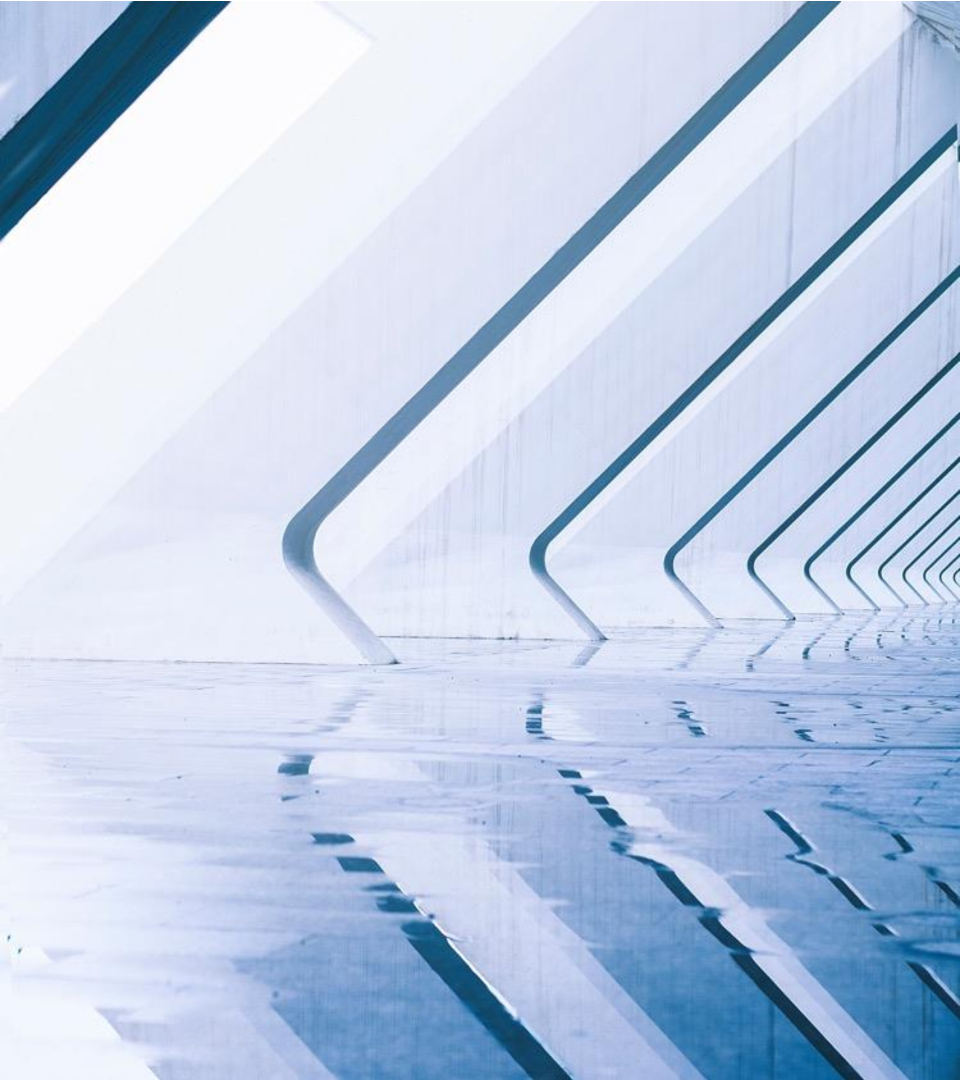


Branching

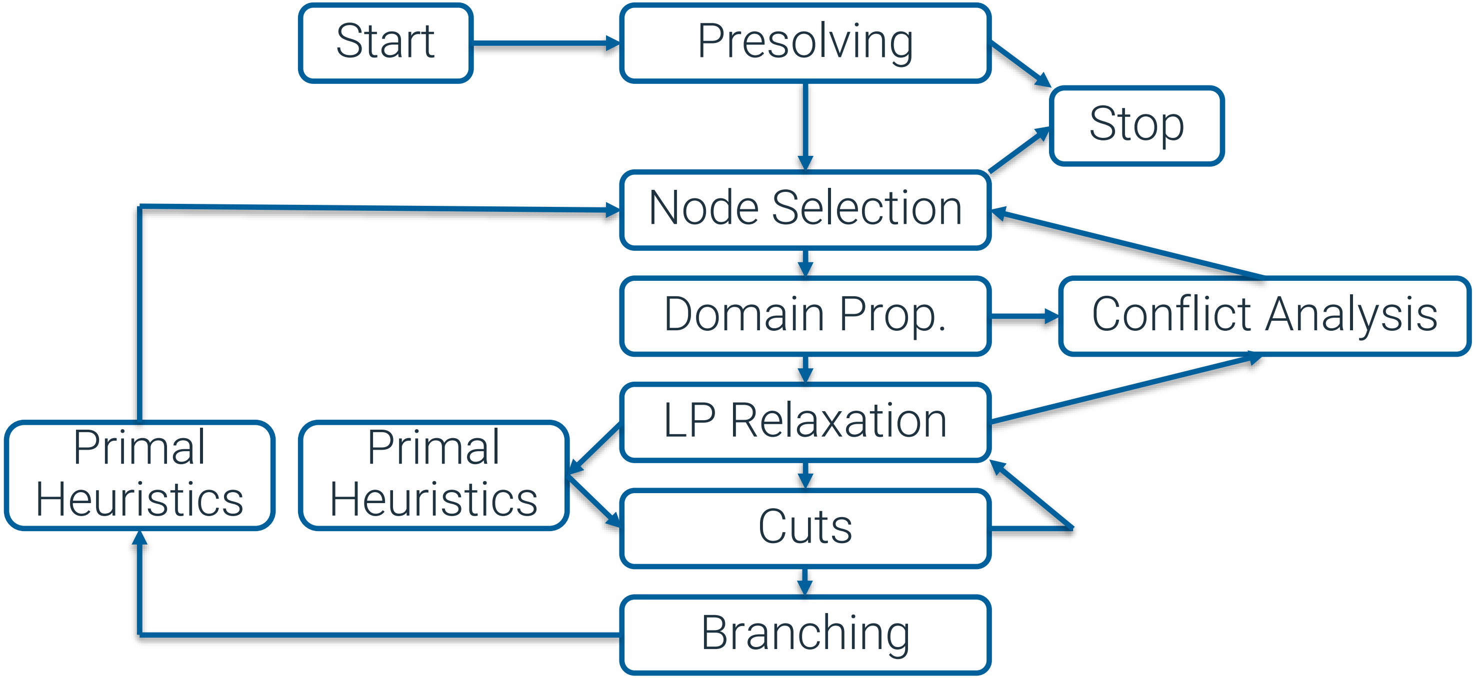
The MIP solver‘s backbone

Timo Berthold

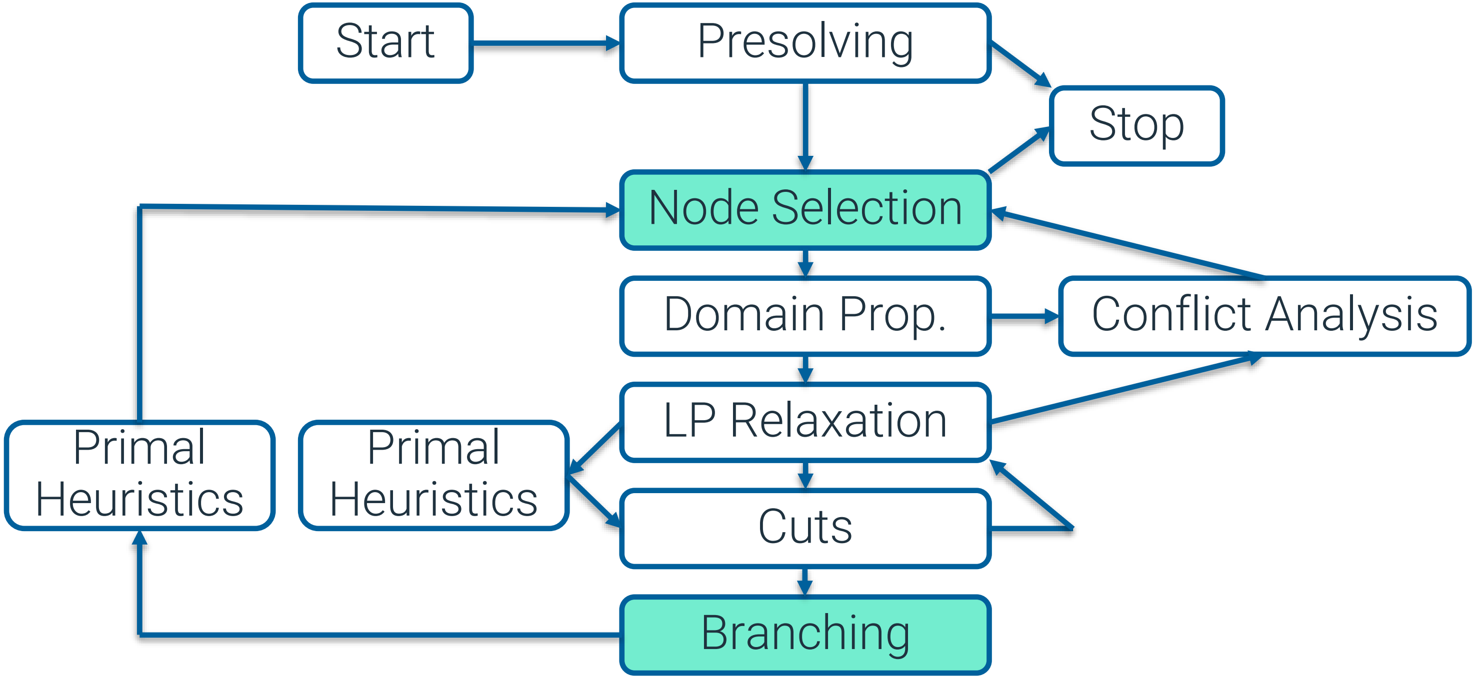
Agenda

- Branching
- Strong Branching, pseudo-costs, reliability
- Hybrid Branching, Cloud branching
- Node selection

