Branching

The backbone of the MIP solver

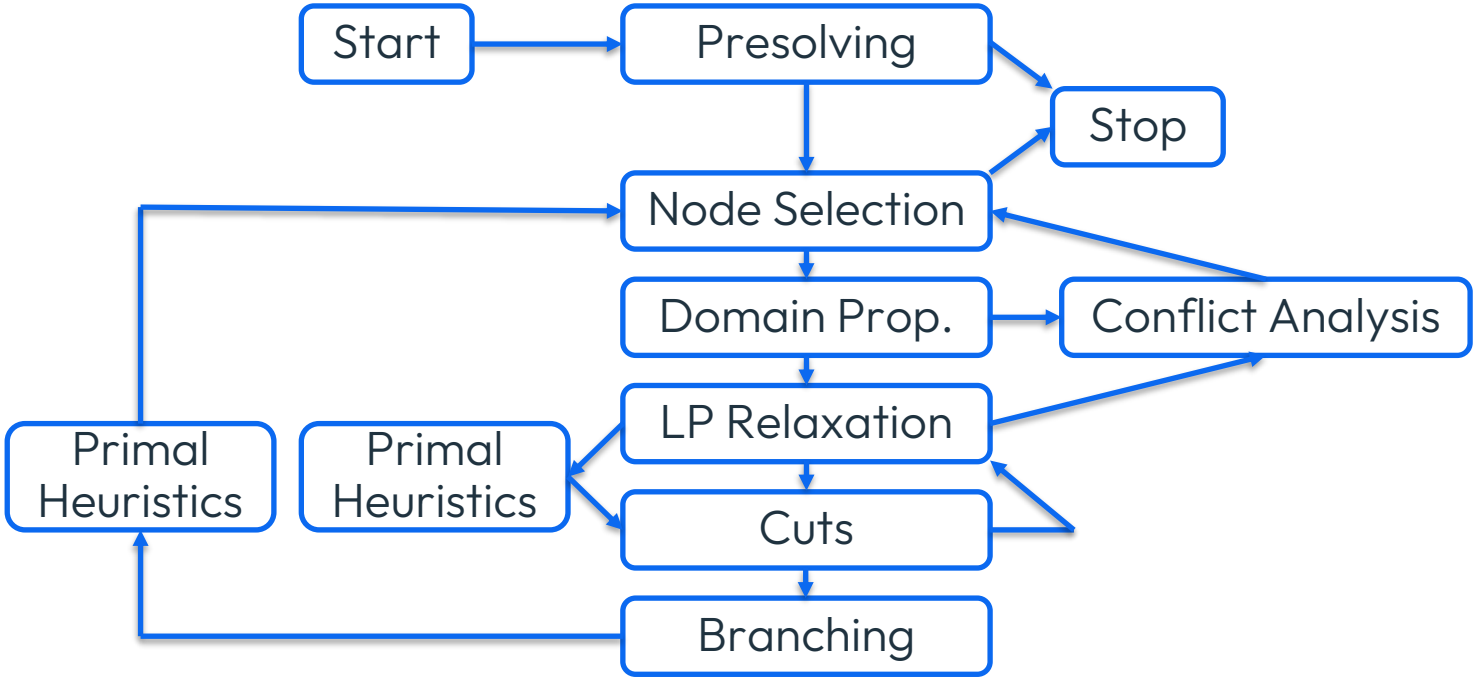
Timo Berthold
TU Berlin, FICO, MODAL

Computational Mixed Integer Programming

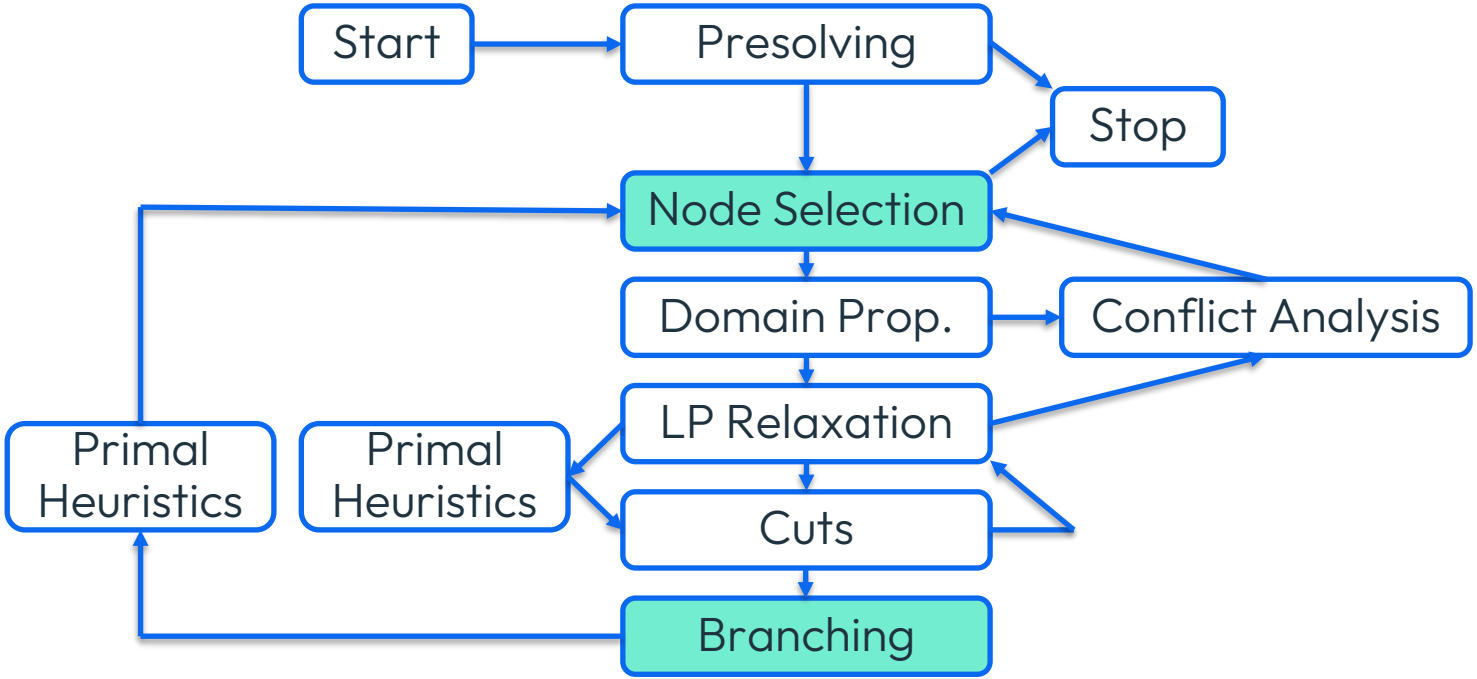
- Bob Bixby (godfather of Computational MIP):
  „MIP solvers are a bag of tricks!“
- In 32 years, hardware got a million times faster
- In 32 years, MIP algorithms got 6.5 million times faster [Bixby 2024]
- In recent years, hardware speedups have gone stale
- MIP solvers are still going strong
  - Xpress: ~20% speedup per year
- Let‘s take a look into the bag…

image source: DALL-E

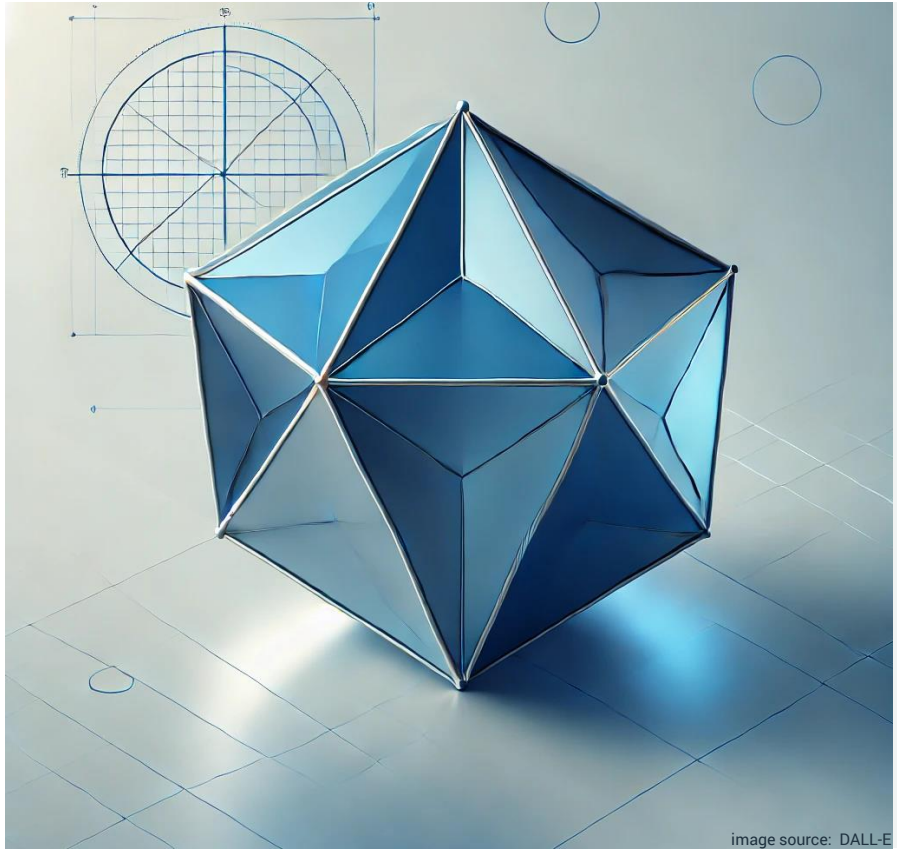
MIP Solver Flowchart



MIP Solver Flowchart



Agenda



1. Branching
2. Strong branching, pseudo-costs, reliability
3. Other history-based rules, hybrid branching
4. Node selection

