Algorithms II – MIP Details

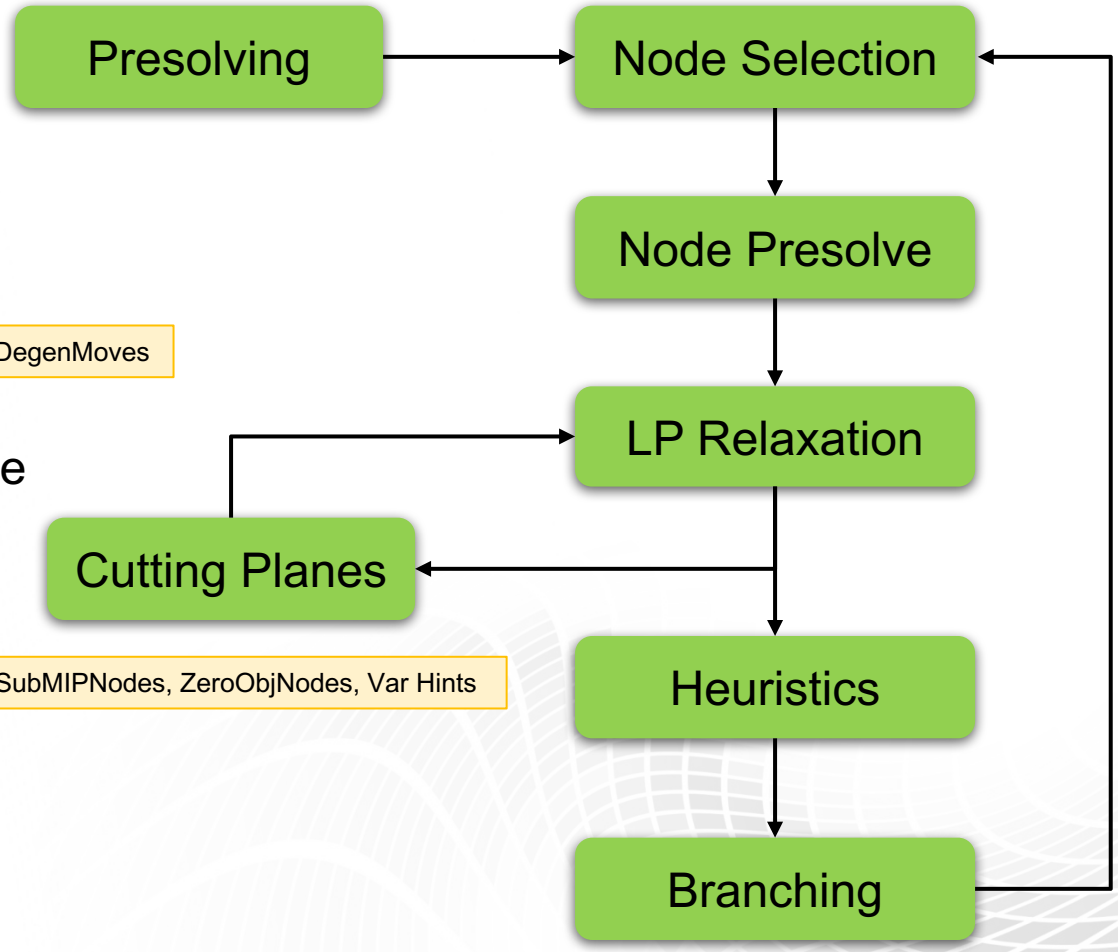
What's Inside Gurobi Optimizer

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

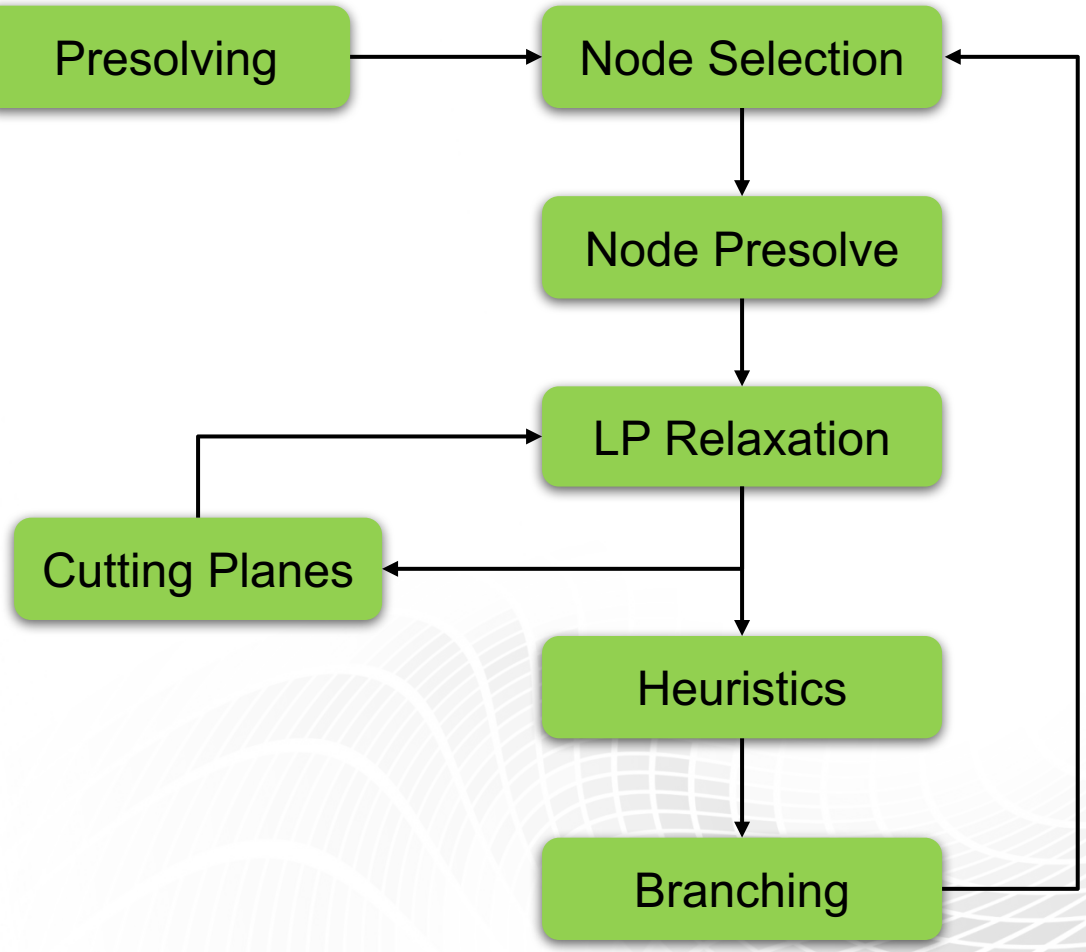
MIP Building Blocks

- Presolve ← Presolve, PrePasses, AggFill, Aggregate, DualReductions, PreSparsify, ...
  - Tighten formulation and reduce problem size
- Node selection ← BranchDir, Var Hints
  - Select next subproblem to process
- Node presolve ← Symmetry
  - Additional presolve for subproblem
- Solve continuous relaxations ← Method, NodeMethod, DegenMoves
  - Ignoring integrality
  - Gives a bound on the optimal integral objective
- Cutting planes ← Cuts, CutPasses, GomoryPasses, CliqueCuts, ...
  - Cut off relaxation solutions
- Primal heuristics ← Heuristics, MinRelNodes, PumpPasses, RINS, SubMIPNodes, ZeroObjNodes, Var Hints
  - Find integer feasible solutions
- Branching variable selection ← VarBranch
  - Crucial for limiting search tree size