MIP Solver Flowchart



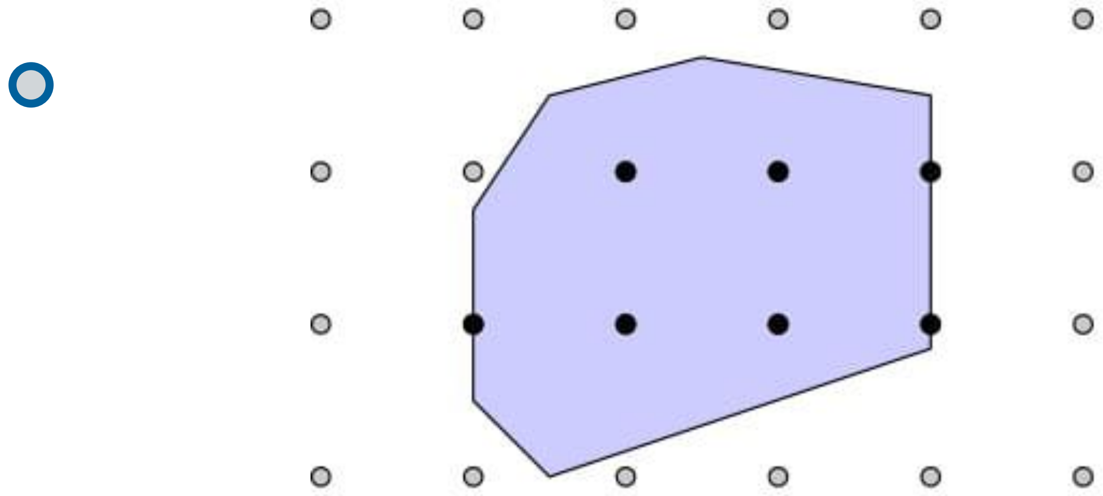
MIP Solver Flowchart



Reminder: Branch&Bound

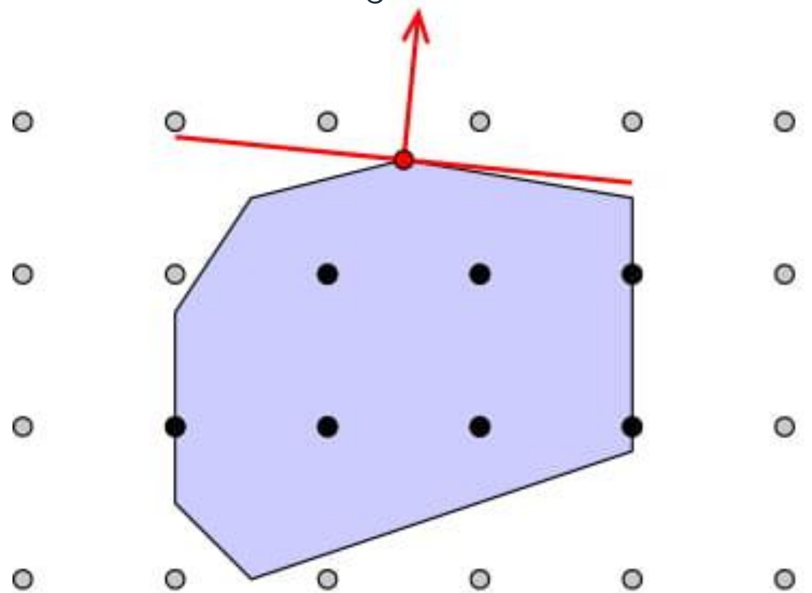
LP-based Branch&Bound (colorful picture)

1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching



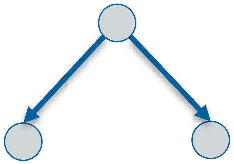
LP-based Branch&Bound (colorful picture)

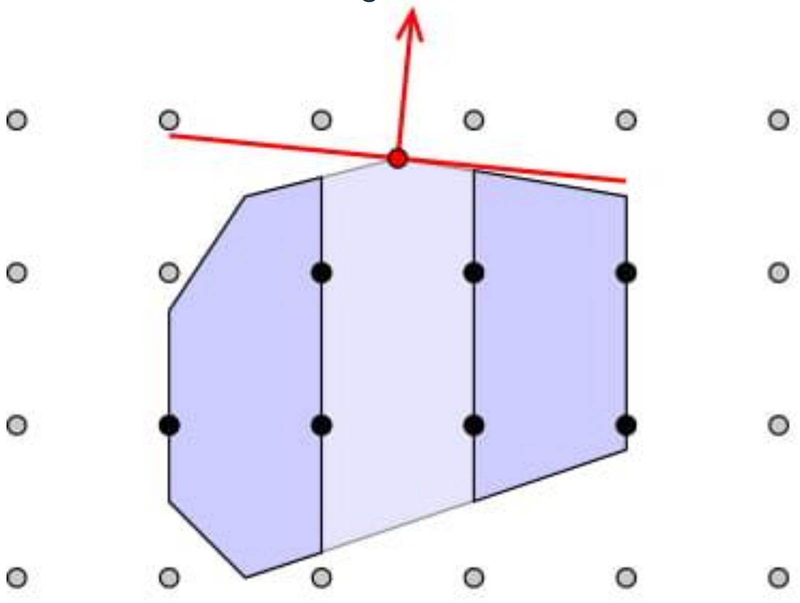
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching



LP-based Branch&Bound (colorful picture)

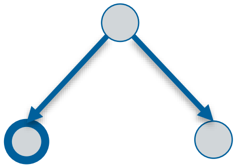
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

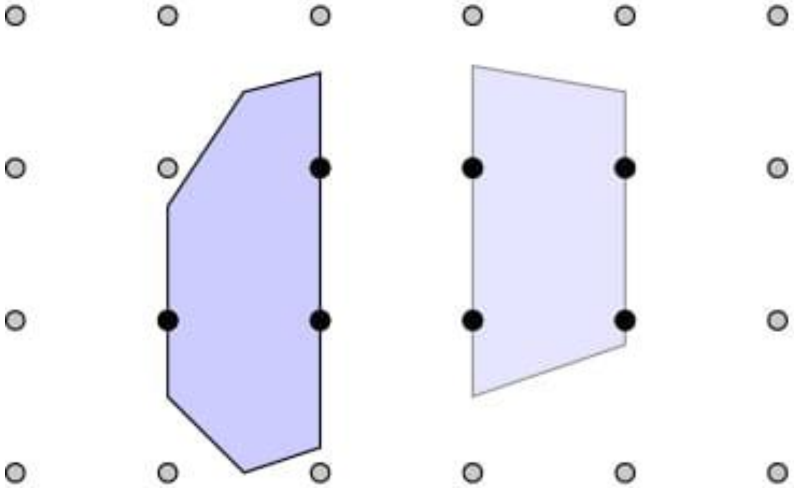




LP-based Branch&Bound (colorful picture)

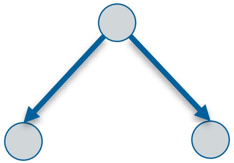
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

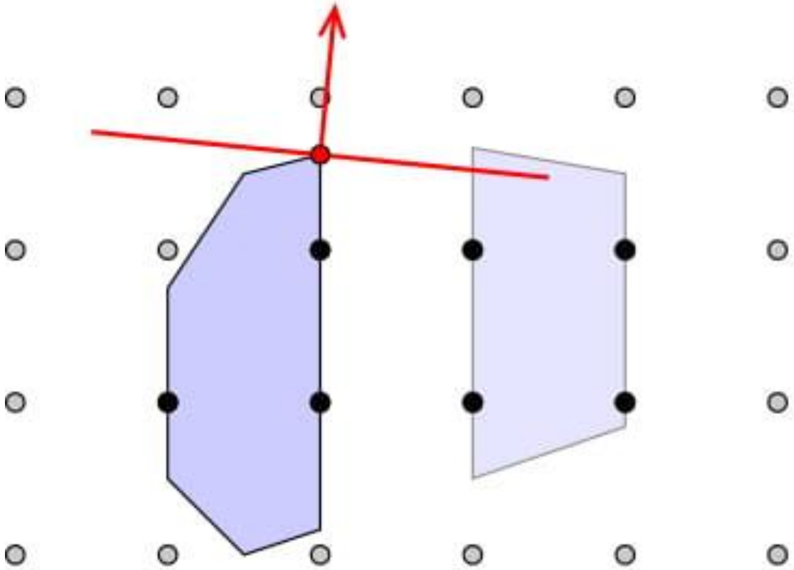




LP-based Branch&Bound (colorful picture)

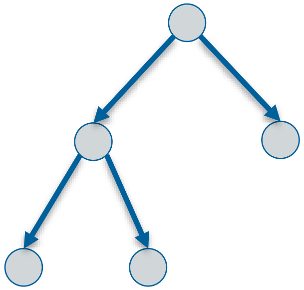
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

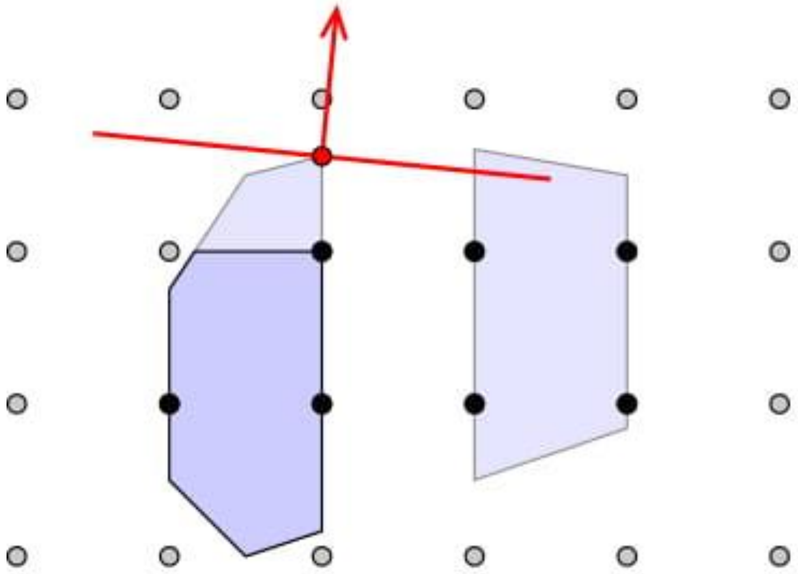




LP-based Branch&Bound (colorful picture)

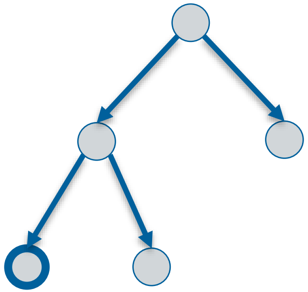
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