Refresh: Branch&Bound

ECONOMETRICA

VOLUME 28 July, 1960 NUMBER 3

AN AUTOMATIC METHOD OF SOLVING DISCRETE PROGRAMMING PROBLEMS

BY A. H. LAND AND A. G. DOIG

In the classical linear programming problem the behaviour of continuous, nonnegative variables subject to a system of linear inequalities is investigated. One possible generalization of this problem is to relax the continuity condition on the variables. This paper presents a simple numerical algorithm for the solution of programming problems in which some or all of the variables can take only discrete values. The algorithm requires no special techniques beyond those used in ordinary linear programming, and lends itself to automatic computing. Its use is illustrated on two numerical examples.

First IFORS meeting: kickstart for LP-based Branch&Bound?

- From Bill Cook's fantastic IFORS distinguished lecture: https://www.youtube.com/watch?v=5VjphFYQKj8

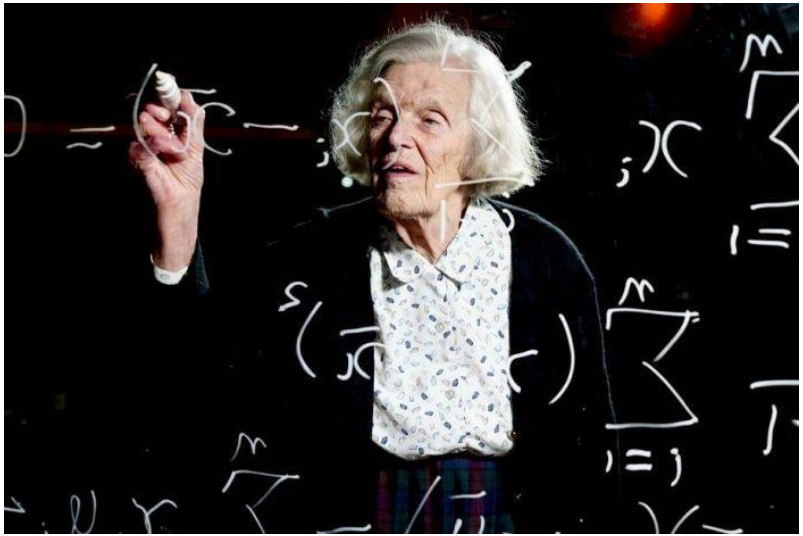


image source: abc.net.au

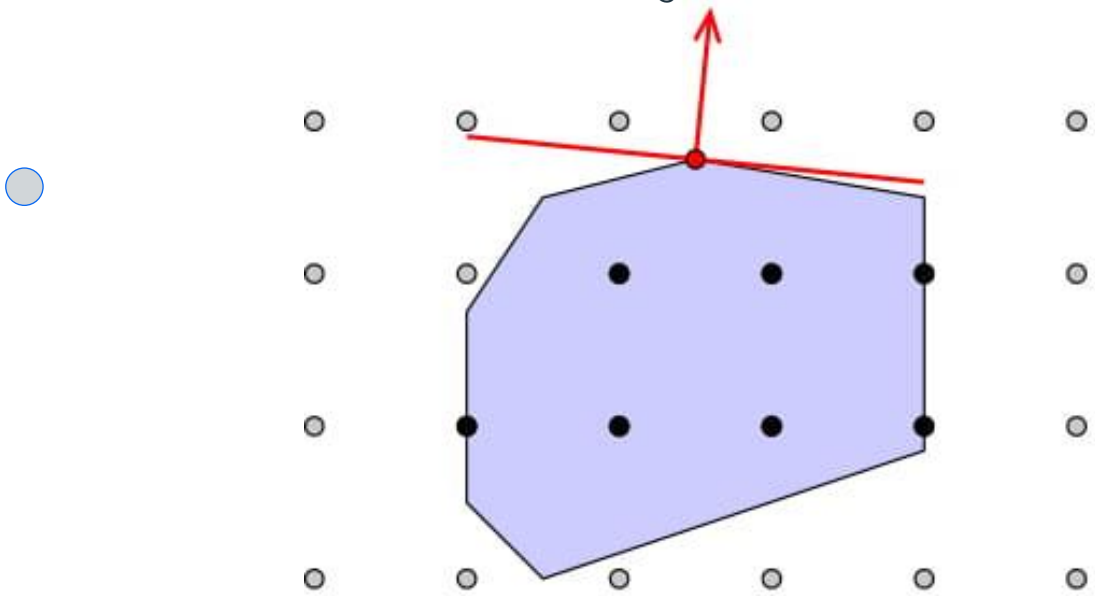
- Alison Harcourt recently appointed Senior Australian of the year 2019

LP-based Branch&Bound (colorful picture)

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

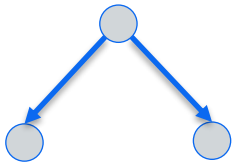
LP-based Branch&Bound (colorful picture)

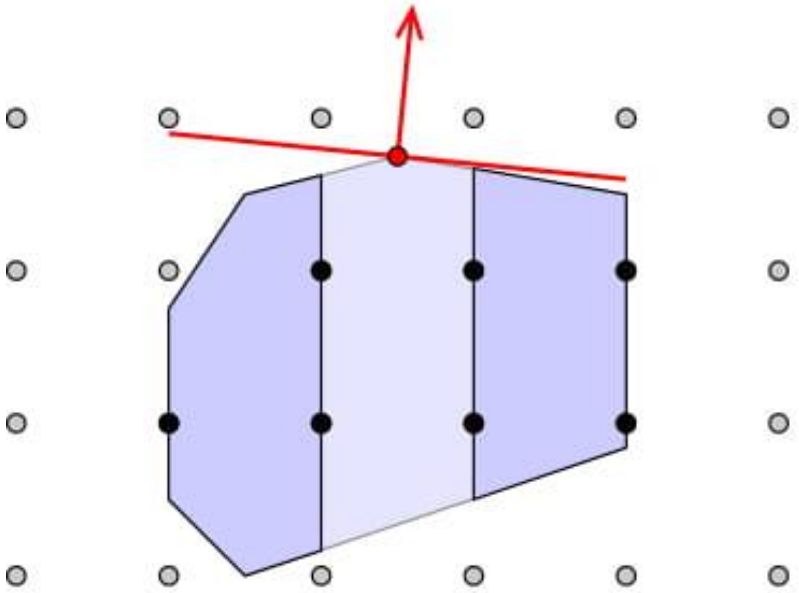
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching



LP-based Branch&Bound (colorful picture)

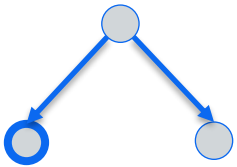
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

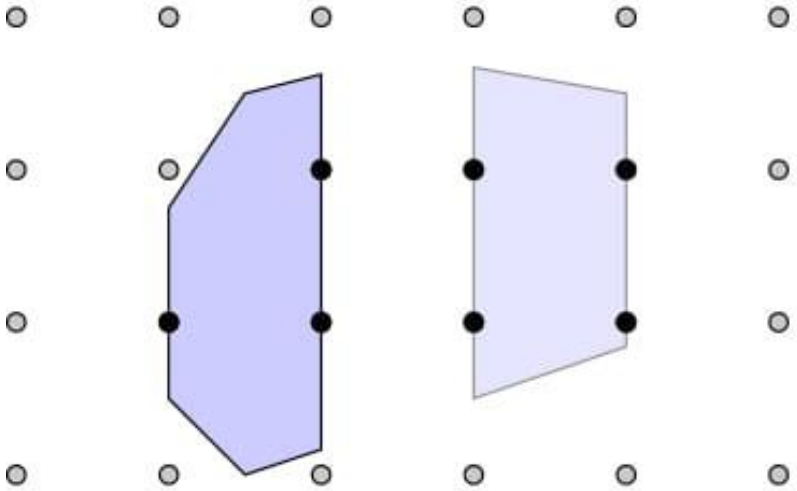




LP-based Branch&Bound (colorful picture)