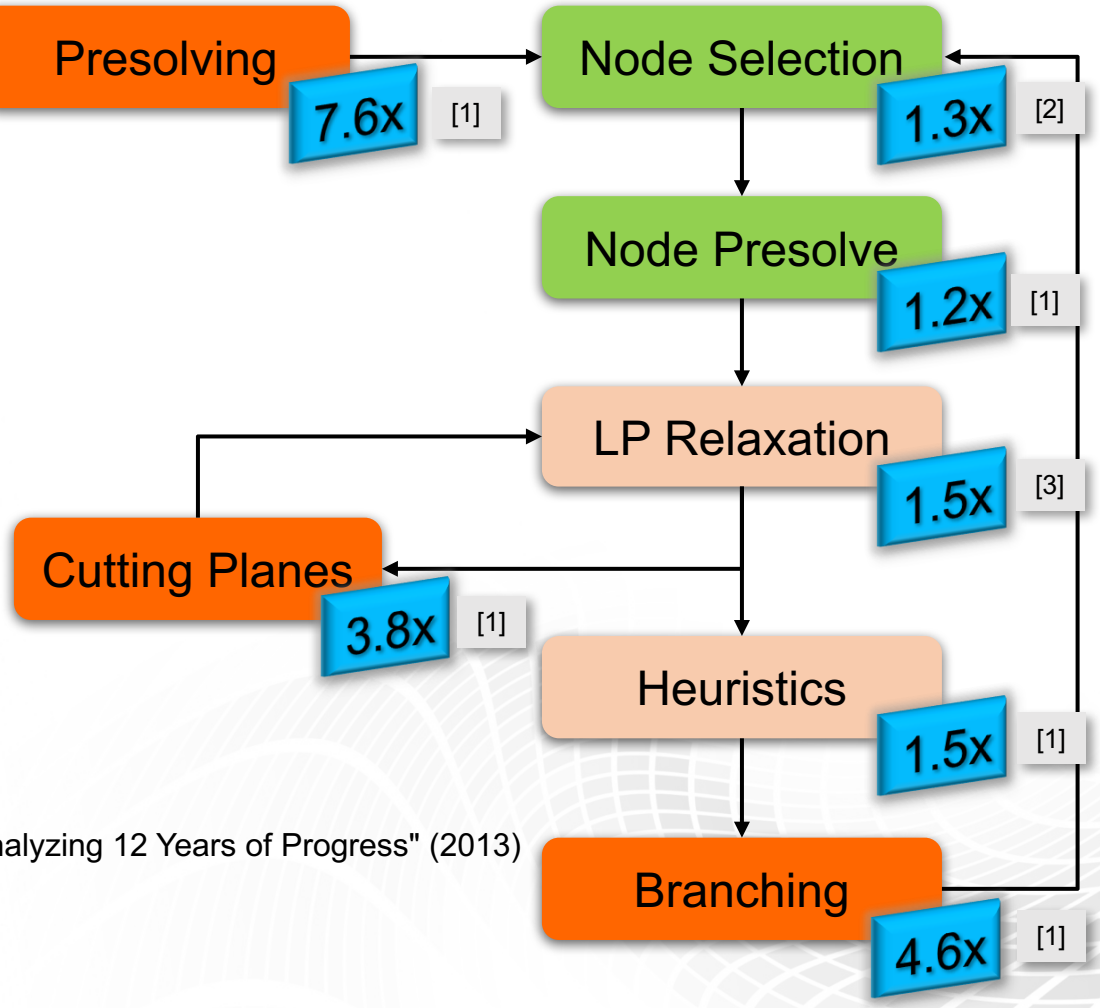
MIP Building Blocks

- Each box represents a giant bag of tricks
  - To cover everything would take weeks
- A sampling of techniques instead
  - One from each of the most important boxes



MIP Building Blocks

- Each box represents a giant bag of tricks
  - To cover everything would take weeks
- A sampling of techniques instead
  - One from each of the most important boxes



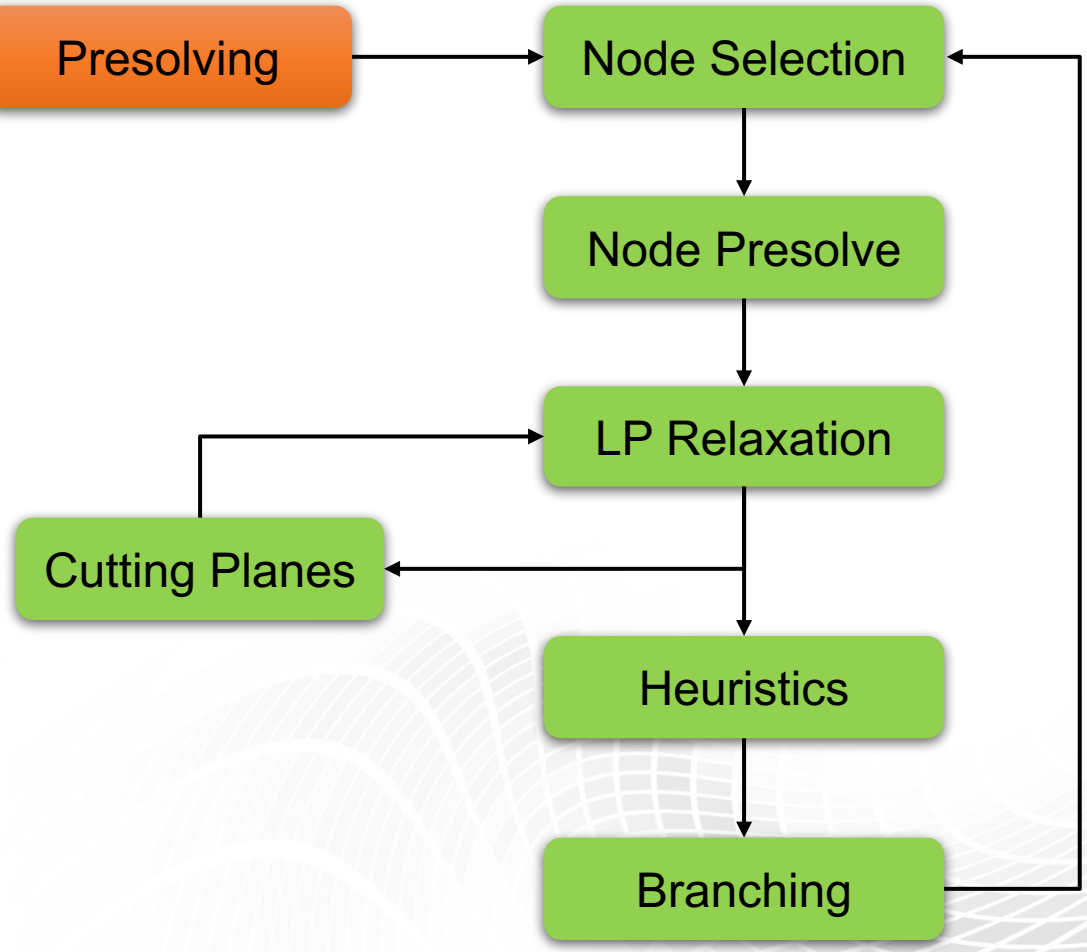
[1] Achterberg and Wunderling: "Mixed Integer Programming: Analyzing 12 Years of Progress" (2013)

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

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

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
- Cutting planes
  - Cut off relaxation solutions
- Primal heuristics
  - Find integer feasible solutions
- Branching variable selection
  - Crucial for limiting search tree size



Specific Presolve Topic: Constraint Strengthening

- Find a strictly stronger version of:
  - $a'x \leq b$, $l \leq x \leq u$, some x_j integral
  - Replace original with strengthened version

Three Parts To Each Strengthening

- Validity
  - Prove that modified constraint doesn't cut off any valid solutions
- Domination
  - Prove that modified constraint is strictly stronger than original
- And make it quick

Strengthening Approaches

- Special cases galore
  - Euclidean reduction
  - Coefficient reduction
  - …
- More general approaches
  - Pure binary knapsack
    - Tilt constraint using up-/down-lifting
  - Chvatal-Gomory rounding
  - MIR inequality
- We'll consider lots of specific examples
  - All arise from MIP models in our test library

Cost Versus Benefit

- Trade off cost of reduction against benefit
- Why worry about cost?
  - Only one presolve for each MIP model
- A few reasons:
  - One presolve can still be expensive
  - Aggressive use of sub-MIP heuristics (RINS)
    - Lots of truncated MIP solves
      - Multiple presolves, on smaller models
      - Presolve can be dominant cost on each
  - Strengthening as a cut separator

Strictly Stronger?

- Consider:
  - $5\ b1 + 3\ b2 + 3\ b3 + 3\ b4 + 8\ b5 \leq 8$
- Stronger…?
  - $4\ b1 + 4\ b2 + 4\ b3 + 4\ b4 + 8\ b5 \leq 8$
- Probably, but…
  - Doesn't strictly dominate original
    - (1, 0, 0, 0, 0.5) satisfies second, but not first
  - Could weaken relaxation
    - No definitive metrics
  - Hippocratic oath of presolve
    - "First, do no harm"
    - Lots of cases where it hurts

Binary Knapsack Constraints

- Binary knapsack strengthening:
  - 10 b24 + 5 b25 + 10 b26 + 11 b3 ≤ 23
  - Can strengthen:
    - RHS to 21
    - Coefficient on b25 to 10
- Binary knapsack strengthening a well studied problem
  - Tilt constraint by up-/down-lifting each variable
  - Pseudo-polynomial time algorithm
  - Cheap for all integer coefficients and small RHS
- Not that cheap in general
  - Similar computation required for cover lifting
  - Practical implementations often don't do exact lifting
    - Heuristic lifting on variables that are integral in relaxation
  - Our goal is to strengthen every constraint in the model
- Reality: practical for knapsacks with ~8 variables

Euclidean Reduction

- Divide through by coefficient GCD
- Example from MIPLIB model opm2.z3.s6:
  - 3203 b1 + 1936 b4 + 1803 b11 + 1430 b14 + 1390 b7 + 752 b2 + 715 b5 + 84 b12 ≤ 6325.5
- Simple special case of Chvatal-Gomory rounding
- Domination is obvious

Rounding Reduction

- Given:
  - $\sum ajxj \leq$ b, xj non-negative int
- Chvatal-Gomory rounding:
  - $\sum \lfloor aj \rfloor xj \leq \lfloor b \rfloor$
  - Why? $\sum \lfloor aj \rfloor xj \leq \sum ajxj$ and integral
  - Complement variable when frac(aj) > frac(b)
- Examples:
  - Bsp: 20 b24 + 24.1 b25 + 20 b26 + 20 b3 + 20 b4 + 20 b27 - 23.5 b5 - i11 ≤ 16.8
  - fmd2: -10.5 b2 + 2 i9 + i36 ≤ 20.5
- Domination check:
  - sup($\sum (aj - \lfloor aj \rfloor)\, xj$) versus
  - $b - \lfloor b \rfloor$

Rounding Reduction

- Need to consider different scalings
  - MIPLIB model opm2.z3.s13
    - $2\ b36 + 2\ b16 + 2\ b32 + 2\ b34 + 2\ b6 + 2\ b40 + 2\ b32 + 2\ b45 + b33 \leq 15$
  - Multiply through by 0.5 (zero-half cut)
- What are good scaling values?
  - Marchand and Wolsey: divide by 1, 2, 3, 4, 5
  - Cornuejols, Li, and Vandenbussche (k-cuts): scale by 1, 2, 3, 4, 5, …
  - Some scalings clearly dominate others
    - Other than those, basically brute force