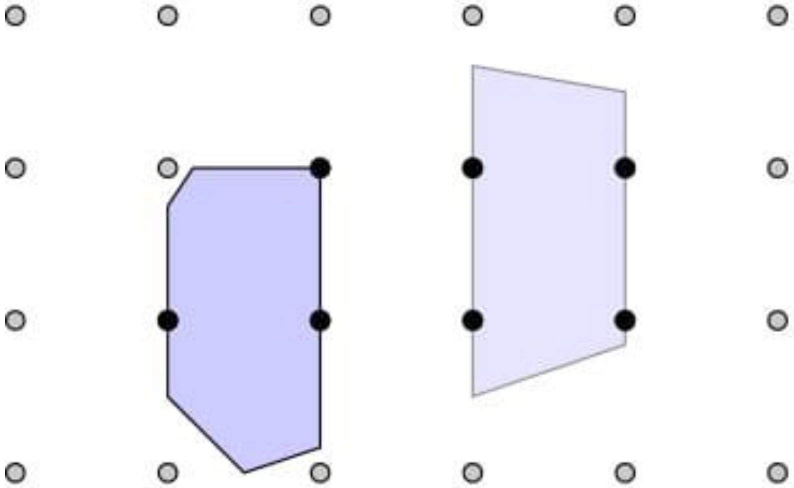




LP-based Branch&Bound (colorful picture)

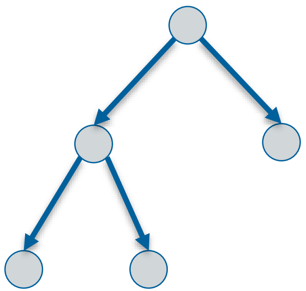
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

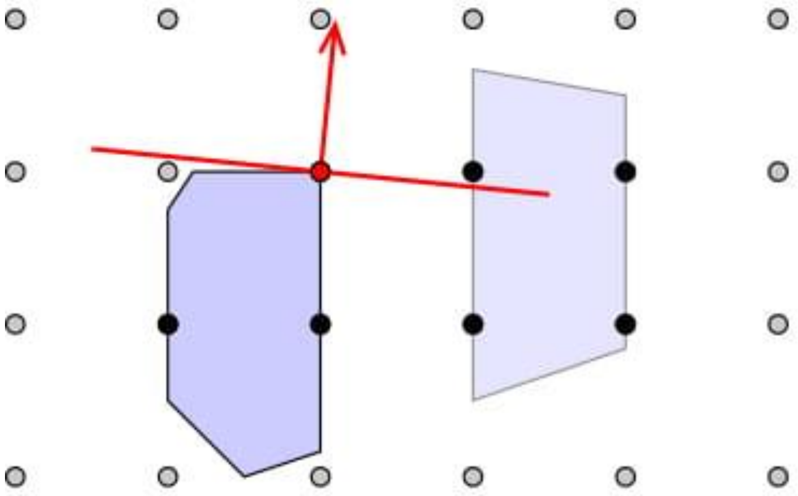




LP-based Branch&Bound (colorful picture)

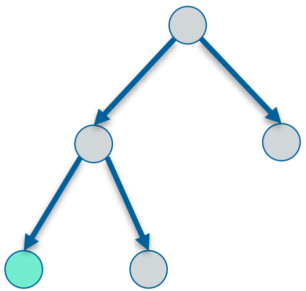
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

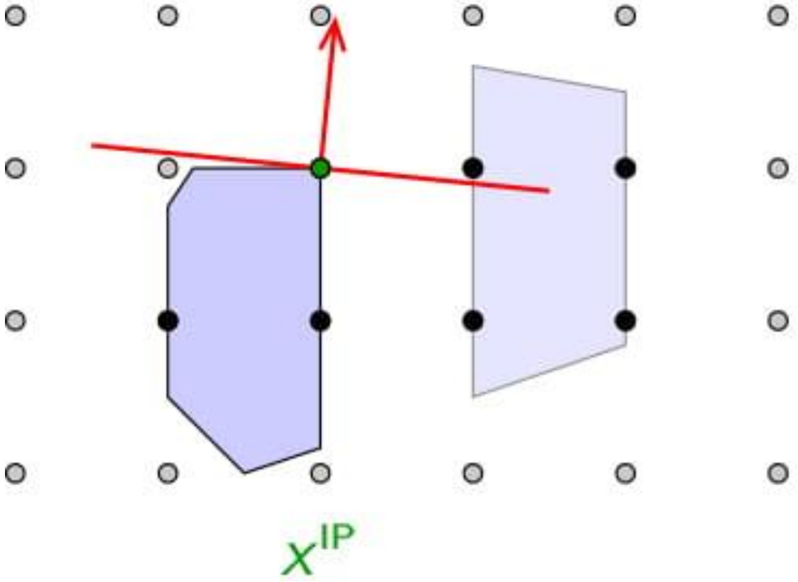




LP-based Branch&Bound (colorful picture)

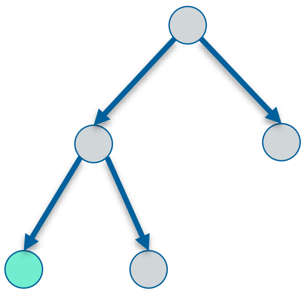
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

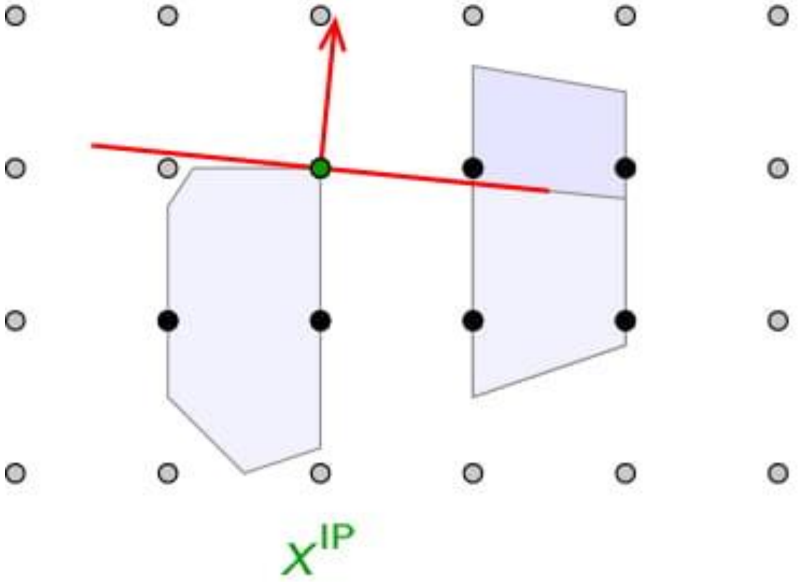




LP-based Branch&Bound (colorful picture)

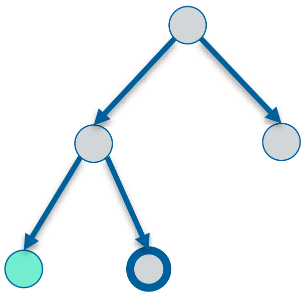
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

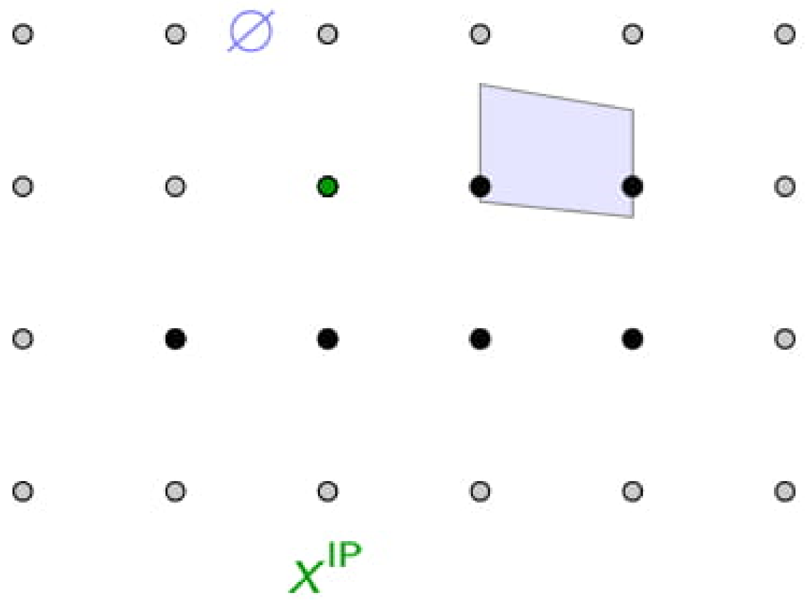




LP-based Branch&Bound (colorful picture)

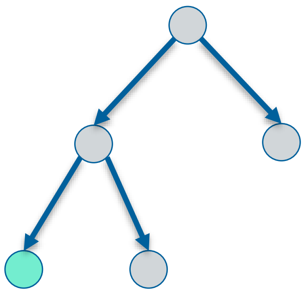
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

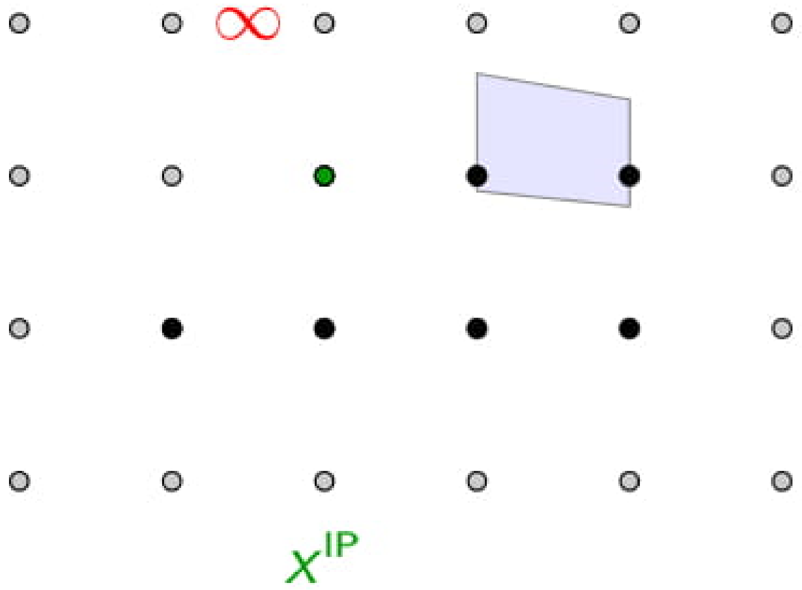




LP-based Branch&Bound (colorful picture)