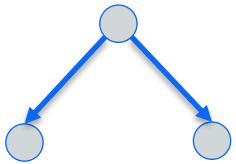
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

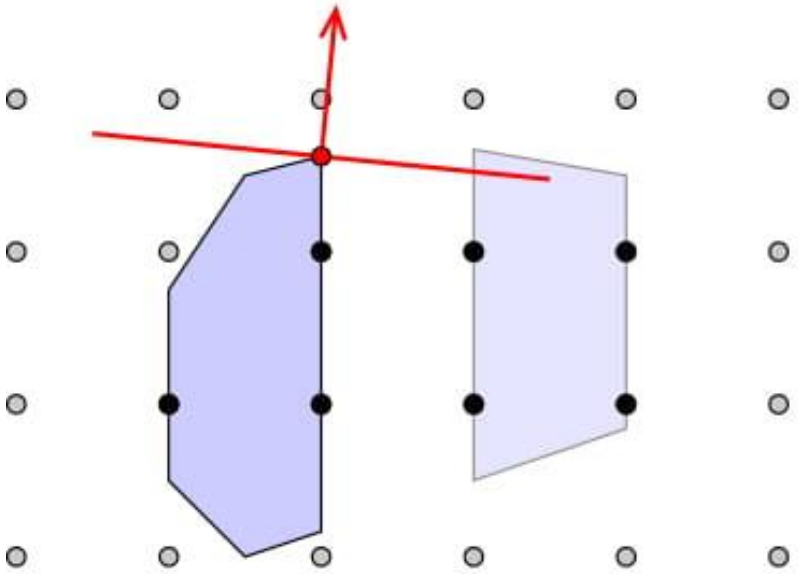




LP-based Branch&Bound (colorful picture)

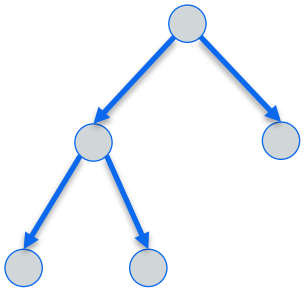
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

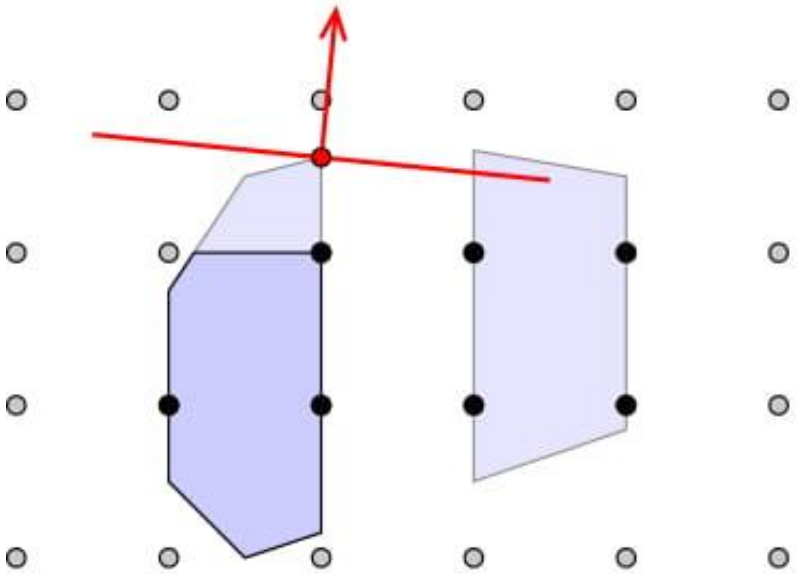




LP-based Branch&Bound (colorful picture)

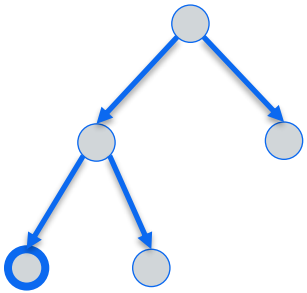
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

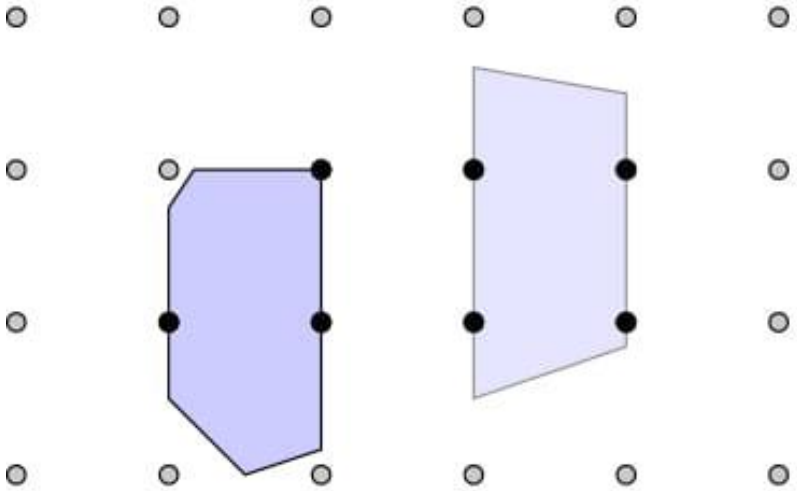




LP-based Branch&Bound (colorful picture)

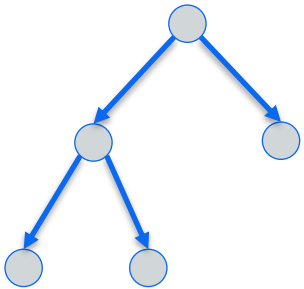
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

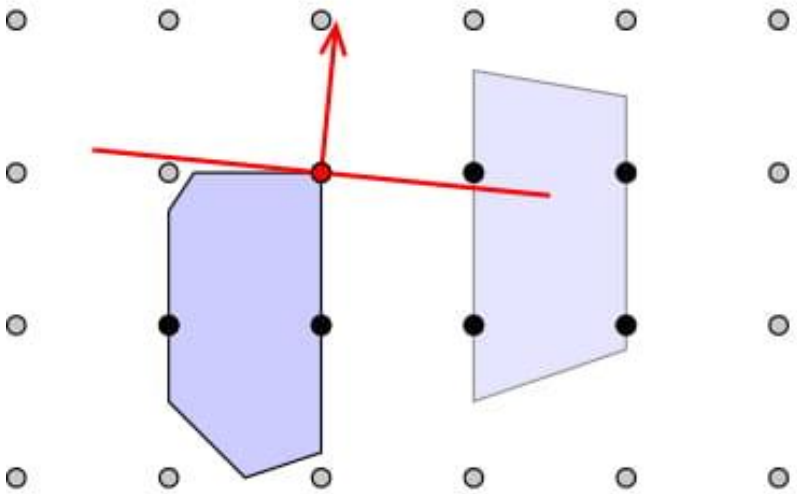




LP-based Branch&Bound (colorful picture)

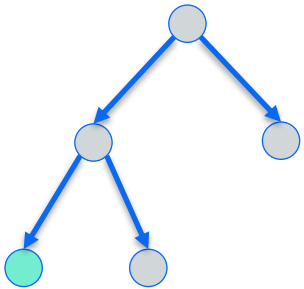
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

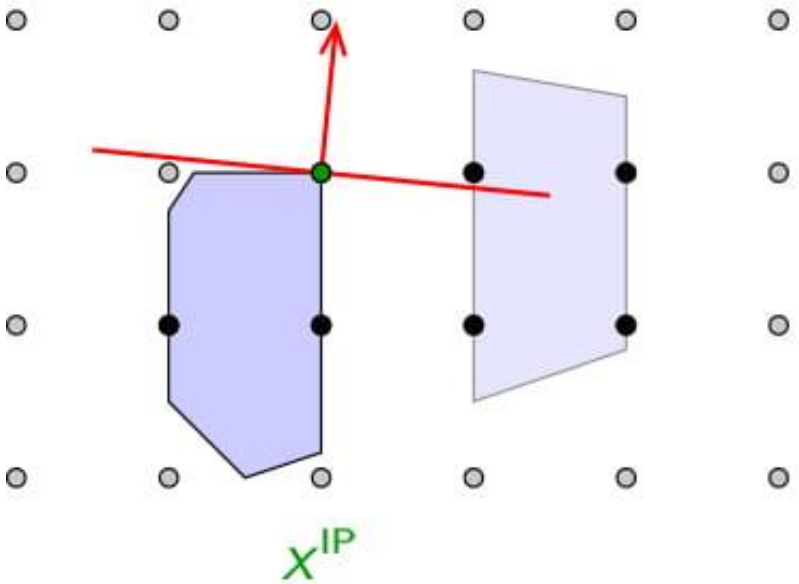




LP-based Branch&Bound (colorful picture)