Different Rounding Reduction

- Different rounding reduction
  - Original: $\sum ajxj \geq$ b, aj > 0, xj non-negative ints
  - Modified: $\sum[aj]$xj≥ $[b\ ]$
  - Model p0548 – original constraint:
    - 5 b1 + 4 b2 + 4 b3 + 3 b4 + 3 b5 ≥ 6
  - Divide by 5 and round up:
    - b1 + b2 + b3 + b4 + b5 ≥ 2
- Domination check:
  - aj ≥ (b/b') a'j for every aj

Coefficient Reduction + Special Cases

- Example (MIPLIB model bm23):
  - After flipping sense and complementing
  - 8 b1 + 5 b2 + 3 b3 + 3 b4 + b5 + b6 + b7 ≥ 4
- A bit more subtle (model bndl20000):
  - 80 b6 + 80 b38 + 111.7 b68 + 112.2 b72 + 112.1 b80 + 160 b2846 ≥ 160
- And…
  - 120 b1 + 120 b2 + 120 b3 + 74 b4 + 18 b5 + 12 b6 ≥ 120

Summary

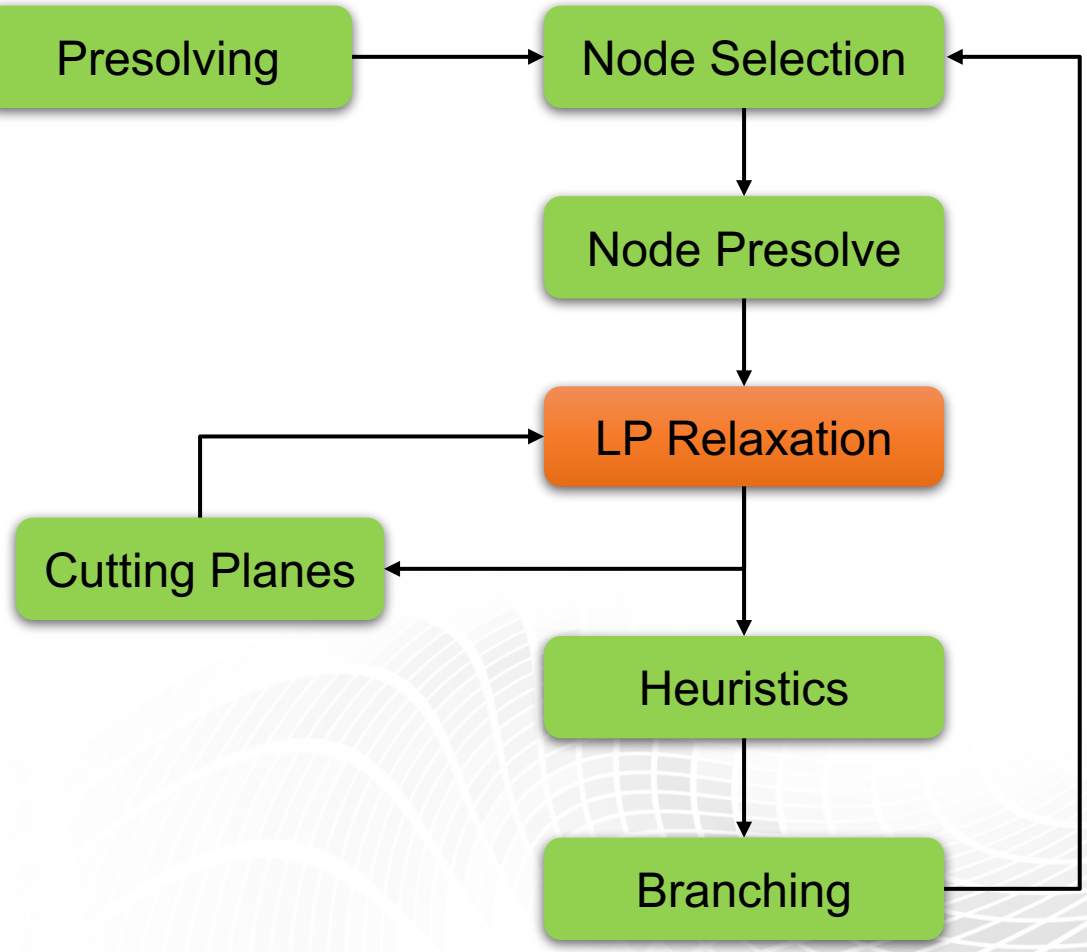
- Our constraint strengthening has 5 different reduction types
  - Multiple scalings for several of them
  - 27 passes through constraint if we don't find any reductions
    - More if we do
  - Our estimate:
    - Roughly 1-2% of total MIP runtime goes to constraint strengthening

Strengthening Performance Impact

- Effect of turning it off entirely:
  - For >1s: 3.0% mean performance degradation
  - For >10s: 4.0% mean performance degradation
- Effect of turning off cut strengthening only:
  - For >1s: 0.3% mean performance degradation
  - For >10s: 1.0% mean performance degradation
- On a test set of 2444 test models

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
- Cutting planes
  - Cut off relaxation solutions
- Primal heuristics
  - Find integer feasible solutions
- Branching variable selection
  - Crucial for limiting search tree size



Specific Relaxation Topic: Degenerate Moves

GUROBI OPTIMIZATION

- Solve a relaxation at each node
  - Relax integrality constraints
  - Find *an* optimal solution to continuous model
- Most models have multiple optimal solutions
  - An *optimal face*
  - Are some better than others?
- Simple idea
  - Search for an optimal solution that minimizes the number of fractional integer variables
  - Use approach similar to Feasibility Pump (Fischetti, Glover, & Lodi, 2005)

Degenerate Moves

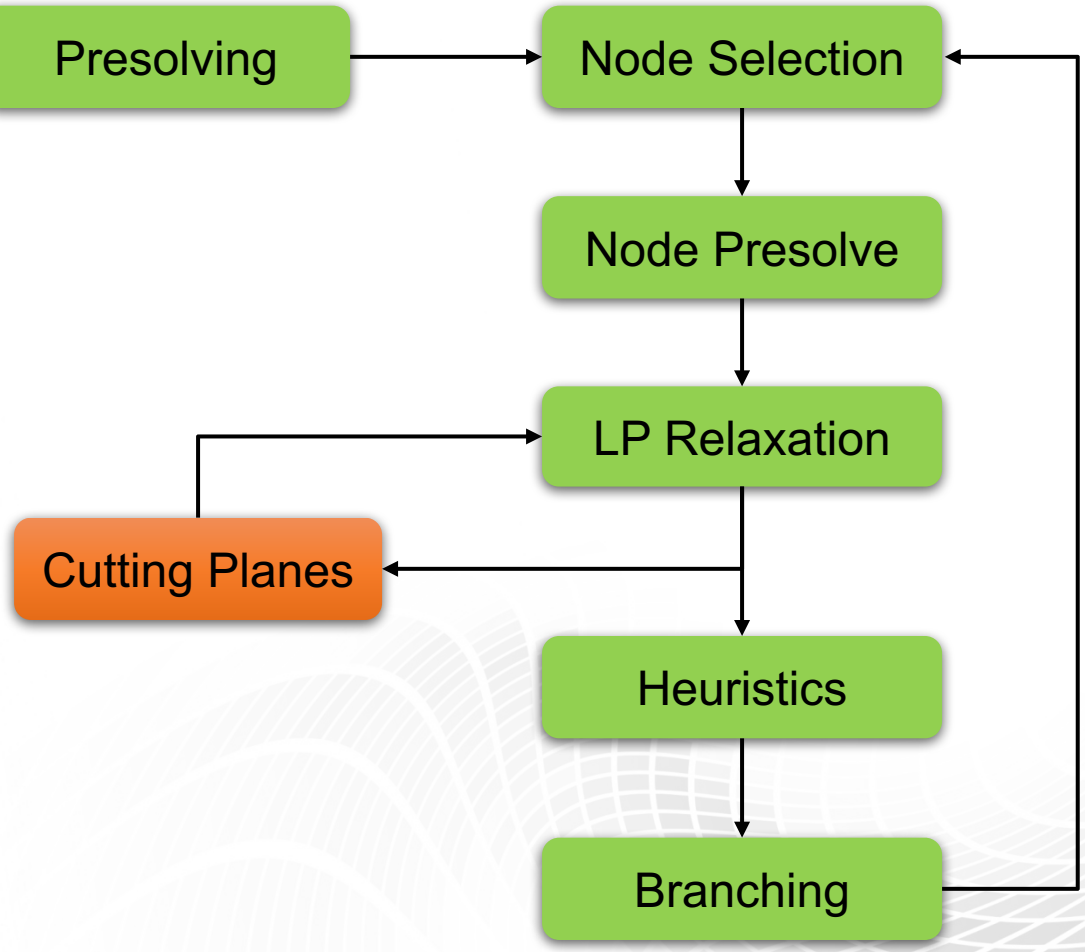
- Given
  - z* = min(c'x: Ax=b)
- Formulate a new model
  - Minimize: f'x
  - Subject to: Ax=b, c'x=z*
- Explore the space of degenerate solutions
  - Any solution is optimal for the original relaxation
  - Objective constraint enforced by fixing variables with non-zero reduced costs
    - Typically fixes a large fraction of the variables
- Choose objective f to incentivize integrality
  - Given current solution x
  - fj = 1 if xj - $\lfloor xj \rfloor \leq 0.5$
  - fj = -1 if xj - $\lfloor xj \rfloor \geq 0.5$
  - Plus a small random perturbation
- Continue solving degenerate move LP until integer infeasible count no longer improves

Degenerate Moves

- How expensive is this process?
  - Most variables typically fixed – LP is small
  - Solving a sequence of LPs can still be expensive
- But, given an integer-infeasible variable…
  - Typically much cheaper to fix it by solving an LP than by branching
- *DegenMoves* parameter allows you to turn this process off

