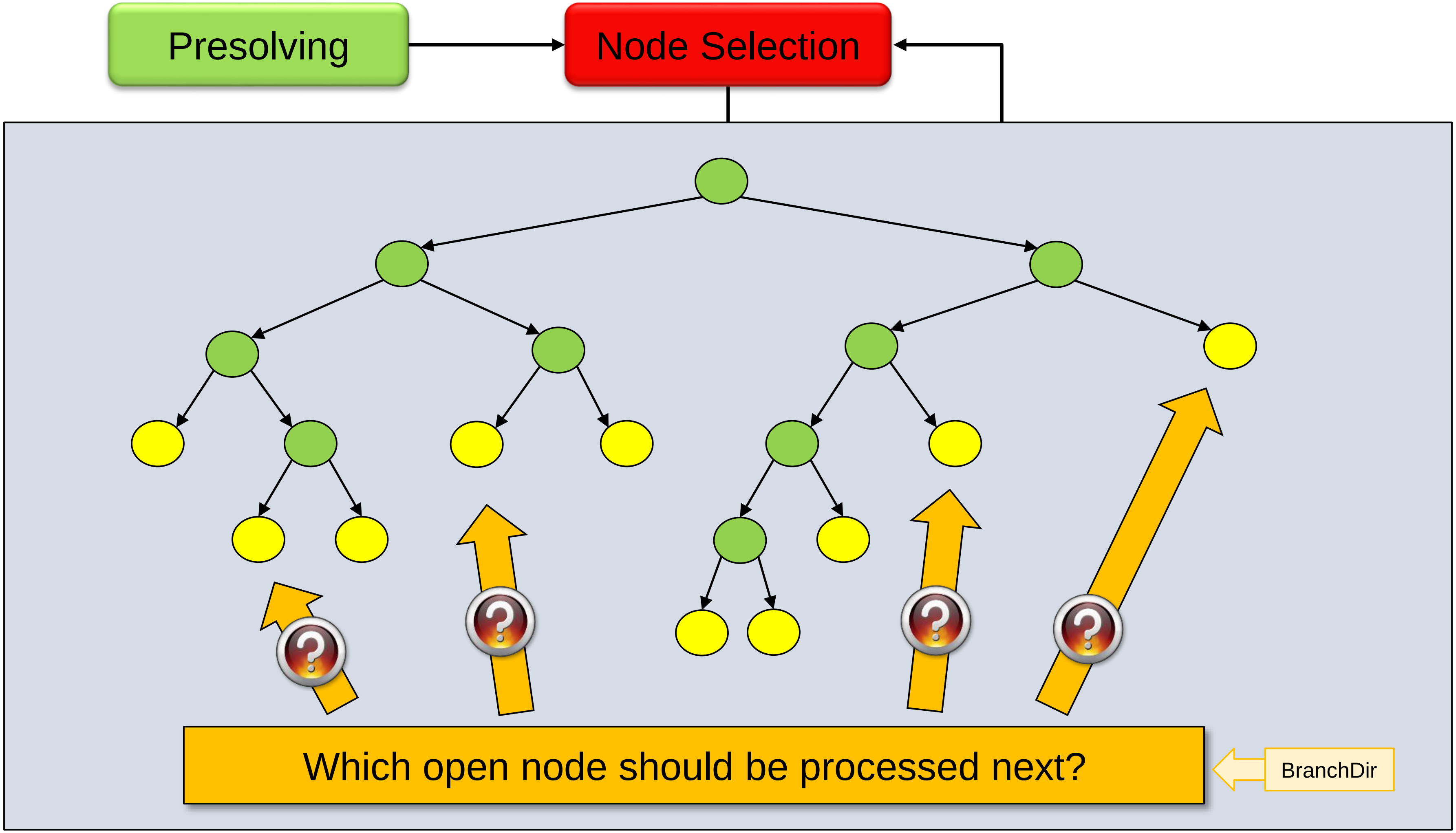
Presolving

Cutting Planes

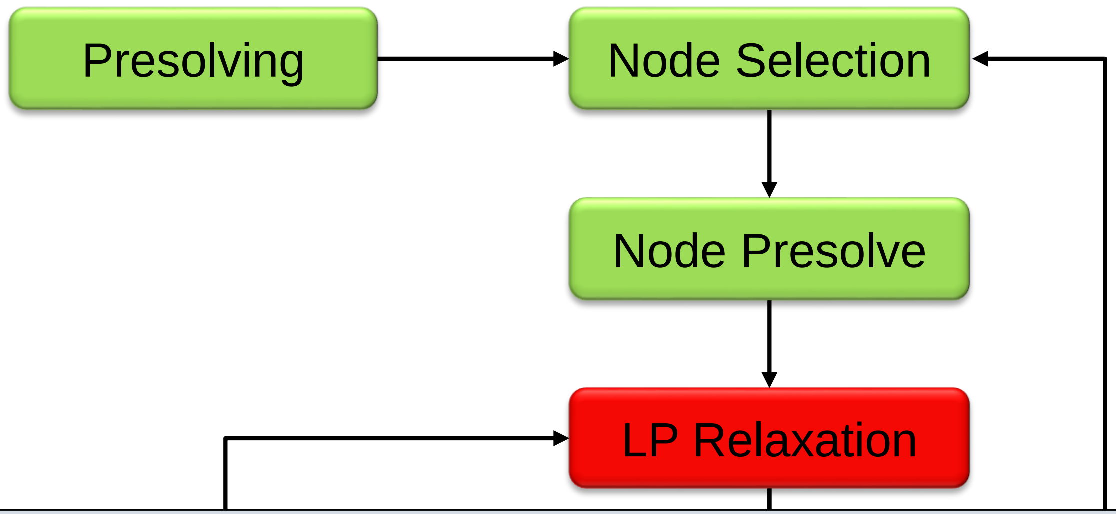
Branching

```
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/Noc

     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