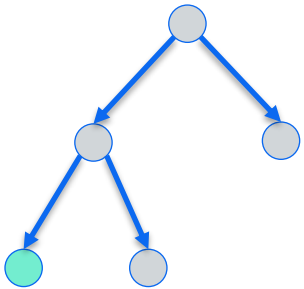
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

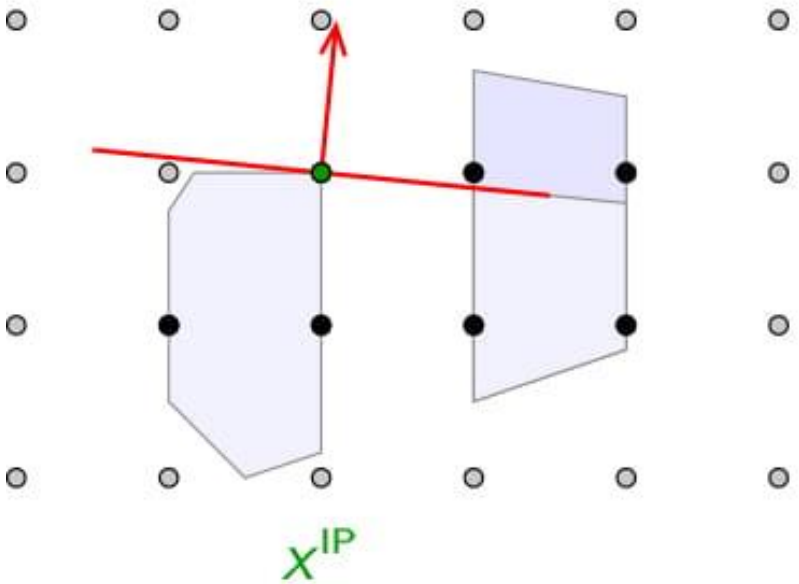




LP-based Branch&Bound (colorful picture)

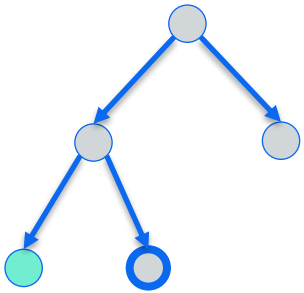
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

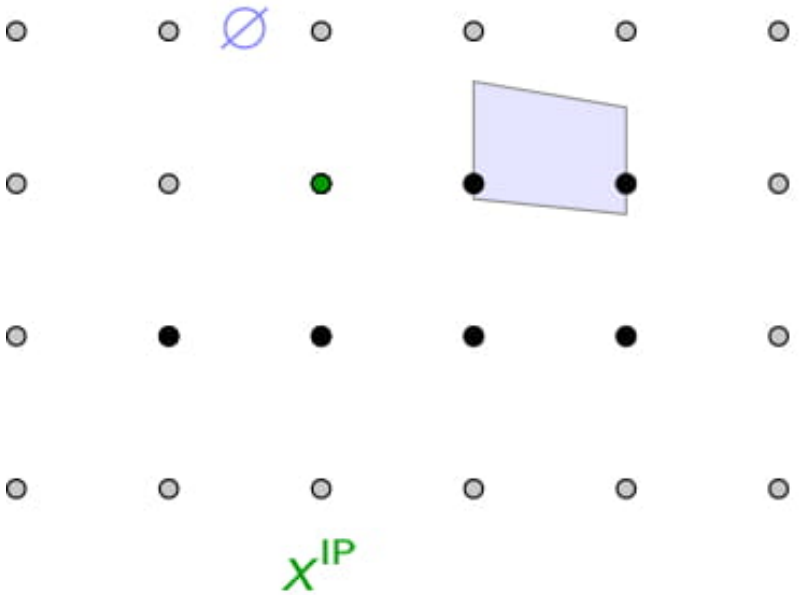




LP-based Branch&Bound (colorful picture)

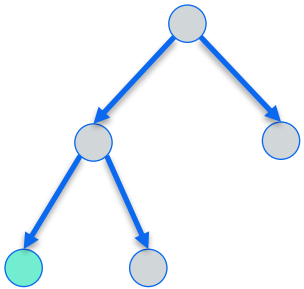
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

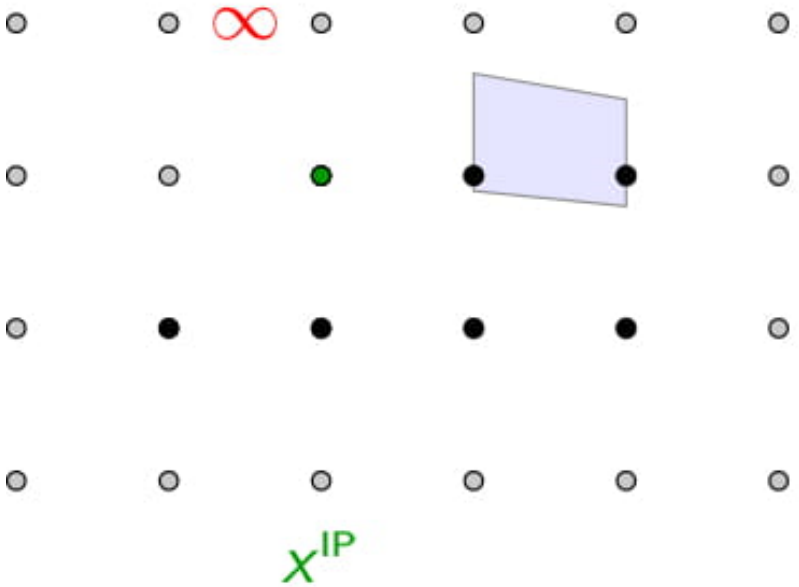




LP-based Branch&Bound (colorful picture)

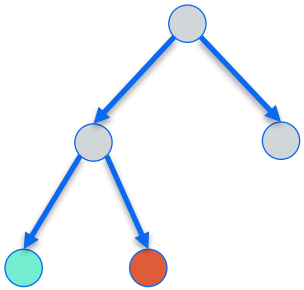
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

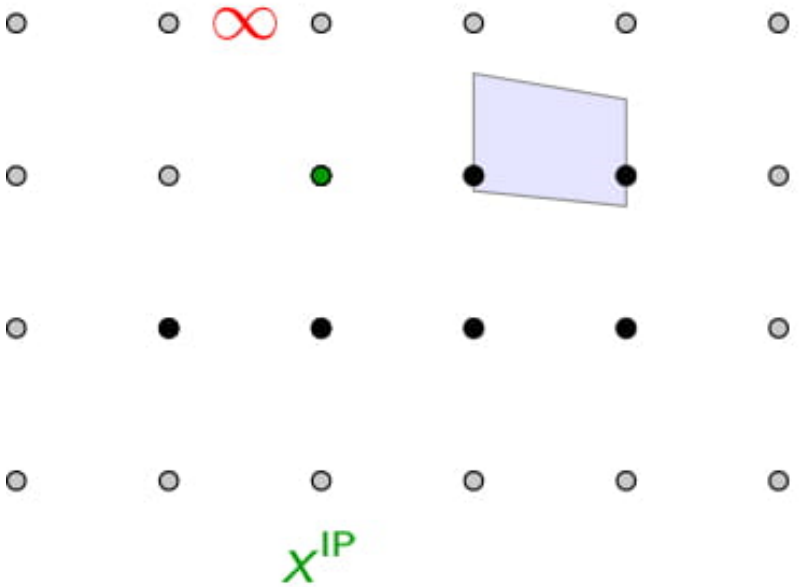




LP-based Branch&Bound (colorful picture)

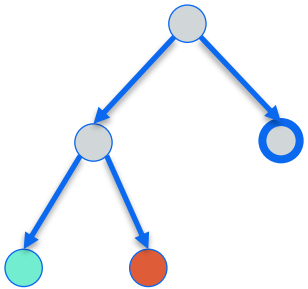
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