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
- Cutting planes
  - Cut off relaxation solutions
- Primal heuristics
  - Find integer feasible solutions
- Branching variable selection
  - Crucial for limiting search tree size



Many Cut Types in Gurobi

- Available types:
  - Gomory
  - Cover
  - Implied bound
  - Clique
  - Mixed Integer Rounding (MIR)
  - Flow cover
  - Flow path
  - GUB Cover
  - Zero-half
  - Mod-K
  - Network
  - SubMIP

Extremely Robust

- Most cut arguments you can think of are covered by one of these general cut types
- Example: sports tournament scheduling
  - Every team must play a game each week
  - Odd-set cut [Padberg & Rao, 1982]:
    - In any odd subset of teams, at least one team must play a team outside the set
- MIP formulation:
  - x_{ijt}: team i plays team j in week t
  - Each team must play a game each week
    - $\sum_j x_{ijt} = 1$ for all i,t
    - A single x_{ijt} variable appears in two such constraints (for a given t)
      - Team i must play a game; team j must play a game

Odd-Set Cut as Zero-Half Cut

- Choose an odd set S
- Sum the play constraints for variables in S
  - $\sum_{i,j\ in\ S} 2\ xijt + \sum_{i\ in\ S, j\ not\ in\ S} xijt = |S|$
- Observe:
  - First term is even
  - Right-hand side is odd
  - Conclusion: second term must be at least one
- This is exactly the odd-set cut
- Widely cited cut type
  - [Padberg & Rao, 1982], [Trick, 2003], [Noronha, Ribeiro, Duran, Soyris, & Weintraub, 2006]
- This is actually a zero-half cut
  - Automatically found by zero-half cut separator
  - No specialized argument required

Zero-Half Cut Separation

- General form:
  - Find aggregated constraint a'x <= b where LHS is even and RHS is odd
  - Approach: mod-2 Gaussian elimination

- Example:
  - 2 x1 + x2 + x3 + x4 <= 2
  - x2 + x4 <= 1
  - 2 x2 + x3 <= 2
  - x1 = 0.5, x2 = 1, x3 = 0, x4 = 0

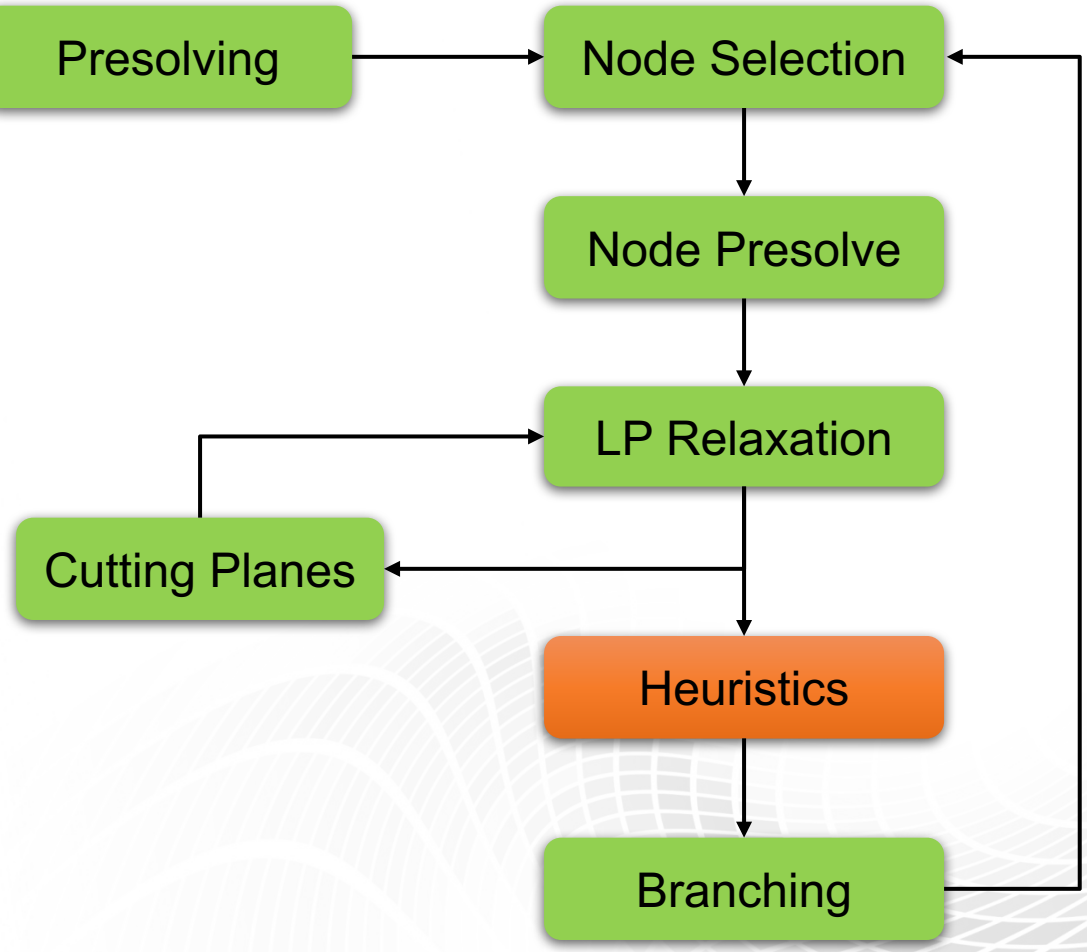
  - 2 x1 + 2 x2 + x3 + 2 x4 <= 3 (add first and second inequalities to "eliminate" x2)
  - 2 x1 + 4 x2 + 2 x3 + 2 x4 <= 5 (add in third inequality to "eliminate" x3)
  - x1 + 2 x2 + x3 + x4 <= 2.5

Specialized Cuts

- Very common experience
  - Specialized argument automatically captured by general cut type
- Current set of automatic cutting planes are extremely powerful and robust

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
- Cutting planes
  - Cut off relaxation solutions
- Primal heuristics
  - Find integer feasible solutions
- Branching variable selection
  - Crucial for limiting search tree size



MIP Heuristics

- MIP solvers find new feasible solutions in two ways
  - Branching
  - Primal heuristics
- Properties of a good heuristic
  - Quick
    - Finds solutions earlier than branching
  - Captures problem structure
    - Exploits structure more effectively than branching
  - General
    - Finds solutions for lots of models

Example MIP Heuristic – Rounding

- Start with node relaxation solution
- Round integer variables

- Quick?
  - Very quick
- Captures problem structure?
  - No
- General?
  - Finds solutions to lots of easy models

Example MIP Heuristic - RINS

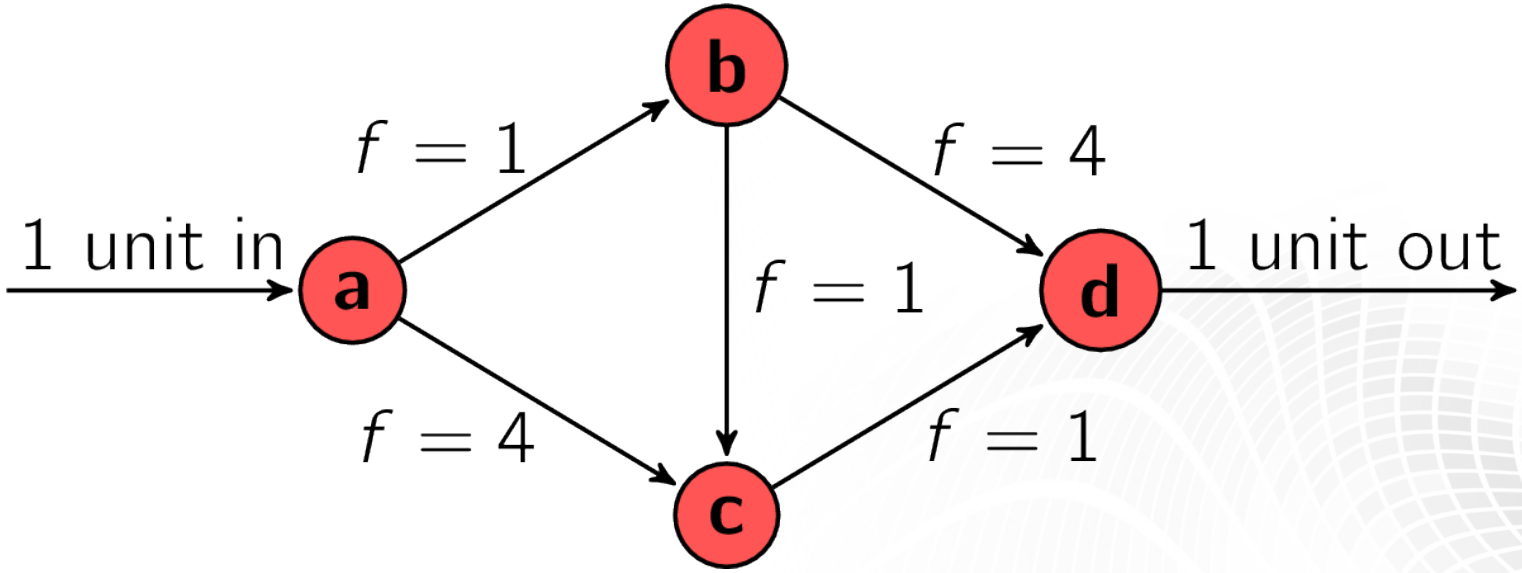
- Relaxation Induced Neighborhood Search
  - Start with:
    - Node relaxation solution
    - Best known integer feasible solution
  - Fix variables whose values agree in both
  - Solve a MIP on the rest
- Quick?
  - No – solves a MIP
- Captures problem structure?
  - Yes – searches a neighborhood of the relaxation
- General?
  - Yes – effective on a variety of models