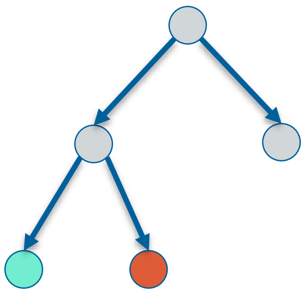
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

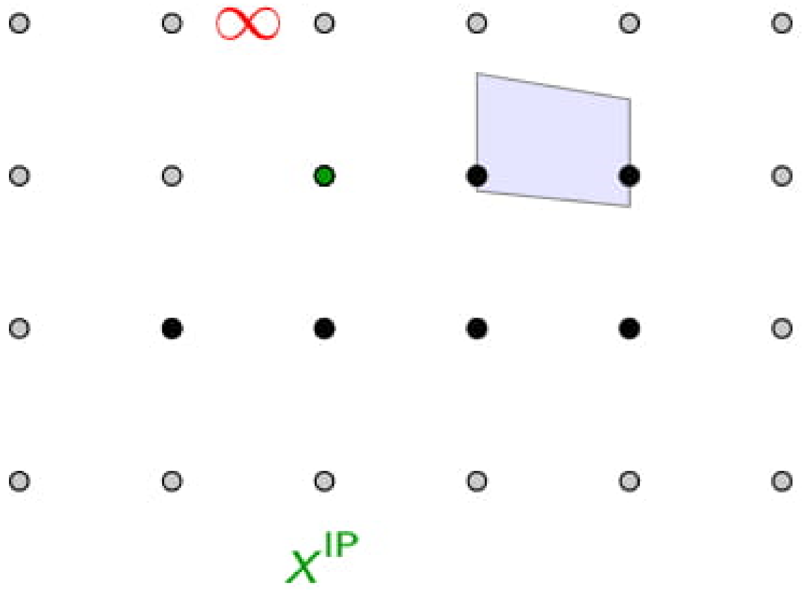




LP-based Branch&Bound (colorful picture)

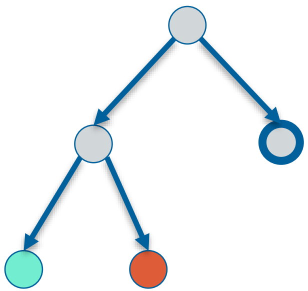
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

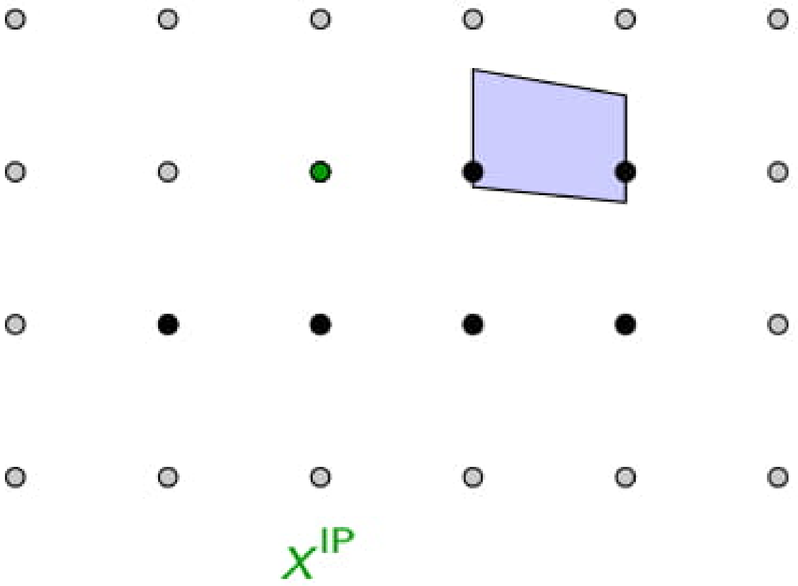




LP-based Branch&Bound (colorful picture)

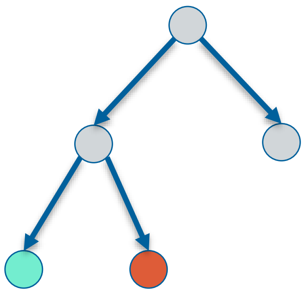
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

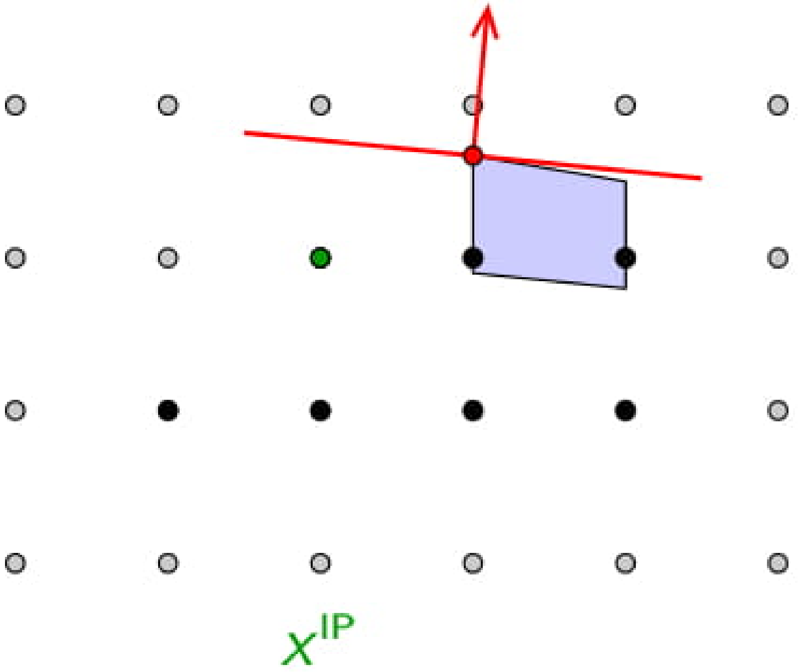




LP-based Branch&Bound (colorful picture)

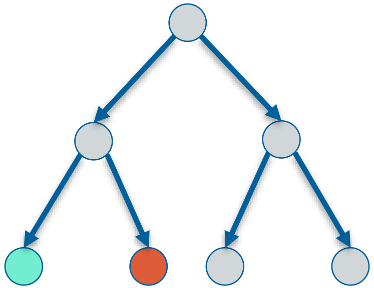
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

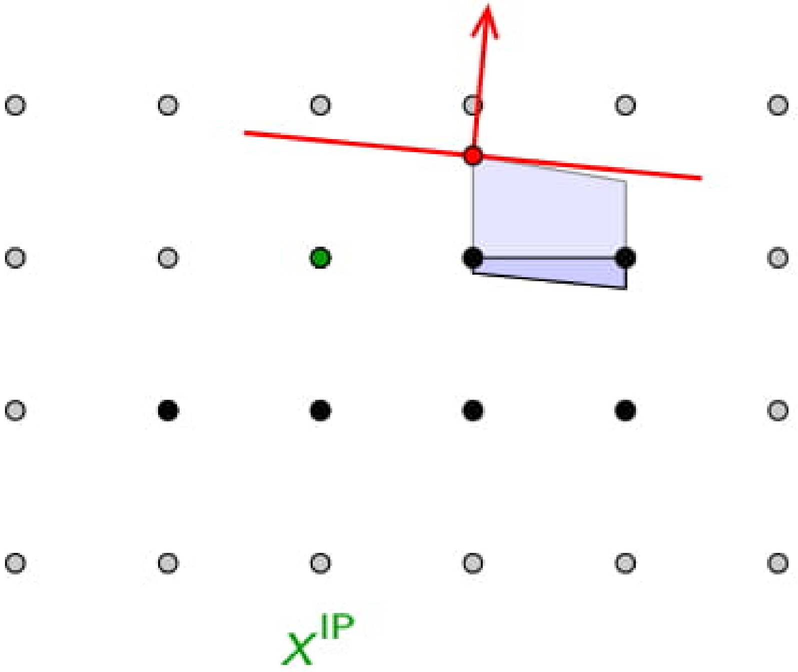




LP-based Branch&Bound (colorful picture)

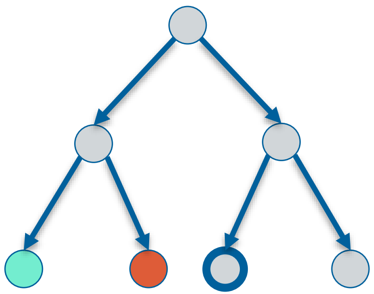
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

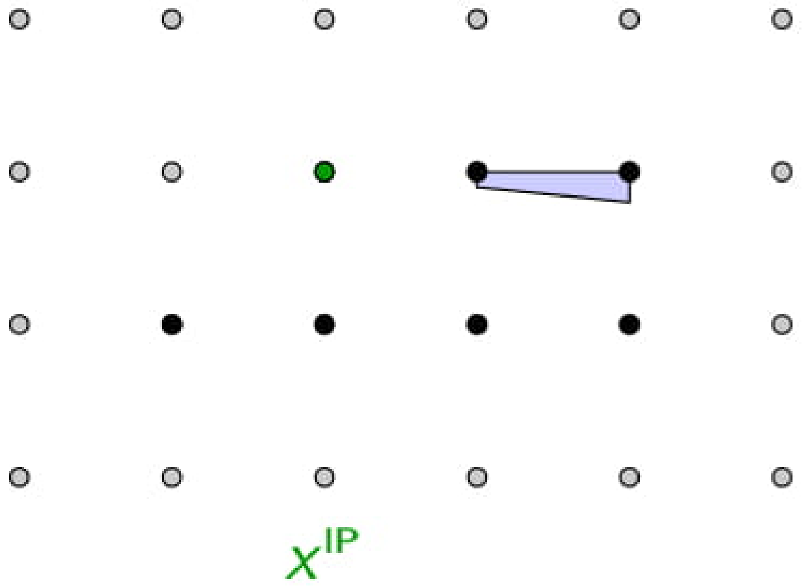




LP-based Branch&Bound (colorful picture)

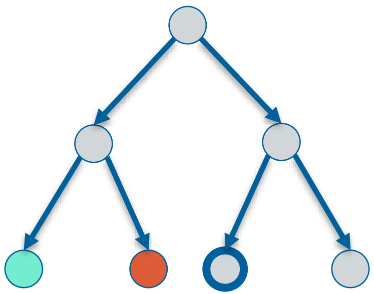
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