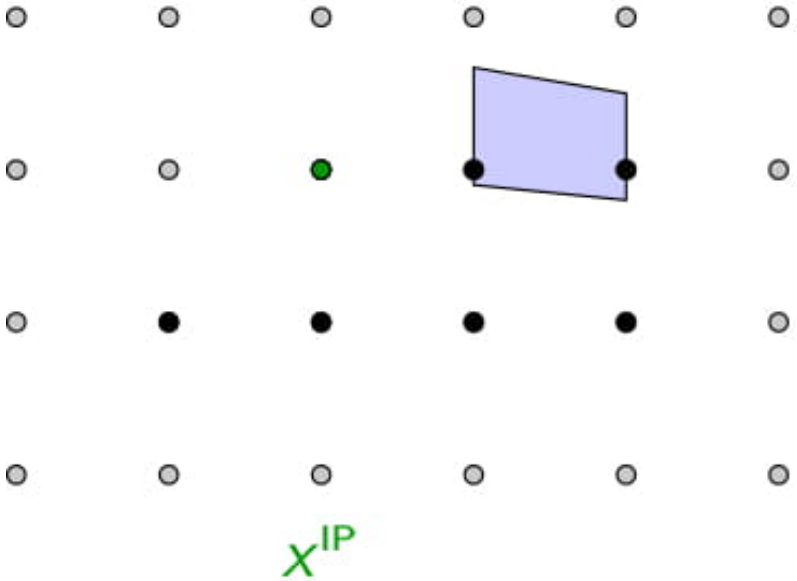




LP-based Branch&Bound (colorful picture)

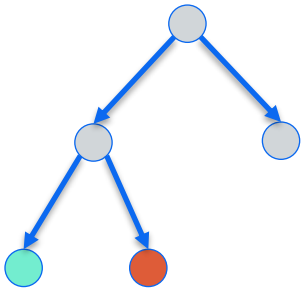
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

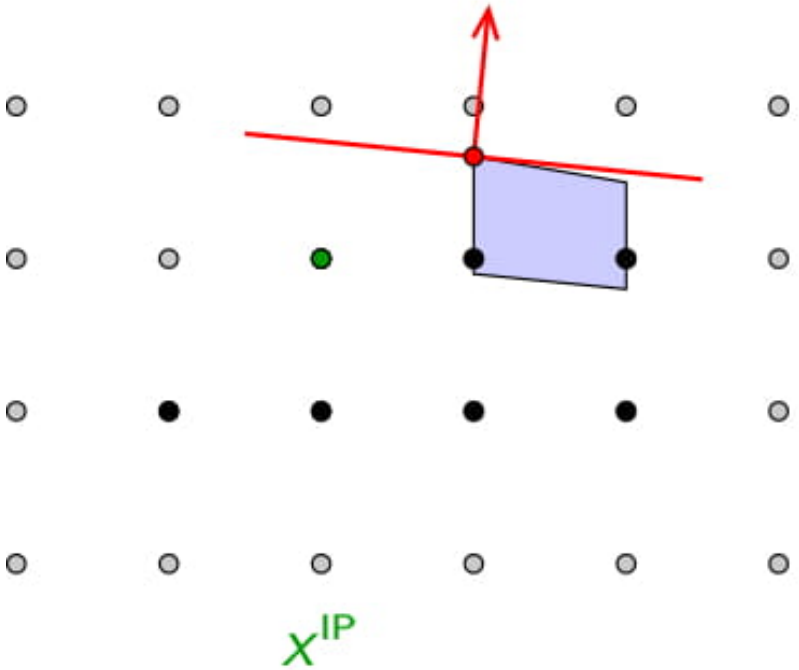




LP-based Branch&Bound (colorful picture)

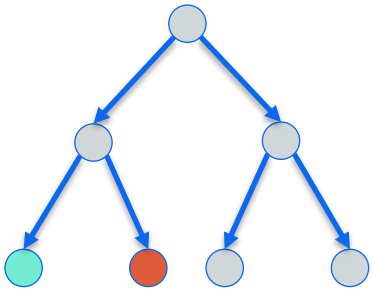
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

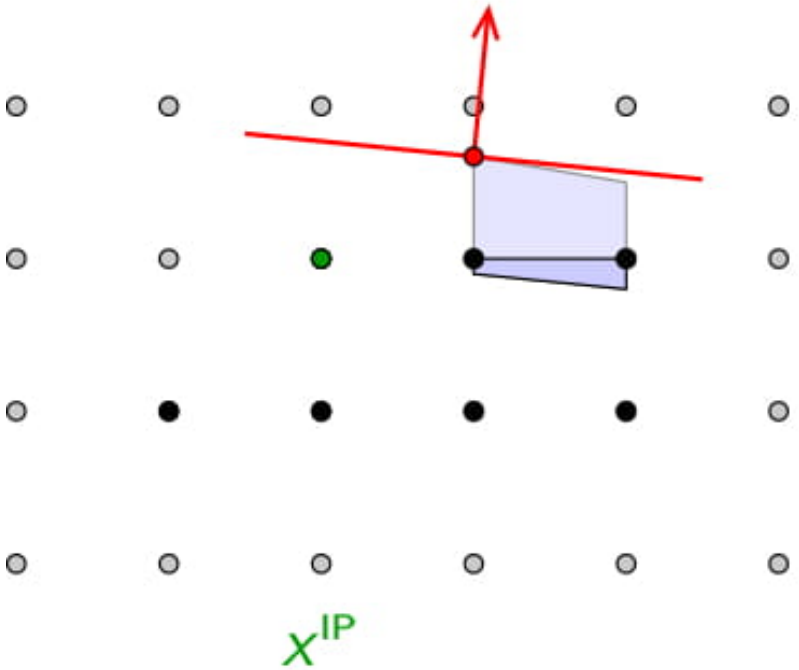




LP-based Branch&Bound (colorful picture)

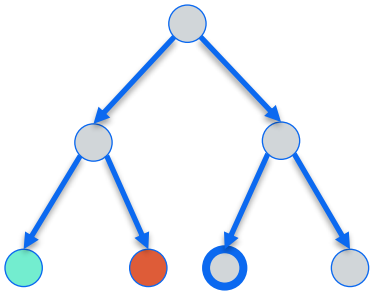
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

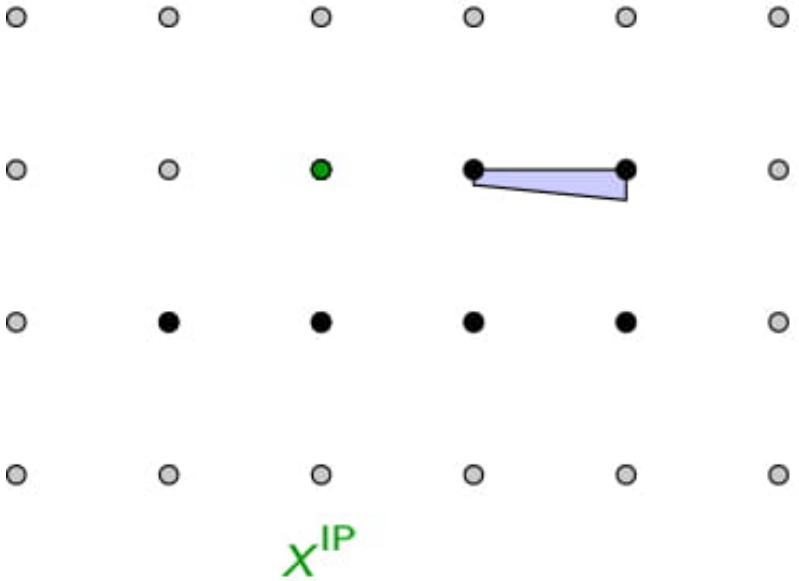




LP-based Branch&Bound (colorful picture)

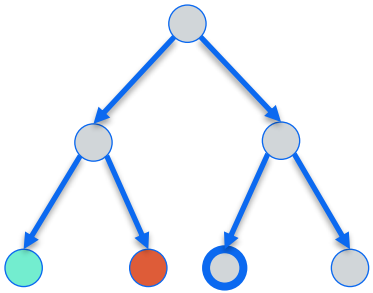
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