```
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
```


GUROBI OPTIMIZATION

Branch-and-Cut

Presolving

Cutting Planes

```
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 secon

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

     0     0 11120.0279    0  154          - 11120.0279     -     -
     0     0 11526.8918    0  207          - 11526.8918     -     -
     0     0 11896.9710    0  190          - 11896.9710     -     -
     0     0 12151.4022    0  190          - 12151.4022     -     -
     0     0 12278.3391    0  208          - 12278.3391     -     -
...
  5485   634 12885.3652   52  143 12890.0000 12829.0134  0.47%  54.5

Cutting planes:
  Learned: 4
  Gomory: 46
  Cover: 39
  Implied bound: 8
  Clique: 2
  MIR: 112
  Flow cover: 27
  GUB cover: 11
  Zero half: 91

Explored 6008 nodes (357915 simplex iterations) in 27.17 seconds
Thread count was 4 (of 8 available processors)
```

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

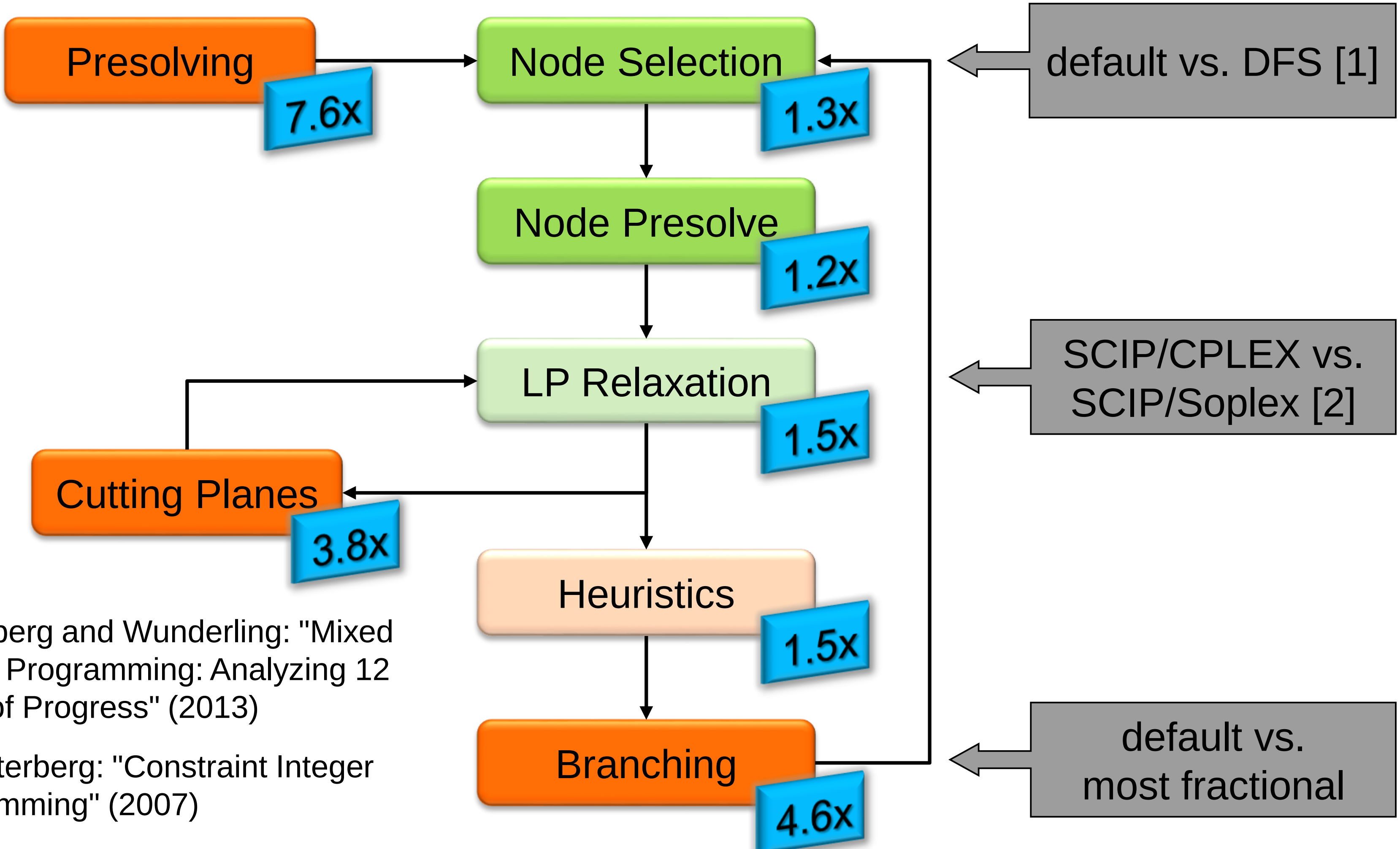
Presolvi

Cutting P

```
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
```

Branching

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