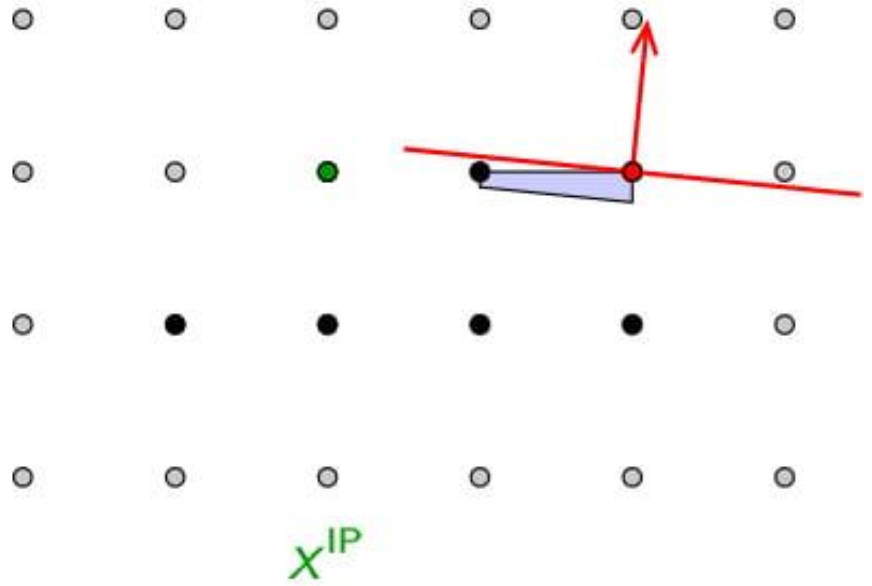




LP-based Branch&Bound (colorful picture)

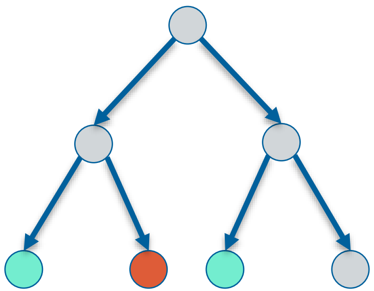
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

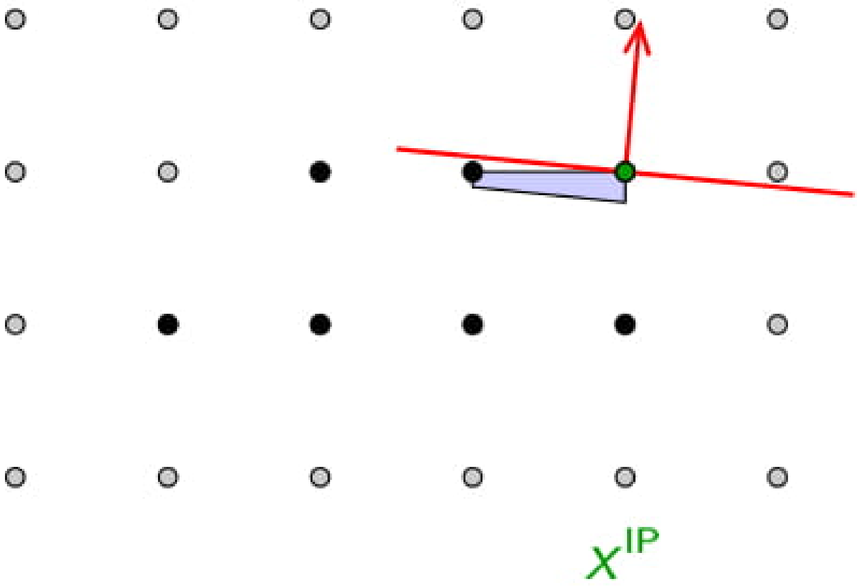




LP-based Branch&Bound (colorful picture)

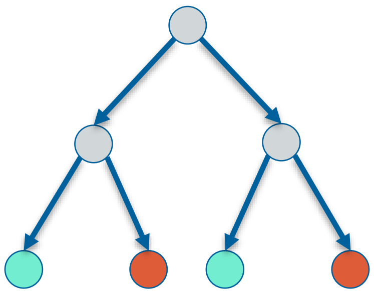
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

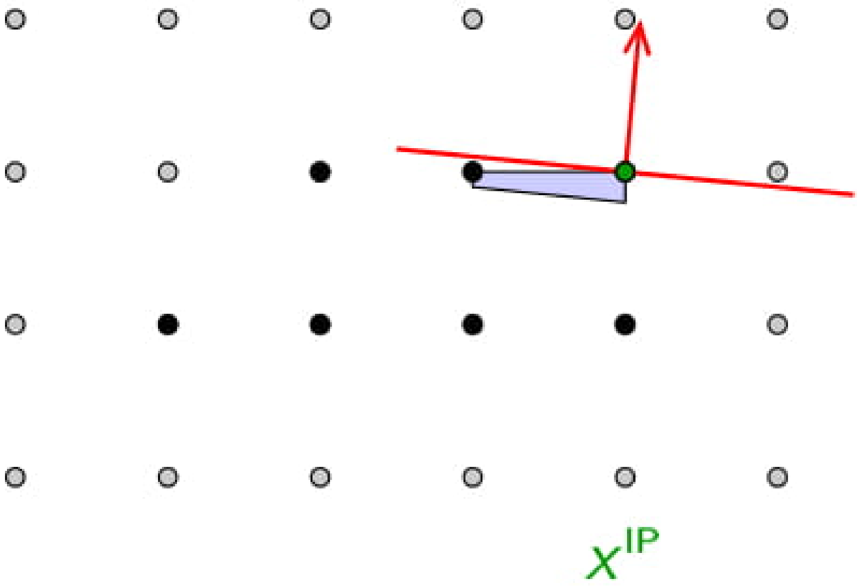




LP-based Branch&Bound (colorful picture)

1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching





LP-based Branch&Bound (colorful picture)

1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Variable selection

Two main decisions:

- Node selection
  - Might be important to find good solutions early
  - When optimum is found: just a matter of traversal order
- Variable selection
  - Bad selection might duplicate search effort
  - at every level….

Strong branching and pseudo-costs

Strong branching (Applegate et al 1995)

Typical goal: Improve dual bound

- Perform an explicit look-ahead by solving all possible descendants of the current node.

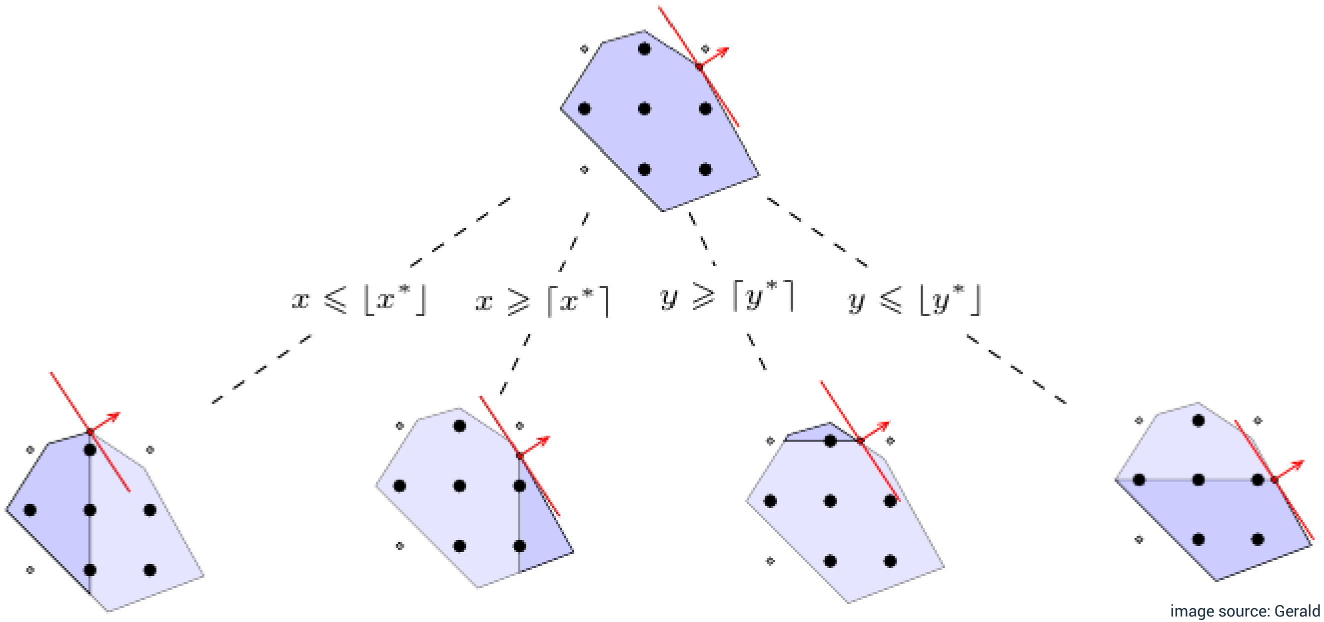


image source: Gerald Gamrath

Strong branching (Applegate et al 1995)

- Effective w.r.t. number of nodes, expensive w.r.t. time
- Strong branching might:
  - Fix variable, when one side is infeasible
  - Detect infeasibility, when both sides are infeasible
  - Find feasible solutions

Speeding strong branching up:

- Only for some candidates, stop if you do not make enough improvement
- Limit number of simplex iterations
- Special case: One iteration → Driebeek penalties (Driebeek 1966)
  - Can be efficiently computed by ratio test

Strong branching + domain propagation (Gamrath 2014)

- Some strong branching LPs further restricted by domain propagation
  - Add branching bound → perform "default" domain propagation → solve LP
- Better predictions, more fixings
- Only domain propagation, no LP:
  - Branching by probing