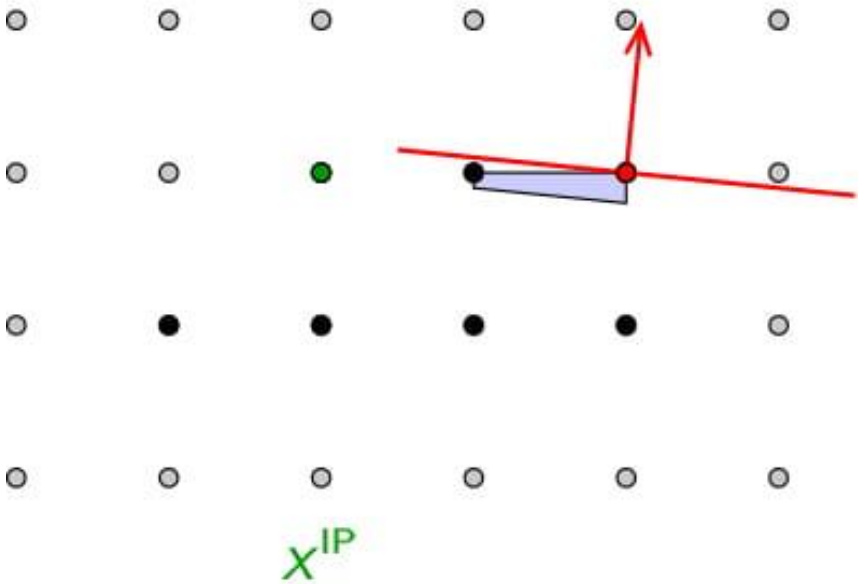




LP-based Branch&Bound (colorful picture)

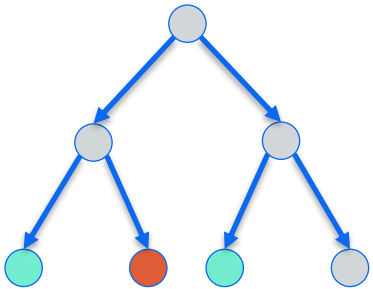
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

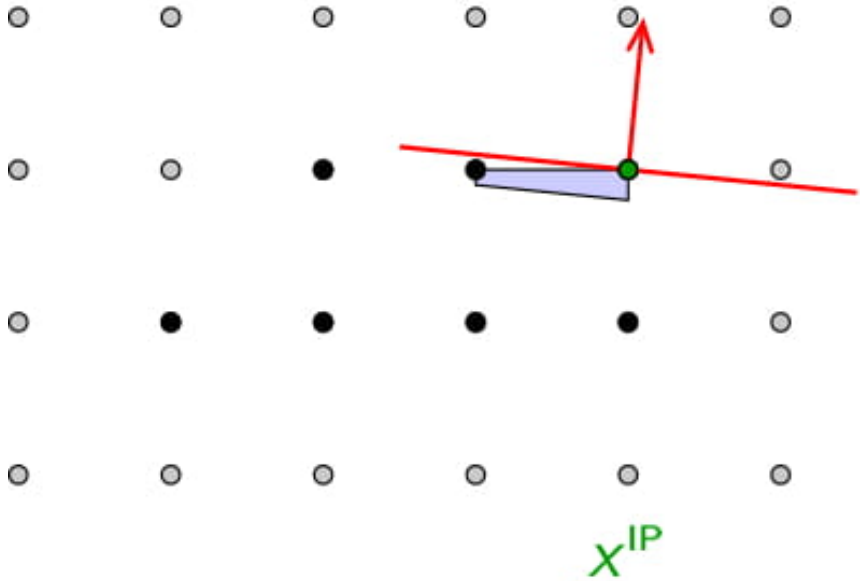




LP-based Branch&Bound (colorful picture)

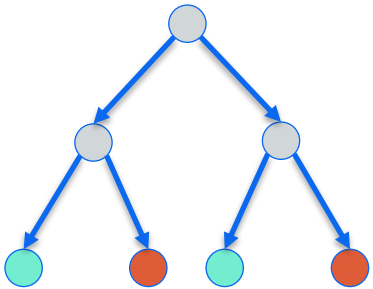
1. Abort Criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility Check
6. Branching

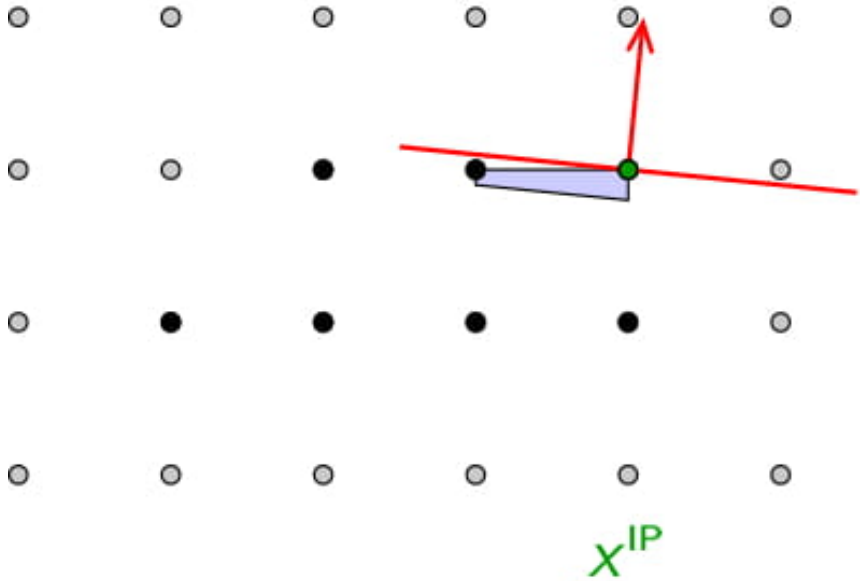




LP-based Branch&Bound (colorful picture)

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





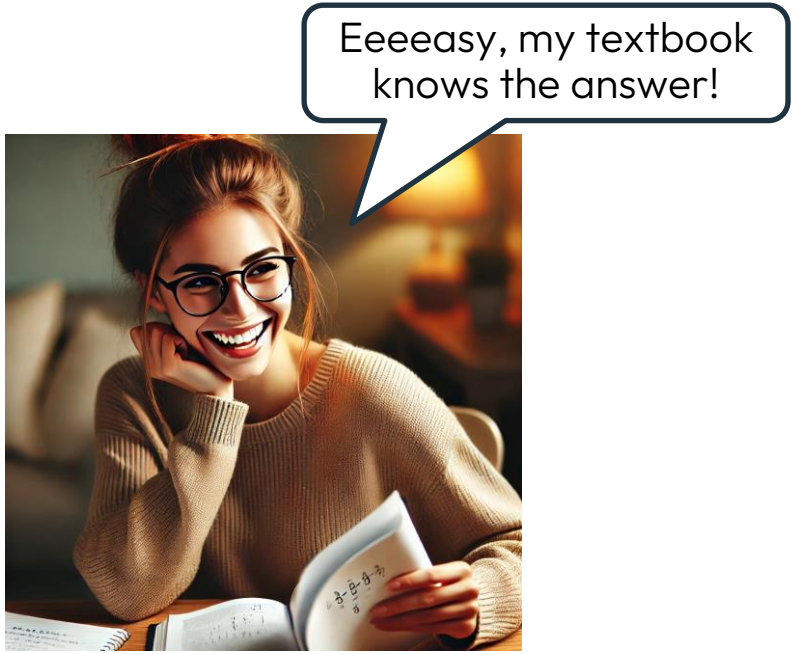
LP-based Branch&Bound (colorful picture)

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
  - Main goal: Improve dual bound
  - Bad selection might duplicate search effort
  - at every level….

So, what is a good branching rule?



- Branch on the most fractional variable
  - „most undecided“

- Most-fractional as bad as random branching
  - partially due to dual degeneracy
  - you can do orders of magnitude better

image source: DALL-E

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.