Fixed-Charge Network Heuristic

- Gurobi has 16 heuristic types
  - Adaptive strategies decide when to apply each
- Specialized heuristic:
  - Recognizes and exploits fixed-charge network structure
  - Very common structure
    - Commodity flows through a network
    - Variable (per-unit) charge for flow on an edge
    - Fixed charge to open an edge
  - Heuristic recognizes:
    - Pure fixed-charge network models
    - Fixed-charge networks embedded in larger MIP models

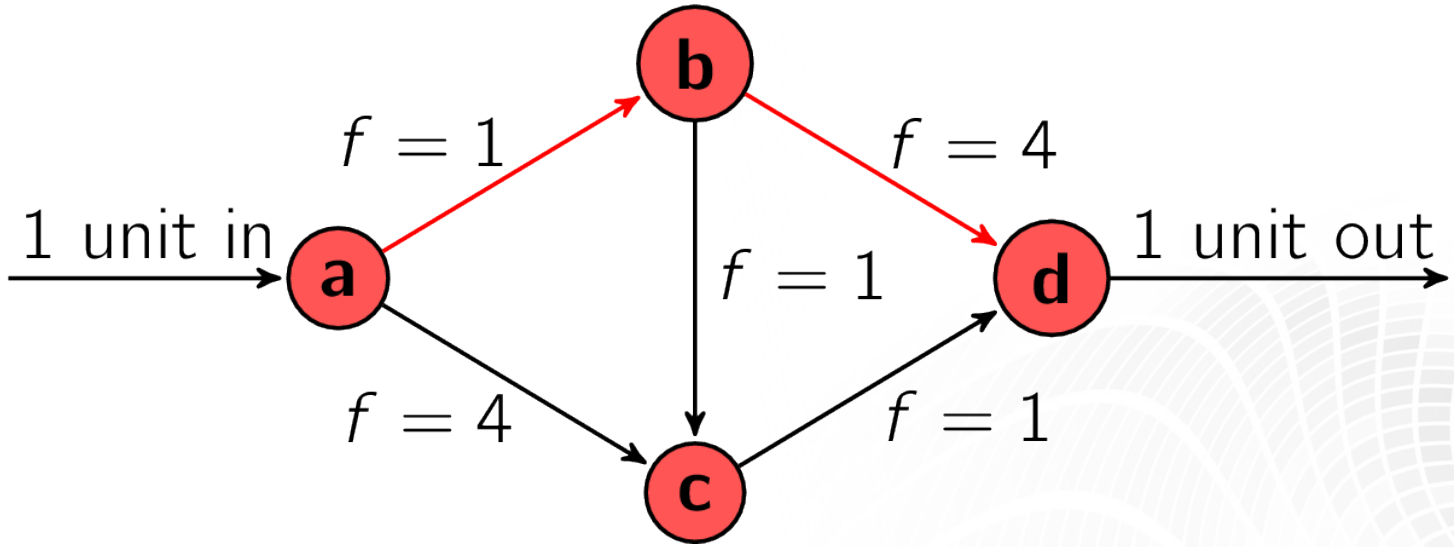
Fixed-Charge Network Flow

- Simple example:
  - Move one unit of flow from source to sink
  - Variable cost on all edges is 1.0
  - Capacity of each edge is 100
  - Fixed charges are listed…



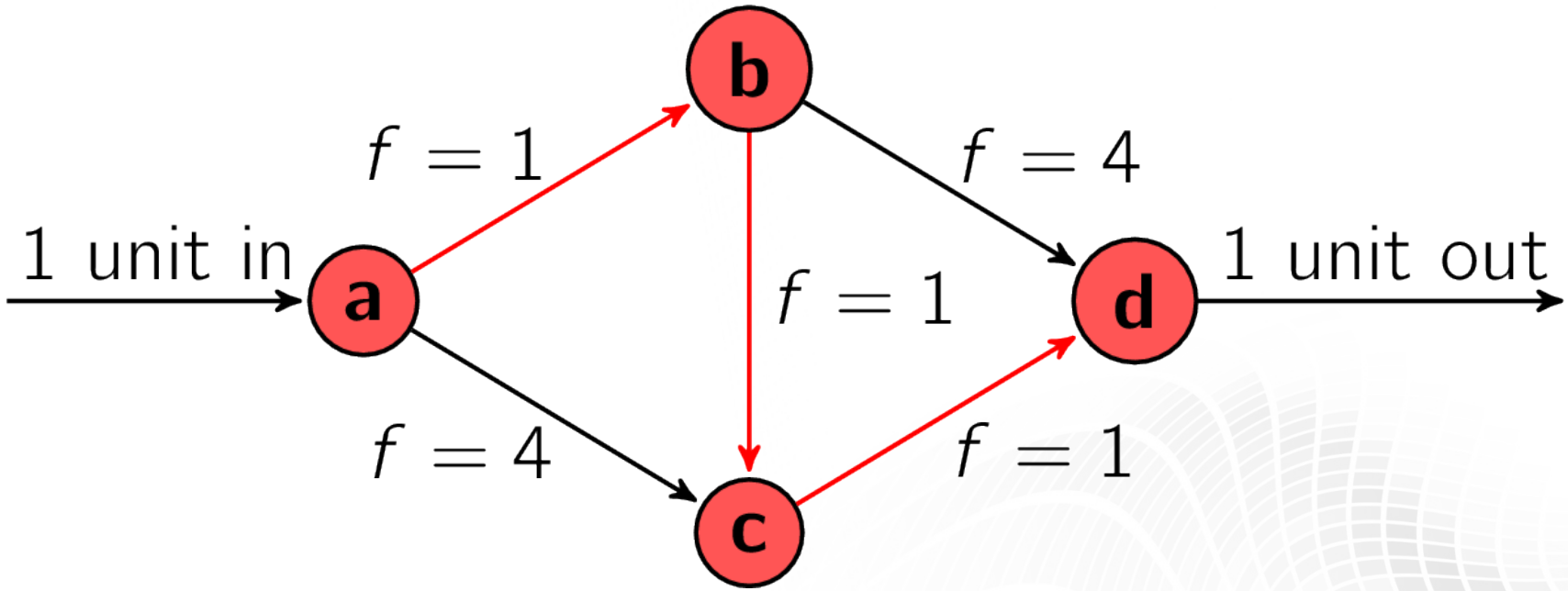
Relaxation solution

- Fixed charges not fully captured in relaxation
- Optimal relaxation cost:
  - 2 (variable) + 0.05 (relaxed fixed charges)
- Cost for corresponding fixed-charge solution:
  - 2 (variable) + 5 (fixed charges) = 7



Integer Solution

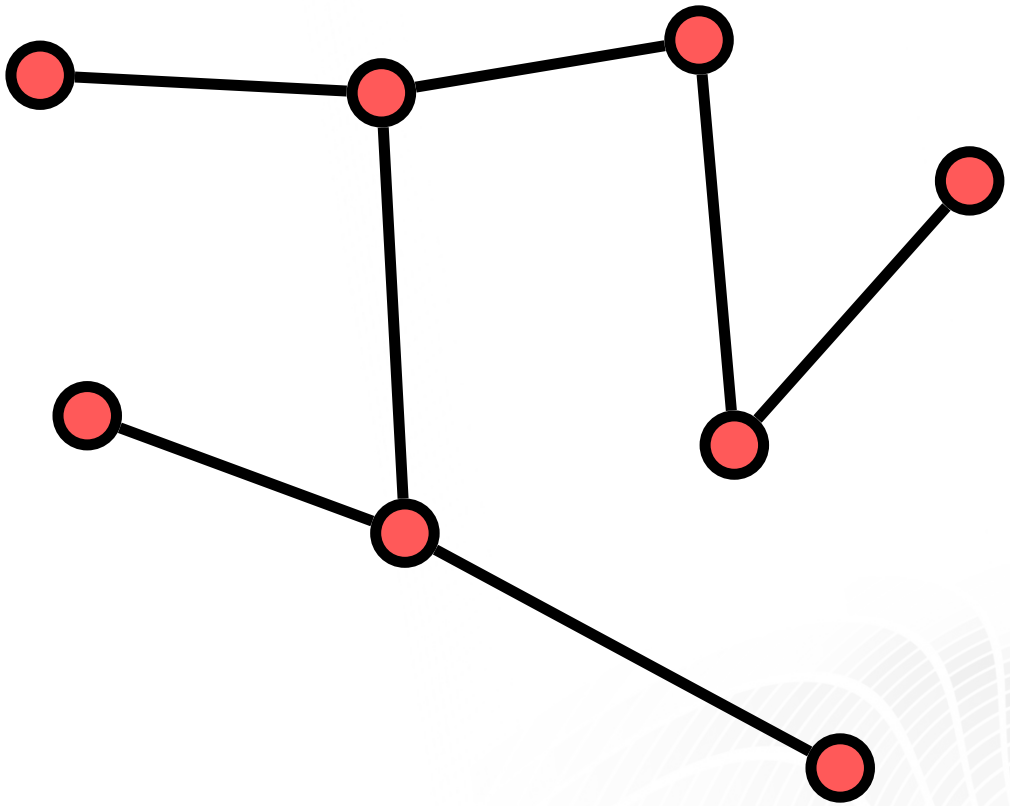
- Optimal integer solution:
  - Cost: 3 (variable) + 3 (fixed charges) = 6