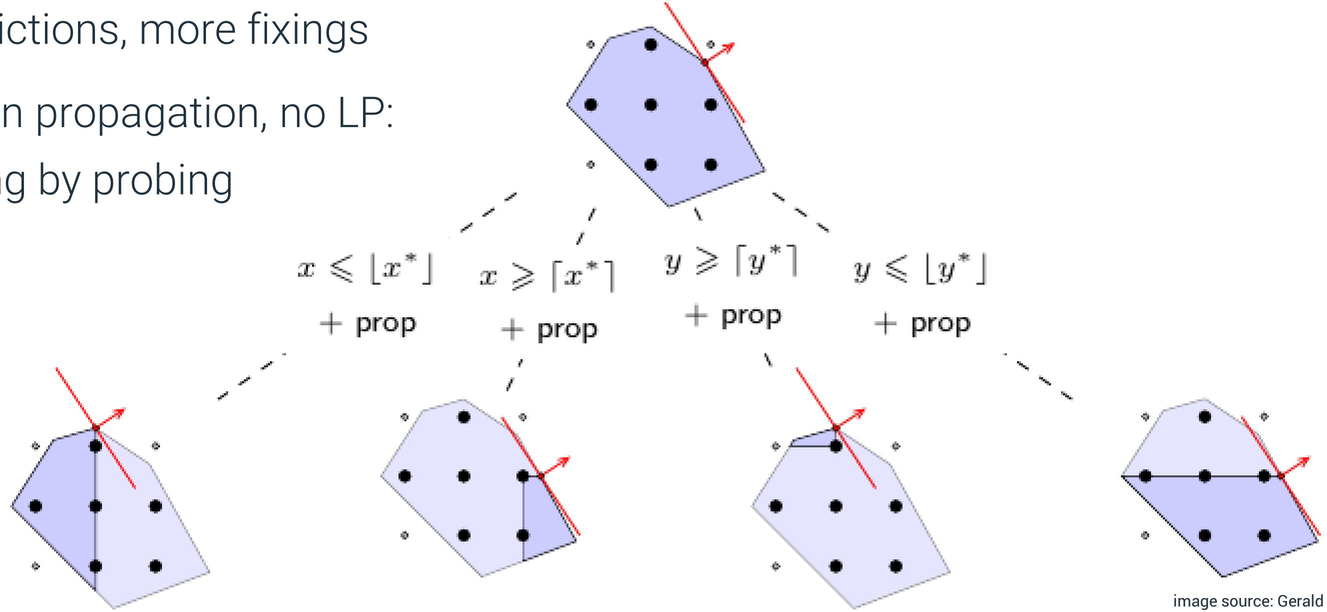
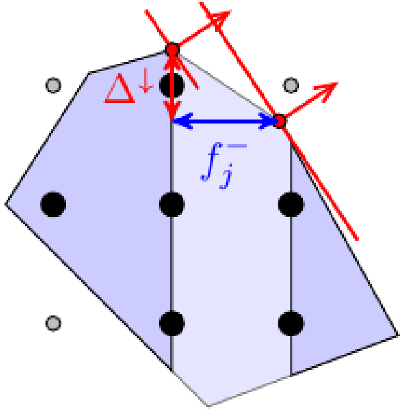
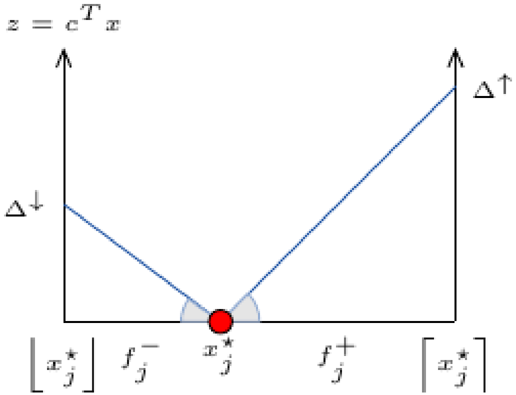


image source: Gerald Gamrath

Pseudo-costs (Bénichou 1971)

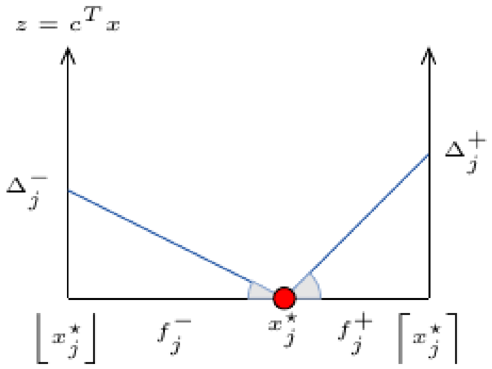
- Strong branching: A-priori observation, pseudo-costs: a-posteriori
- Estimate for objective gain based on past branching observations.
- Objective gain per unit fractionality:
  computed from fractionalities f_j^-, f_j^+ and
  differences $\Delta^\downarrow, \Delta^\uparrow$ in LP values
- Pseudo-costs Ψ_j^-, Ψ_j^+: average unit gain taken over all nodes that branched on same variable





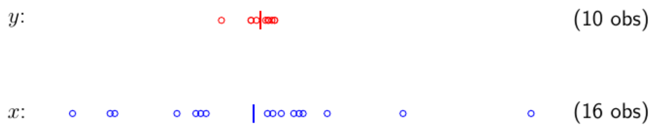
Pseudo-cost branching

- Estimated increase of objective $\Delta_j^- = f_j^- \Psi_j^-, \Delta_j^+ = f_j^+ \Psi_j^+$ based on current fractionalities f_j^-, f_j^+
- Core of most state-of-the-art branching schemes
- Gets better and better during the search
- Values might show a large variance
- Attributes all change to the last branching



Reliability branching (Achterberg et al 2005)

- Pseudo-cost branching gets better and better during the search
  - Most important branchings are made in the beginning
- Standard approach: Pseudo-cost branching with strong branching initalization
- Even better: consider variable unreliable, as long as there are less than k strong branches
  - Typical values for k: 4-8
  - k might depend on variance of pseudo-cost values



- Should a strong branch that hit the iteration limit be considered reliable?
- Should we reconsider strong branching when some subproblem behaves „differently“?

Quiz time

- Pseudo-costs are an
  a) Underestimator for the objective change when pivoting
  b) Underestimator of the objective change when relaxing a constraint
  c) Estimate of the objective change when branching
- Strong Branching is very competitive w.r.t. the
  a) Running time
  b) Number of nodes
  c) Primal-dual integral

Quiz time

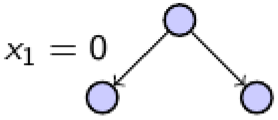
- Pseudo-costs are an
  a) Underestimator for the objective change when pivoting
  b) Underestimator of the objective change when relaxing a constraint
  c) Estimate of the objective change when branching
- Strong Branching is very competitive w.r.t. the
  a) Running time
  b) Number of nodes
  c) Primal-dual integral

Hybrid Branching

Inference branching

$$
\begin{aligned}
x_1 \qquad\quad + x_2 &= 1 \\
x_1 + x_3 + x_4 &\leq 1 \\
x_1 \qquad\quad + z &\geq 3 \\
z &\in \mathbb{Z}_+ \\
x_i &\in \{0, 1\}
\end{aligned}
$$

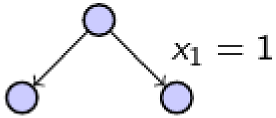
- Inference branching:
  - Average number of implied bound reductions
  - History based
  - Captures combinatorial structure
  - Estimates tightening of subproblems



$$
\begin{aligned}
x_1 = 0 \Rightarrow x_2 &= 1 \\
\Rightarrow z &\geq 3
\end{aligned}
$$

$$s_j^{\text{infer}}(-) = 2$$

- Analogy to pseudo-cost values in MIP
- One value for upwards branch, one for downwards
- Initialization: probing (≈ strong branching)



$$
\begin{aligned}
x_1 = 1 \Rightarrow x_2 &= 0 \\
\Rightarrow x_3 &= 0 \\
\Rightarrow x_4 &= 0
\end{aligned}
$$

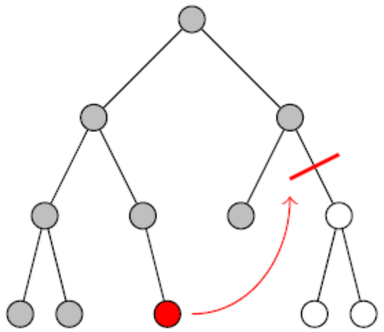
$$s_j^{\text{infer}}(+) = 3$$

FICO

VSIDS branching (Moskewicz et al 2001)

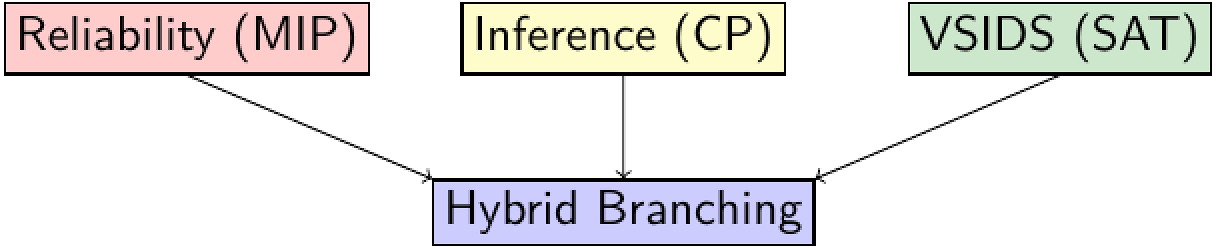
Conflict analysis:

- Learn additional constraints which trigger infeasibility
- Important for feasibility problems



- VSIDS branching:
  - Variable which appears in highest number of (conflict) clauses
    - Branch towards infeasibility
  - Prefer “recent” conflicts: exponentially decreasing importance
  - Works particularly well for feasibility problems
  - State-of-the-art in SAT solving

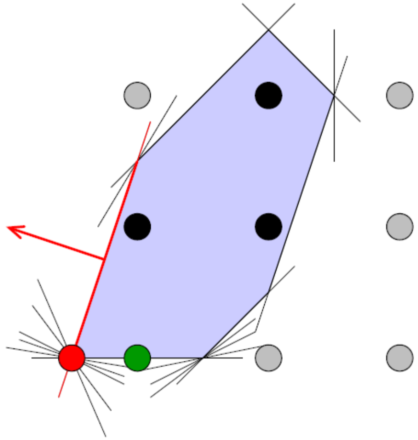
Hybrid branching (Achterberg and Berthold 2009)



- Additional tie-breakers: number of pruned subproblems, variable counts in Farkas proofs, ...
- Scaling: divide each value by average over all variables
- Use a weighted sum of all criteria
- Or: Use a leveled filtering approach. First filter leaves 100 candidates, second filter 10,...

A cloud of solutions (Berthold & Salvagnin 2013)

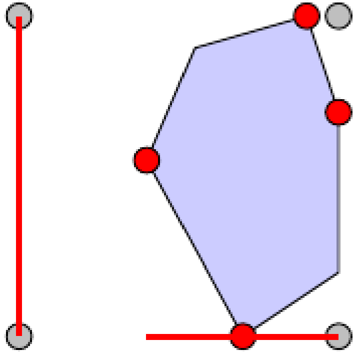
- Often many optimal LP solutions (an optimal polyhedron)
- “The” optimal LP solution is more or less random
- Idea: exploit knowledge of multiple (a cloud of) LP optima



How do we get extra optimal solutions?