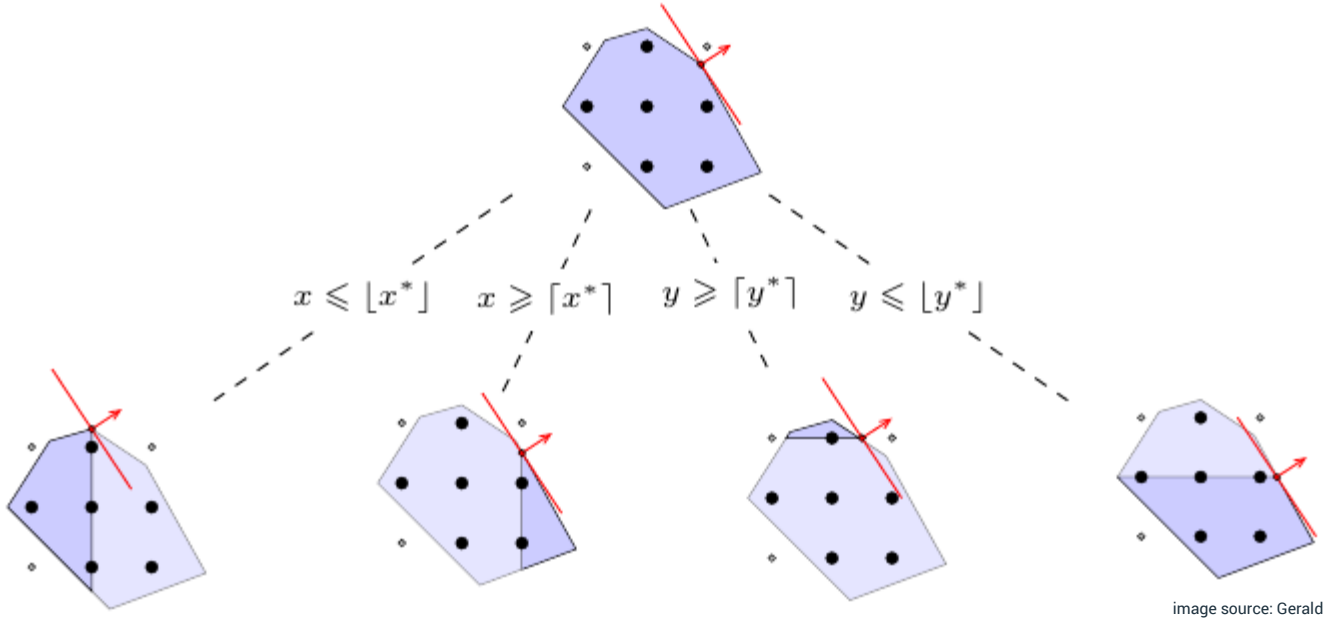


image source: Gerald Gamrath

Strong branching (Applegate et al 1995)

- Effective w.r.t. number of nodes, expensive w.r.t. time
- Strong branching might:
  - Change variable bound, when one side is infeasible
  - Detect local 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
- Important special case: probing
  - Only domain propagation, no LP

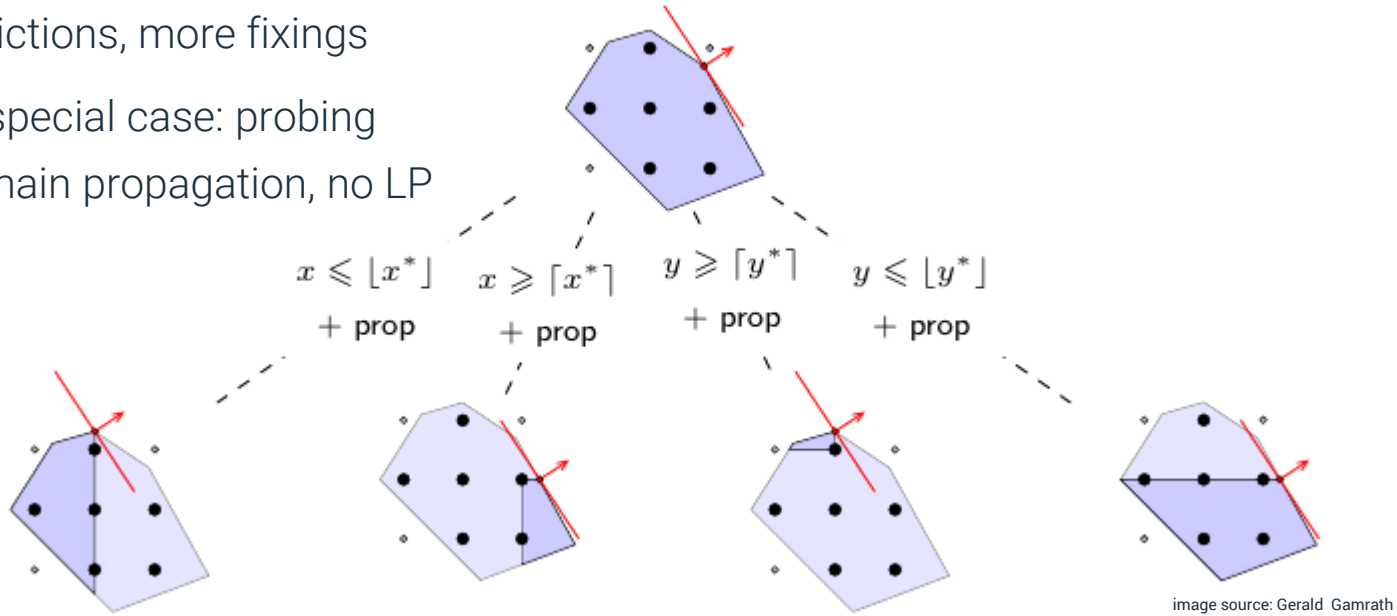
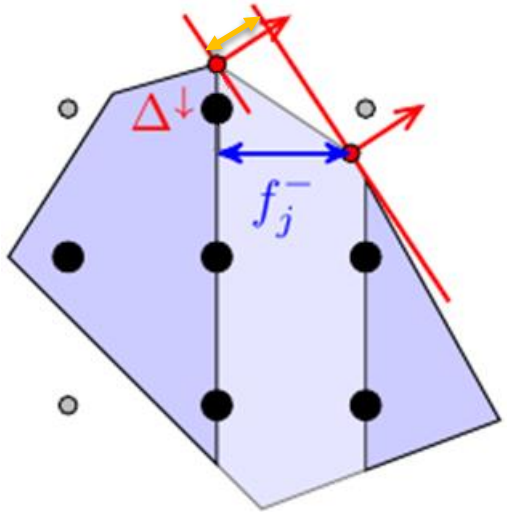
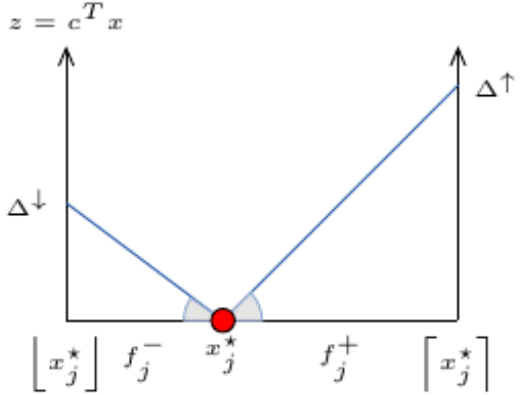


image source: Gerald Gamrath

Pseudo-costs (Bénichou 1971)

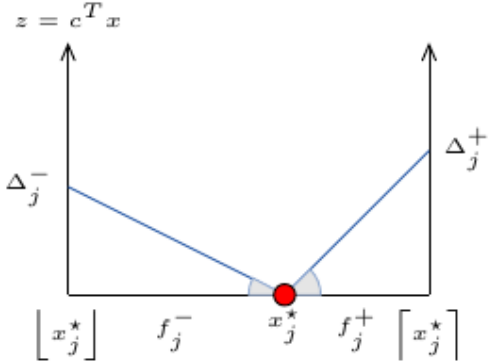
- Strong branching: A-priori observation, pseudo-costs: a-posteriori
- Estimate for objective gain based on past branching observations.
- Objective gain per unit fractionality: computed from fractionalities $f_j^-, f_j^+ := 1 - f_j^-$ and differences $\Delta^\downarrow, \Delta^\uparrow$ in LP values
  - downward obj. gain for var. j at current node: $\varphi_j^- = \frac{\Delta^\downarrow}{f_j^-}$
  - upward obj. gain for var. j at current node: $\varphi_j^+ = \frac{\Delta^\uparrow}{f_j^+}$
- Pseudo-costs Ψ_j^-, Ψ_j^+:average unit gain taken over all nodes that branched on same variable
  - downward pseudo-cost: $\Psi_j^- = \frac{\varphi_{j,1}^- + \cdots + \varphi_{j,k}^-}{k}$
  - upward pseudo-cost: $\Psi_j^+ = \frac{\varphi_{j,1}^+ + \cdots + \varphi_{j,k}^+}{k}$





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



Pro:

- Very cheap to compute

Cons:

- 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

Open questions:

- 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 of the objective change when branching
  b) Overestimator of the objective change when branching
  c) Heuristic 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
- A main drawback of pseudo-costs ist that
  a) They are uninitialized when the most important branchings are taken
  b) They are expensive to compute
  c) They become less reliable over time

Quiz time

- Pseudo-costs are an
  a) Underestimator of the objective change when branching
  b) Overestimator of the objective change when branching
  c) Heuristic 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
- A main drawback of pseudo-costs ist that
  a) They are uninitialized when the most important branchings are taken
  b) They are expensive to compute
  c) They become less reliable over time

Quiz time

- Pseudo-costs are an
  a) Underestimator of the objective change when branching
  b) Overestimator of the objective change when branching
  c) Heuristic 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
- A main drawback of pseudo-costs ist that
  a) They are uninitialized when the most important branchings are taken
  b) They are expensive to compute
  c) They become less reliable over time

Quiz time

- Pseudo-costs are an
  a) Underestimator of the objective change when branching
  b) Overestimator of the objective change when branching
  c) Heuristic 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
- A main drawback of pseudo-costs ist that
  a) They are uninitialized when the most important branchings are taken
  b) They are expensive to compute
  c) They become less reliable over time

Finally, all good now?