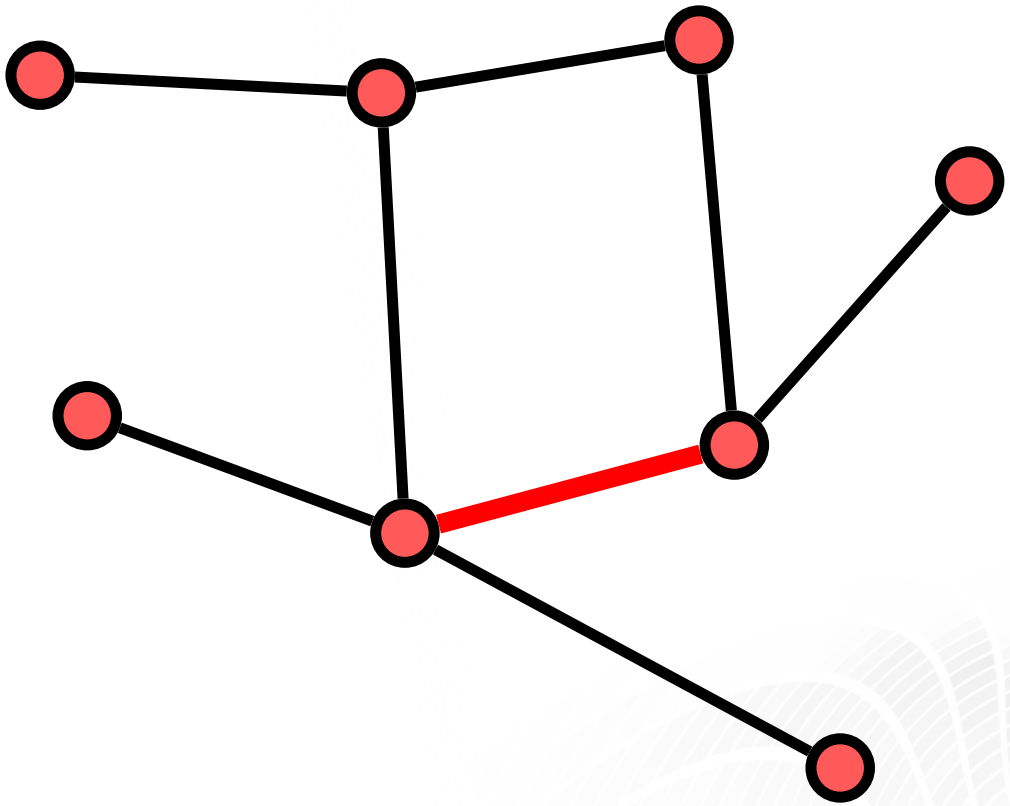
Network Simplex

- Specialized algorithm for solving continuous network models
- Key property of continuous networks:
  - Optimal solution has no cycle of free edges
- Simplex pivots performed directly on graph:
  - Simplex basis defined by a spanning tree
  - Use fast tree and graph algorithms
  - No linear algebra required