- Restrict LP to optimal face
- Feasibility pump objective (pump-reduce)
- min/max each variable (OBBT)
- → Intervals instead of single values

$$x_1 \in [0.4, 1]$$
$$x_2 \in [0, 1]$$



Cloud-based pseudo-costs

- Pseudo-cost update

$$\varsigma_j^+ = \frac{\Delta^\uparrow}{\lceil x_j^\star \rceil - x_j^\star} \quad \ldots\text{better: } \tilde{\varsigma}_j^+ = \frac{\Delta^\uparrow}{\lceil x_j^\star \rceil - u_j}$$

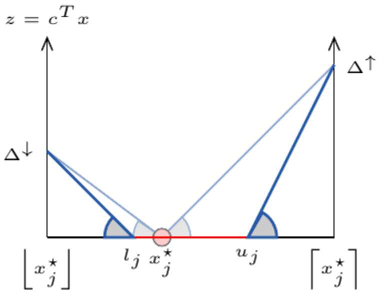
- Pseudo-cost-based estimation

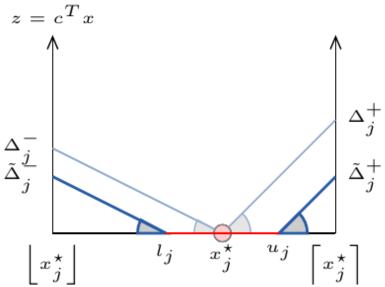
$$\Delta_j^+ = \Psi_j^+(\lceil x_j^\star \rceil - x_j^\star) \quad \ldots\text{better: } \tilde{\Delta}_j^+ = \Psi_j^+(\lceil x_j^\star \rceil - u_j)$$

Lemma

Let $x^\star$ be an optimal solution of the LP relaxation at a given branch-and-bound node and $\lfloor x_j^\star \rfloor \leq l_j \leq x_j^\star \leq u_j \leq \lceil x_j^\star \rceil$. Then

1. for fixed $\Delta^\uparrow$ and $\Delta^\downarrow$, it holds that $\tilde{\varsigma}_j^+ \geq \varsigma_j^+$ and $\tilde{\varsigma}_j^- \geq \varsigma_j^-$, respectively;
2. for fixed Ψ_j^+ and Ψ_j^-, it holds that $\tilde{\Delta}_j^+ \leq \Delta_j^+$ and $\tilde{\Delta}_j^- \leq \Delta_j^-$, respectively.





Cloud-based strong branching

Benefit of cloud intervals:

- Fractional variable gets integral in cloud point: one LP spared!
- Cloud branching acts as a filter
- New fractional variables → new candidates (one side known)
- Use 3-partition of branching candidates

Similar idea: Non-chimerical branching (Fischetti & Monaci 2012)

- Use values from other strong branches to compute underestimators

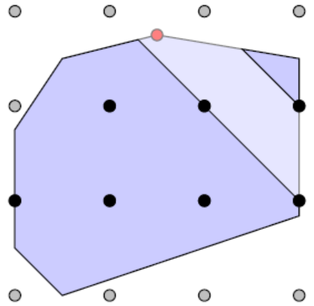
Branching score

- Most branching rules yield two values: One for down-, one for up-branch
- need to combine them to a single value
- usually: convex sum
  - $\text{score}(x_j) = \lambda \max\{s_j^-, s_j^+\} + (1 - \lambda) \min\{s_j^-, s_j^+\}$
  - traditionally $\lambda = 6$
  - includes minimum and maximum as extreme cases
- better: multiplication
  - $\text{score}(x_j) = \max\{s_j^-, s_j^+\} \cdot \min\{s_j^-, s_j^+\}$
  - computational results: 10% faster

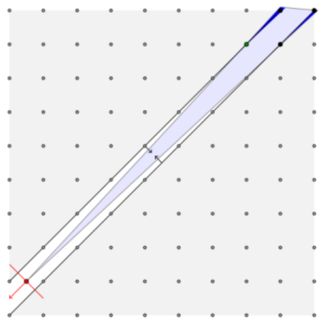
Branching on general disjunctions

$$\pi^T x \le \pi_0 \quad \bigvee \quad \pi^T x \ge \pi_0 + 1$$

with $(\pi, \pi_0) \in \mathbb{Z}^n \times \mathbb{Z}$, and $\pi_i = 0$ for all $i \notin \mathcal{I}$.



- potentially better branching decisions
- choosing the best candidate computationally much more expensive
- no generic scheme improving the overall MIP performance
  - Xpress branches on general disjunctions in some cases



Branching on multi-aggregated variables (Gamrath et al 2015)

- Some variables get multi-aggregated in presolving $x_j = \beta + \sum_{j \in S} \alpha_j x_j$
- multi-aggregated variables not part of presolved problem
  - not used as branching candidates
- branch on corresponding general disjunctions
  - extend variable-based branching by these disjunctions

$$\sum_{j \in \mathcal{S}} \alpha_j x_j \geq \left\lceil \sum_{j \in \mathcal{S}} \alpha_j \tilde{x}_j \right\rceil \quad \bigvee \quad \sum_{j \in \mathcal{S}} \alpha_j x_j \leq \left\lfloor \sum_{j \in \mathcal{S}} \alpha_j \tilde{x}_j \right\rfloor$$

- represents decisions in original problem
- moderately enlarged candidate set

Quiz time

- Strong Branching + Pseudocost Branching =
  a) Cloud Branching
  b) Reliability Branching
  c) Inference Branching
- Cloud branching makes use of
  a) Multiple LP optima
  b) Multiple integer solutions
  c) A combination of LP optima and integer solutions

Quiz time

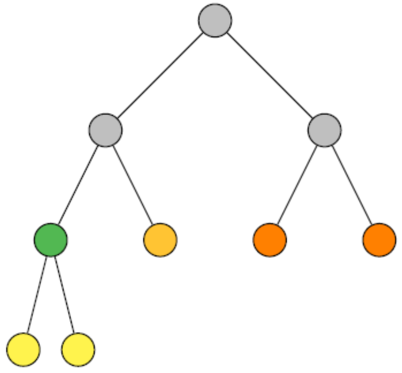
- Strong Branching + Pseudocost Branching =
  a) Cloud Branching
  b) Reliability Branching
  c) Inference Branching
- Cloud branching makes use of
  a) Multiple LP optima
  b) Multiple integer solutions
  c) A combination of LP optima and integer solutions

Node Selection

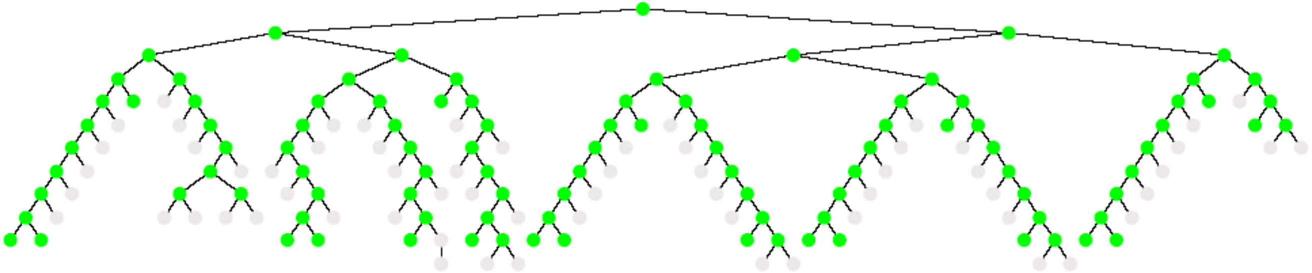
Considerations

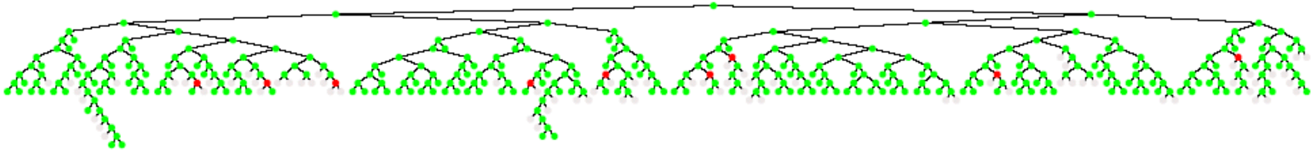
Goals:

- Improve primal bound to enable pruning
- Keep computational effort small
    - Prefer children over siblings over others
- Improve global dual bound
- Ramp-up
    - For parallelization



What do typical branching trees look like?





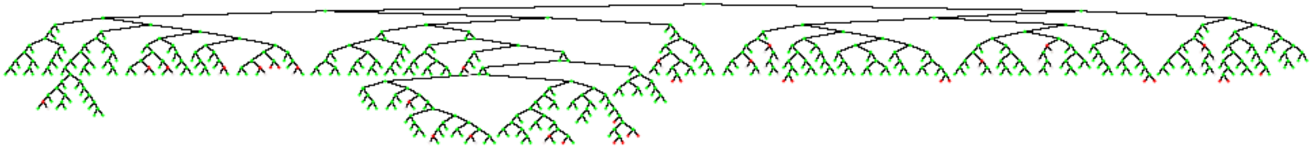


image source: Thorsten Koch

What do typical branching trees look like?

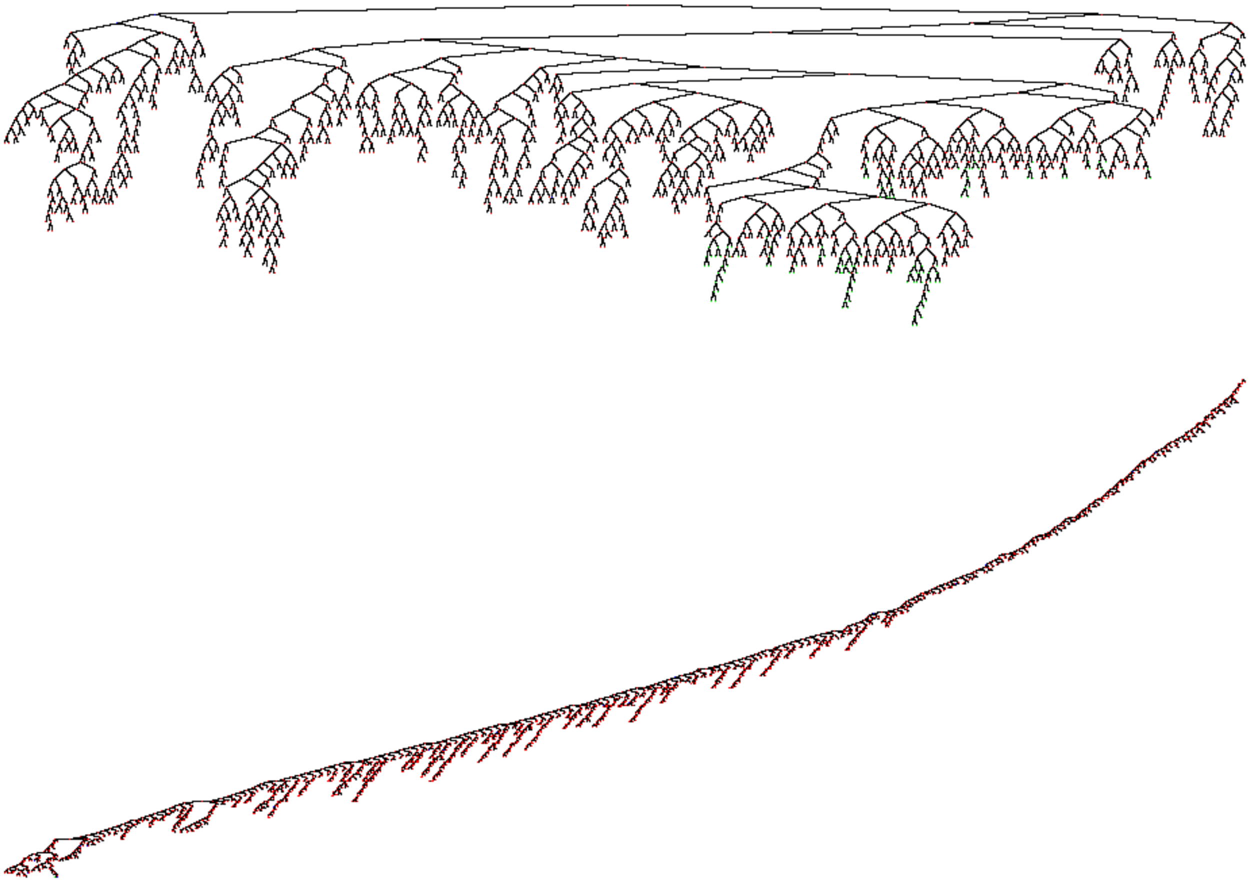
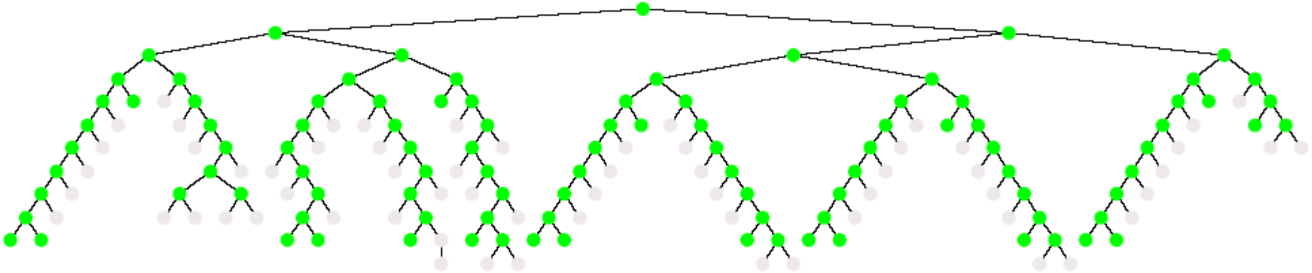
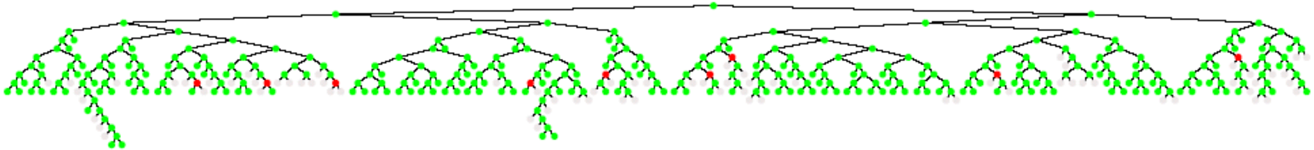


image source: Thorsten Koch

What do typical branching trees look like?





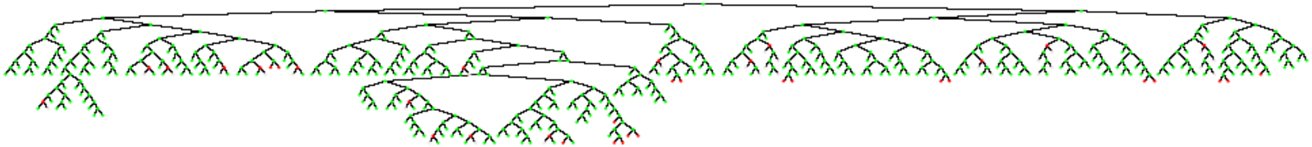
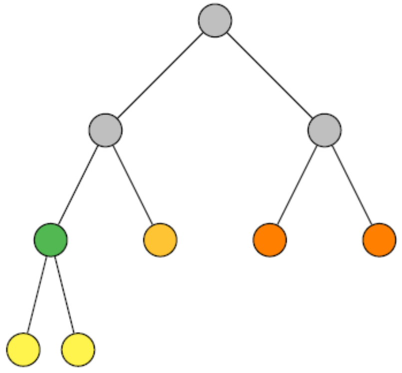


image source: Thorsten Koch

Node Selection Rules

Basic rules

- Depth first search (DFS) → early feasible solutions
  - Most of the time, MIP solvers do DFS
- Breadth first search (BFS) → diversification, ramp-up
- Best bound search (BBS) → improve dual bound
- Best estimate search (BES) → improve primal bound



Combinations:

- BBS or BES with plunging
- Hybrid BES/BBS / Interleaved BES/BBS

UCT node selection (Sabharwal et al 2012)

- Inspired by Monte-Carlo tree search
  - Chess, games, balancing exploration and exploitation
- Upper Confidence intervals applied to Trees
- „Which path to choose“: $s_j = E_j + c\frac{v_p}{v_j}$
  - E_j: estimate, v_p: parent visits, v_j: child visits, c: balancing
  - Estimate permanently updated, average dual bound in subtree
- Quickly gets expensive, only apply to first few nodes

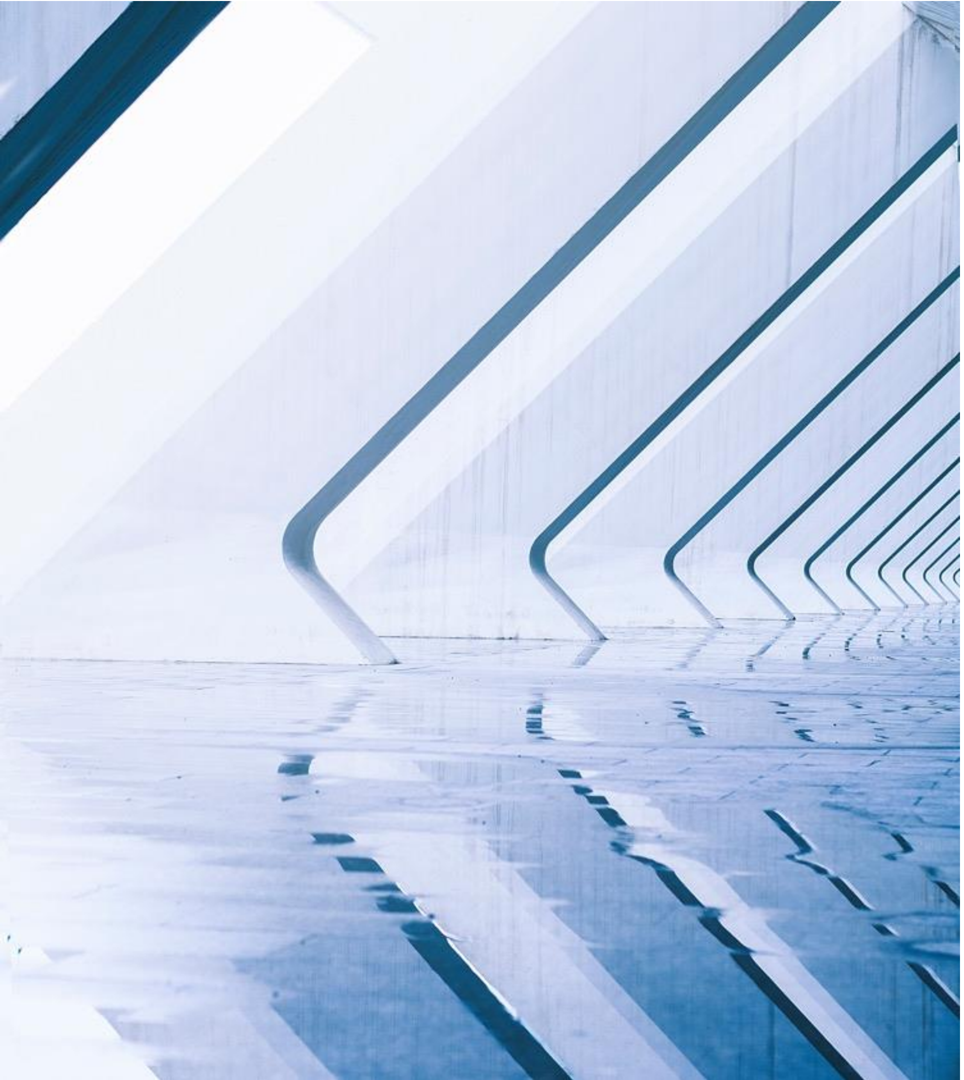
image source: pexels.com

Quiz time

- Most of the times, a MIP solver will select as next node
  a) A child or sibling of the current node
  b) A node close to the root
  c) A node with the best dual bound
- W.r.t. running time, node selection empirically has
  a) A larger impact than the branching rule
  b) A smaller impact than the branching rule
  c) About the same impact as the branching rule

Quiz time

- Most of the times, a MIP solver will select as next node
  a) A child or sibling of the current node
  b) A node close to the root
  c) A node with the best dual bound
- W.r.t. running time, node selection empirically has
  a) A larger impact than the branching rule
  b) A smaller impact than the branching rule
  c) About the same impact as the branching rule



FICO®

Thank You!

Timo Berthold