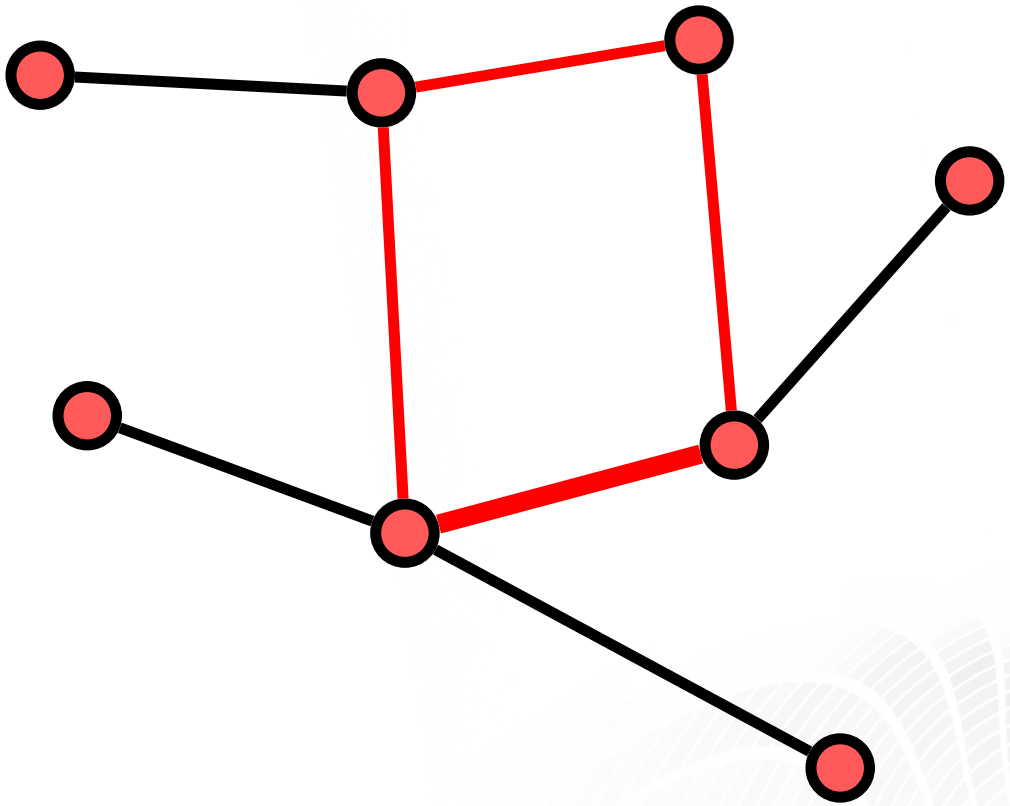
Network Simplex



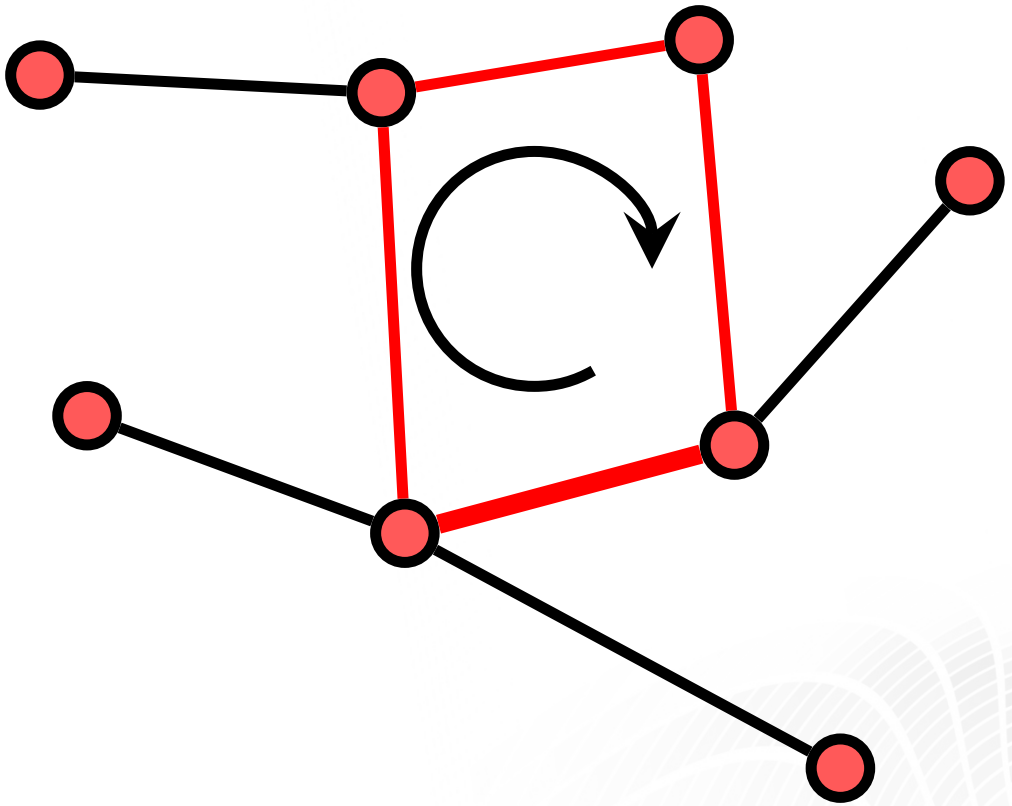
Choose an entering variable



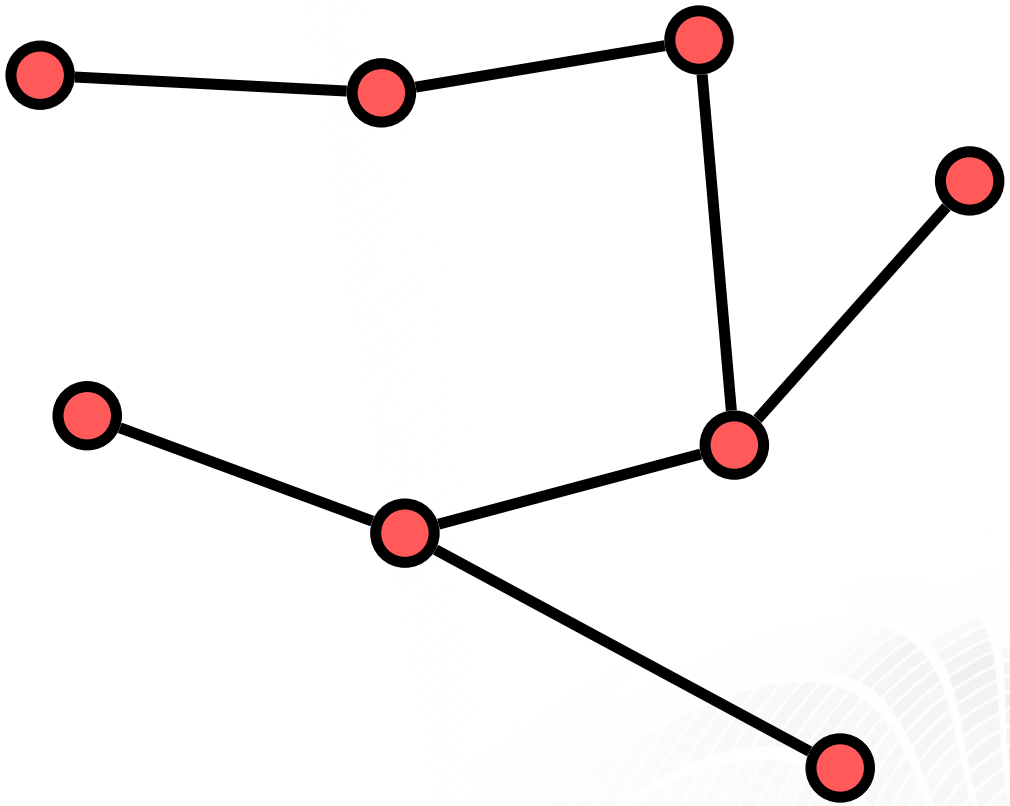
Find a cycle



Push flow around a cycle



Remove leaving variable



Network Simplex

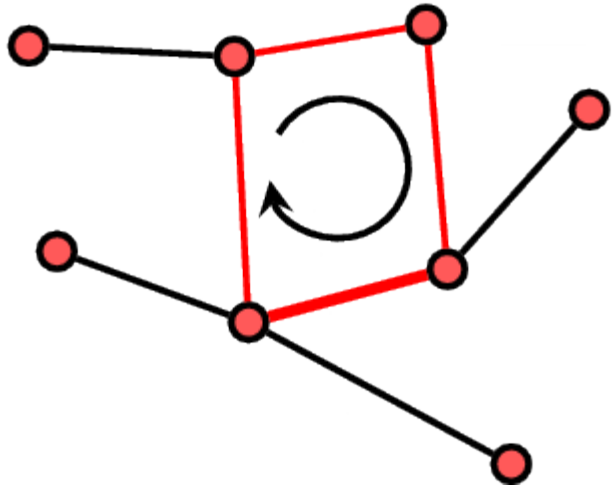
- Use reduced costs to choose entering variables
- Repeat pivots until optimal basis identified
- Pivots are *much* faster than standard simplex pivots
  - Number of pivots typically much larger too
    - Simplistic pricing

Integer Network Heuristic

- Idea:
  - Apply network simplex algorithm to fixed-charge network model
- Steps:
  - Choose entering variable
  - Find cycle
  - Push flow around cycle
  - Identify leaving variable

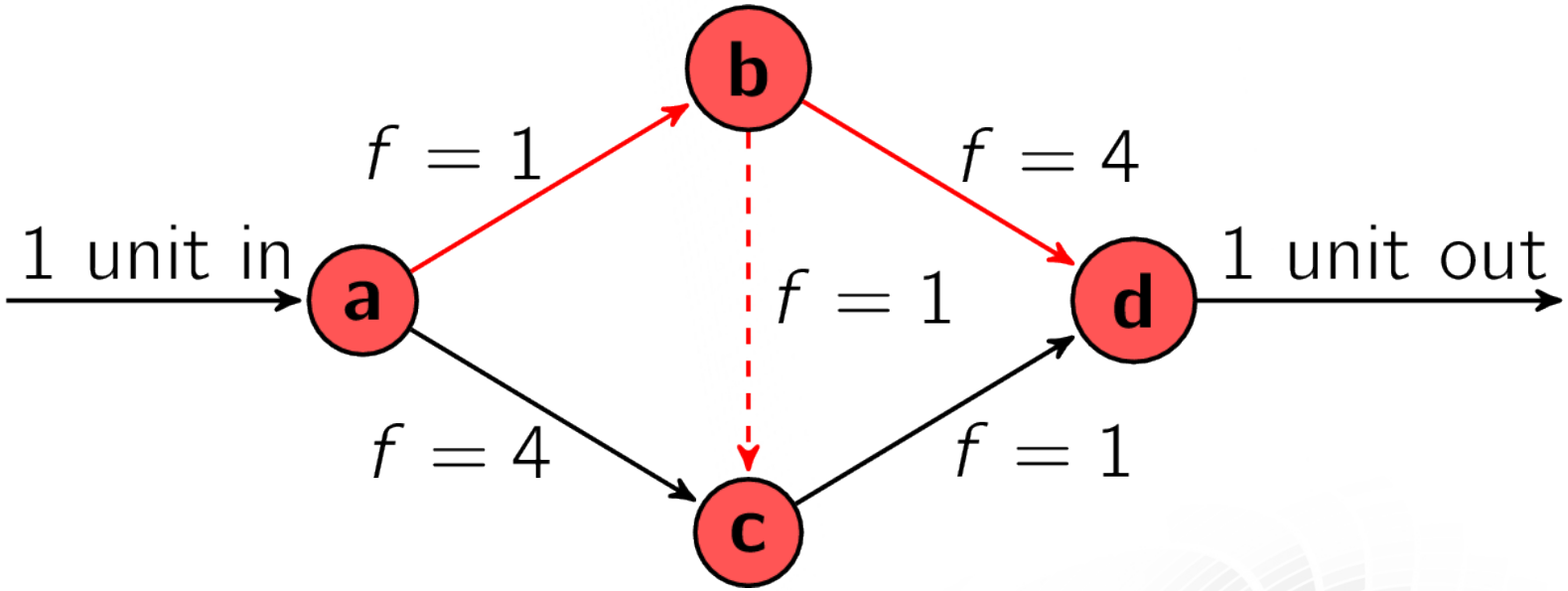
Pushing Flow Around A Cycle

- For continuous model, cost is linear function of flow
- For fixed-charge model, cost is a step function
  - When flow on an edge goes to zero, fixed-charge also goes to zero
- Net result:
  - Need to also consider circular flows that increase the continuous flow cost



Our Earlier Example

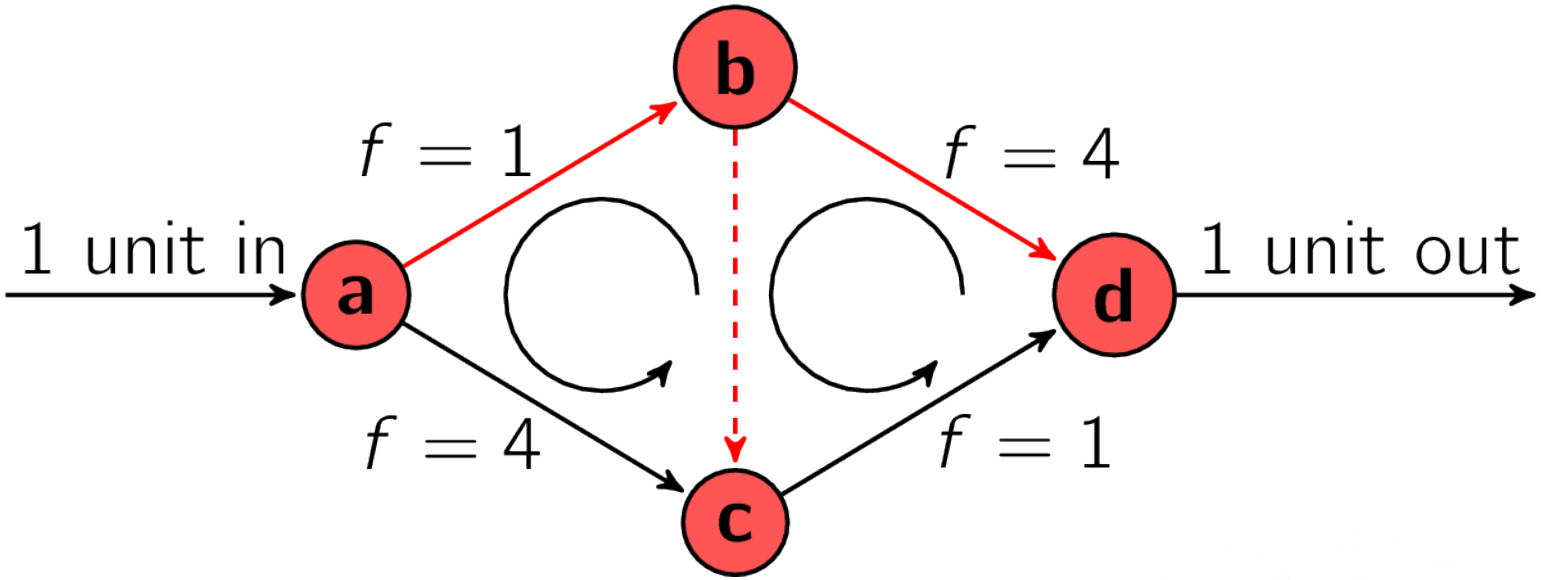
- Relaxation solution (as a spanning tree):



- 1 unit flow on *ab*, 1 on *bd*, 0 on *bc*
- Consider adding *ac* or *cd*

Our Earlier Example

- Add ac: can't push flow backward on *bc*



- Add *cd*:
  - Increase flow on *bc* and *cd* (to 1 unit)
    - Increases cost by 2 (continuous) + 2 (fixed charges)
  - Decrease flow on *bd* (to 0)
    - Decreases cost by 1 (continuous) + 4 (fixed charge)

How to choose entering variable?

- Problem:
  - No notion of reduced costs in fixed charge network
  - How should we price?
- Consider adding every edge in the graph
  - Underlying graph typically small compared to MIP representation of fixed-charge model

Evaluating the Heuristic

- Quick?
  - Very quick
- Captures problem structure?
  - Network structure
- General?
  - Any model with embedded fixed-charge network

Results

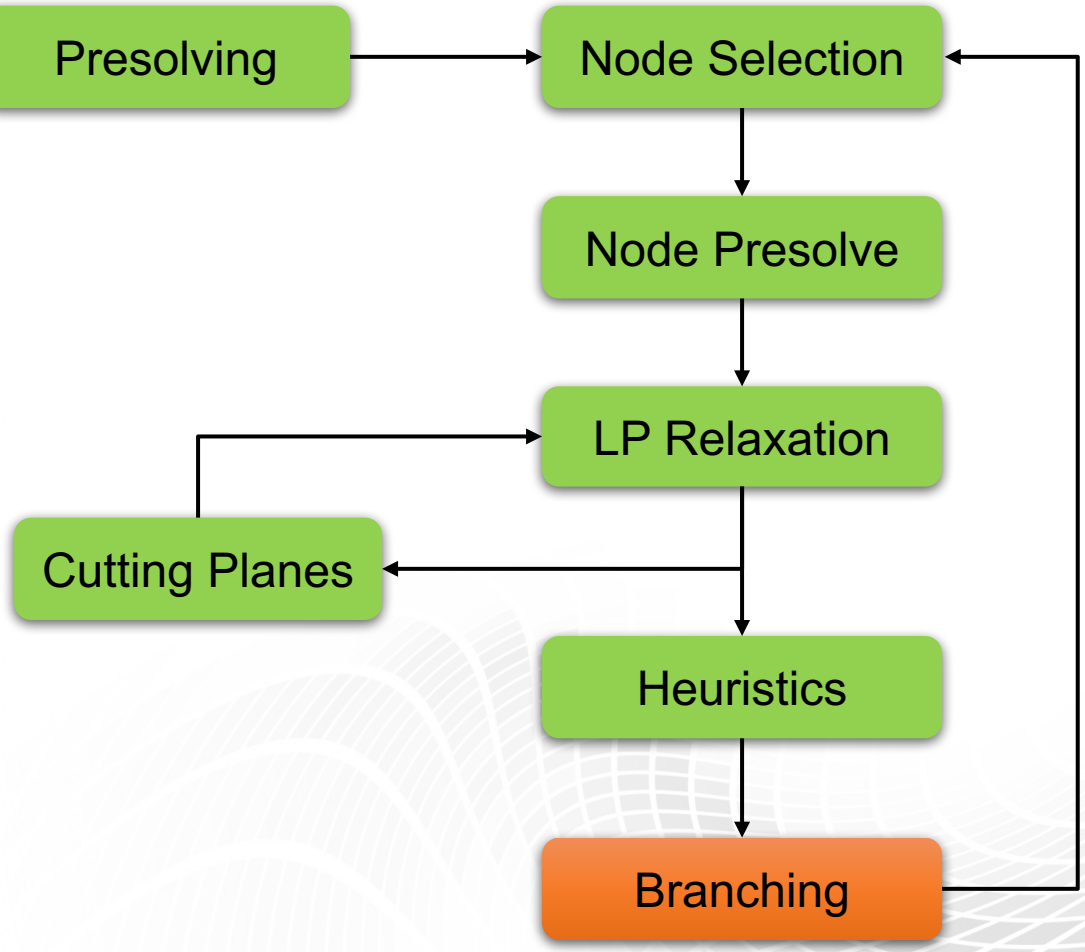
- Cost of considering every possible entering edge
  - Less than one-tenth the cost of solving one MIP node
- Use heuristic in two situations:
  - When a new feasible solution is found
  - At some nodes in the MIP search (using the relaxation solution)

Results

- Computational results:
  - A set of 54 fixed-charge network models
  - Run on an i7-3770K system
- Time to solve the model to optimality
  - 4% improvement
- Time to find the optimal solution (without proving it is optimal)
  - 24% improvement

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
- Cutting planes
  - Cut off relaxation solutions
- Primal heuristics
  - Find integer feasible solutions
- Branching variable selection
  - Crucial for limiting search tree size



Branching Variable Selection

- Given a relaxation solution x*
  - Branching candidates:
    - Integer variables x_j that take fractional values
      - $x_j = 0.5$ produces two child nodes ($x_j = 0$ or $x_j = 1$)
  - Need to pick one
    - 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

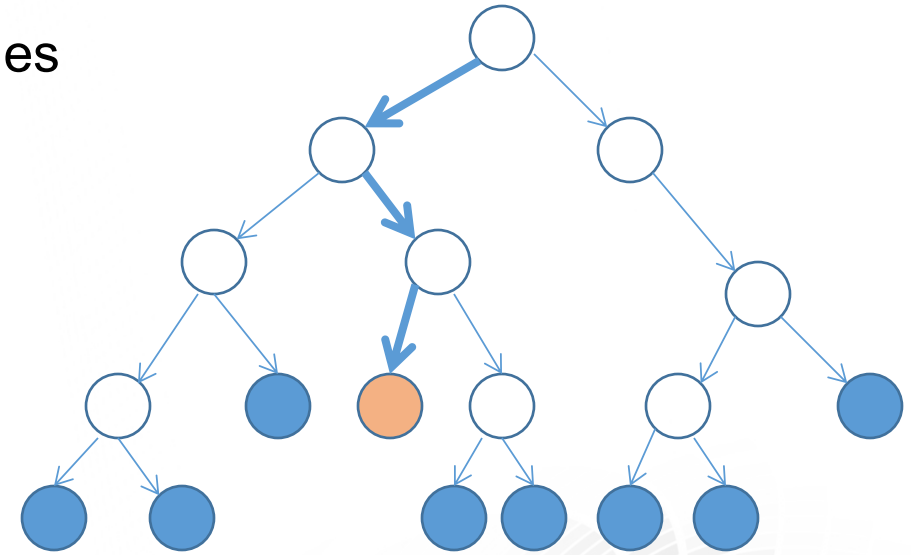
- Use historical data to predict impact of a branch:
  - Record cost$_x$ = Δ_{obj} / Δ_x for each branch
    - Need a scheme for infeasible branches too
  - Store results in a pseudo-cost table
    - Two entries per integer variable
      - Average (or max) down cost
      - Average (or max) up cost
  - Use table to predict cost of a future branch

Pseudo-Cost 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, 2002]
  - Don’t rely on historical data until pseudo-cost for a variable has been recomputed r times

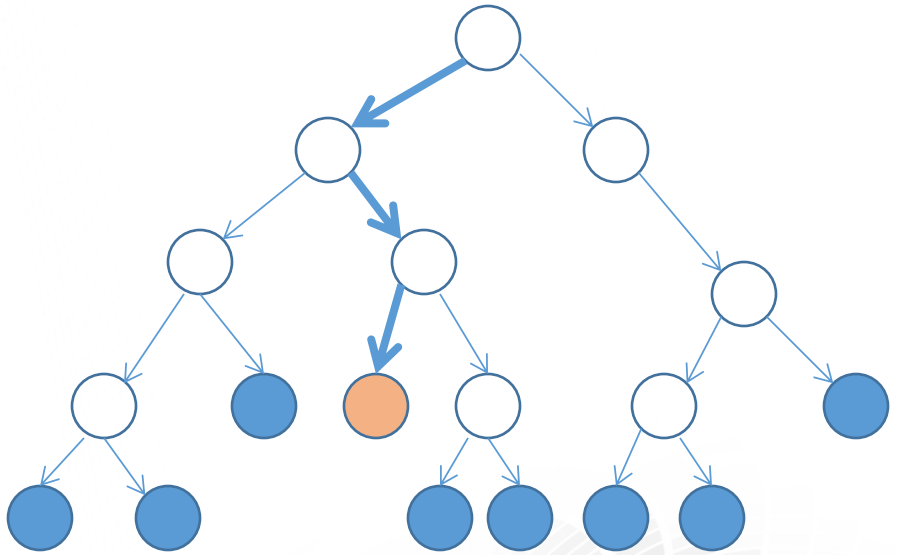
Ancestor Adjustment

- Objective move isn't entirely the result of most recent branch variable
  - Depends on ancestors as well
  - Particularly true for infeasible nodes
- Adjust pseudo-costs for ancestor branches



Ancestor Adjustment

- Need an adjustment strategy
- Empirically, adjustment should decrease as you move up the tree
- Our approach:
  - Exponential backoff
  - ½ for parent, ¼ for grandparent, etc.



Questions?

Less-Than Reductions

- As greater-than constraint (model ANDY3):
  - 31 b'1 + 28 b'2 + 20 b'3 + 13 b'4 + 11 b'5 ≥ 63
- As less-than constraint:
  - 31 b1 + 28 b2 + 20 b3 + 13 b4 + 11 b5 ≤ 40
- A bit more subtle:
  - 40 b1 + 28 b2 + 20 b3 + 13 b4 + 11 b5 ≤ 40
- Useful to consider both senses of constraint

MIR Inequality

- Given a'x ≤ b, compute MIR cut m'x ≤ n
- Domination check:
  - Solve a pair of one-constraint LPs
    - m0 = max{a'x subject to m'x ≤ n}
    - m1 = max{m'x subject to a'x ≤ b}
  - If m0 ≤ b and m1 > n, MIR inequality dominates original
- Performance impact (5 scalings):
  - On a set of 2444 test models
  - >1s: 3.5% slower
  - >10s: 4.1% slower

MIR Inequality

- Main value of MIR approach…
  - Help to identify missed special cases
- Example:
  - Most common MIR reduction on continuous variables:
    - $y + b1 + b2 \leq 1$
    - $0 \leq y \leq u << 1$
  - Easy to catch as a special case