What if pseudo-costs don‘t work?

- Many situations where pseudocosts might not work well:
  - No objective → pseudo-costs all zero
  - Solution symmetries: Solution is “pushed around” → objective gains almost always zero
  - High dual degeneracy → objective gains often zero
  - Even if none of the above applies, we might need a tie-breaker!

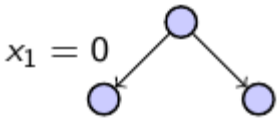
- Some (cheap) local rules might still work well: probing
- What other useful statistics can we collect about variables?

Hybrid branching and other history rules

Inference branching

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

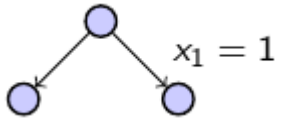
$$
\begin{aligned}
x_1 \quad\quad\quad + x_2 &= 1 \\
x_1 + x_3 + x_4 &\leq 1 \\
x_1 \quad\quad\quad + z &\geq 3 \\
z &\in \mathbb{Z}_+ \\
x_i &\in \{0,1\}
\end{aligned}
$$



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

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

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



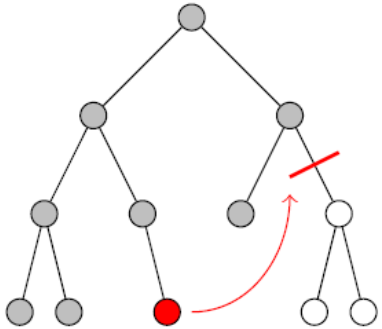
$$
\begin{aligned}
x_1 = 1 &\Rightarrow x_2 = 0 \\
&\Rightarrow x_3 = 0 \\
&\Rightarrow x_4 = 0
\end{aligned}
$$

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

VSIDS branching (Moskewicz et al 2001)

Conflict analysis:

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

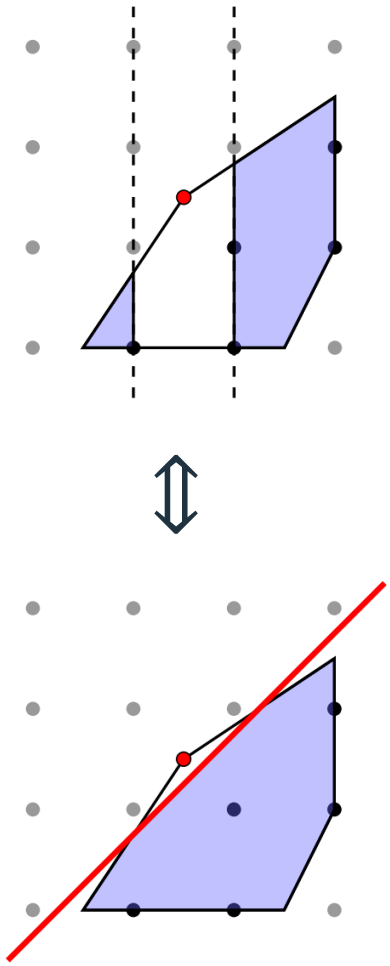


- VSIDS branching:
  - Variable which appears in highest number of (conflict) clauses
    - Branch towards infeasibility
  - Prefer “recent” conflicts: exponentially decreasing importance
  - Works particularly well for feasibility problems
  - State-of-the-art in SAT solving

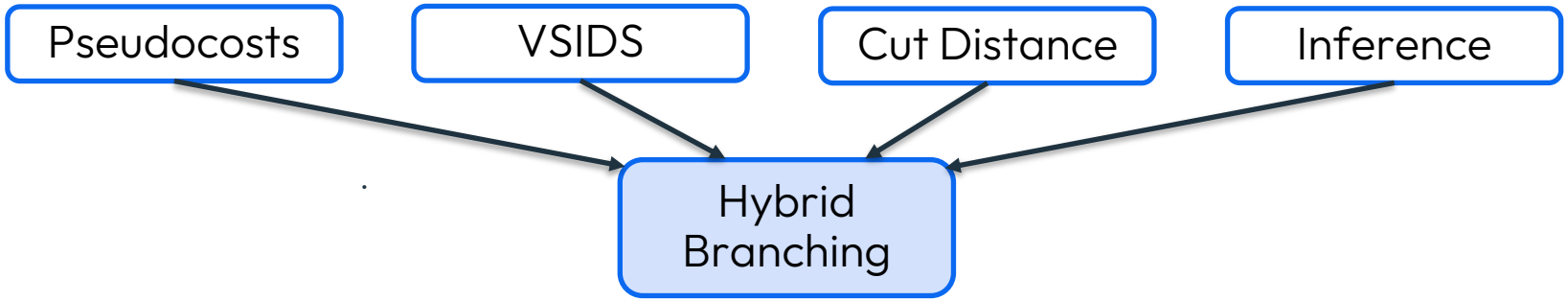
Branching scores based on intersection cuts (Turner et al. 2023)

- Observation:
  - Gomory Mixed Integer (GMI) cuts are split cuts equivalent to branching on a basic variable
- Idea:
  - Deep cuts correspond to important branchings
  - Favor branching variables associated with “strong” GMI cuts
  - Cut Distance score:

$CD(x_j) :=$ Depth of the intersection cut from the tableau row of x_j



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 50% candidates, second filter 25%,...

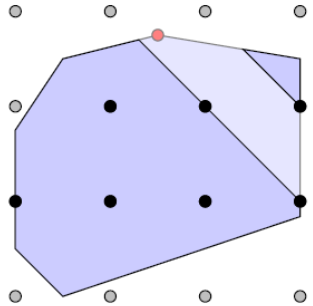
Branching score

- Most branching rules yield two values: One for down-, one for up-branch
- Need to combine them to a single value
- First idea: average
- Slightly more involve: convex sum
  - $\text{score}(x_j\ ) = \lambda \max\{s_j^-\ , s_j^+\} + (1-\lambda)\ \min\{s_j^-\ , s_j^+\}$
  - 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

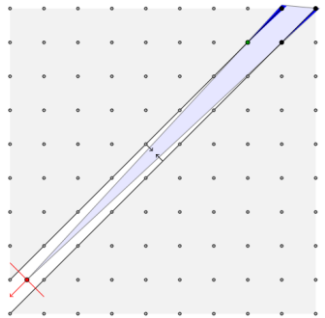
Branching on general disjunctions

$$\pi^T x \leq \pi_0 \quad \bigvee \quad \pi^T x \geq \pi_0 + 1$$

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

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

- represents decisions in original problem
- moderately enlarged candidate set

Quiz time

- Strong Branching + Pseudocost Branching =
  a) Hybrid Branching
  b) Reliability Branching
  c) Inference Branching
- Which of the following are history-based branching scores?
  a) VSIDS
  b) Inference Scores
  c) Probing

Quiz time

- Strong Branching + Pseudocost Branching =
  a) Hybrid Branching
  b) Reliability Branching
  c) Inference Branching
- Which of the following are history-based branching scores?
  a) VSIDS
  b) Inference Scores
  c) Probing

Quiz time

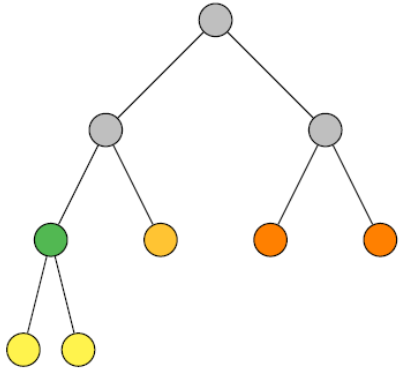
- Strong Branching + Pseudocost Branching =
  a) Hybrid Branching
  b) Reliability Branching
  c) Inference Branching
- Which of the following are history-based branching scores?
  a) VSIDS
  b) Inference Scores
  c) Probing

Node selection

Considerations

Goals:

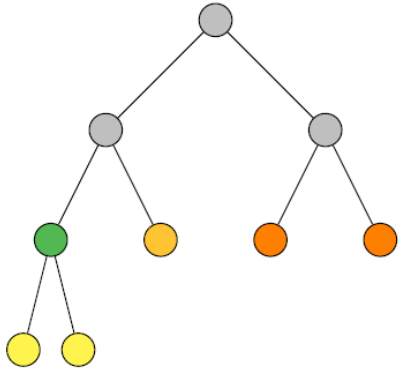
- Improve primal bound to enable pruning
- Improve global dual bound
- Keep computational effort small
  - Prefer children over siblings over others
- Ramp-up:
  - Diversification
  - For parallelization



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

What do typical branching trees look like?

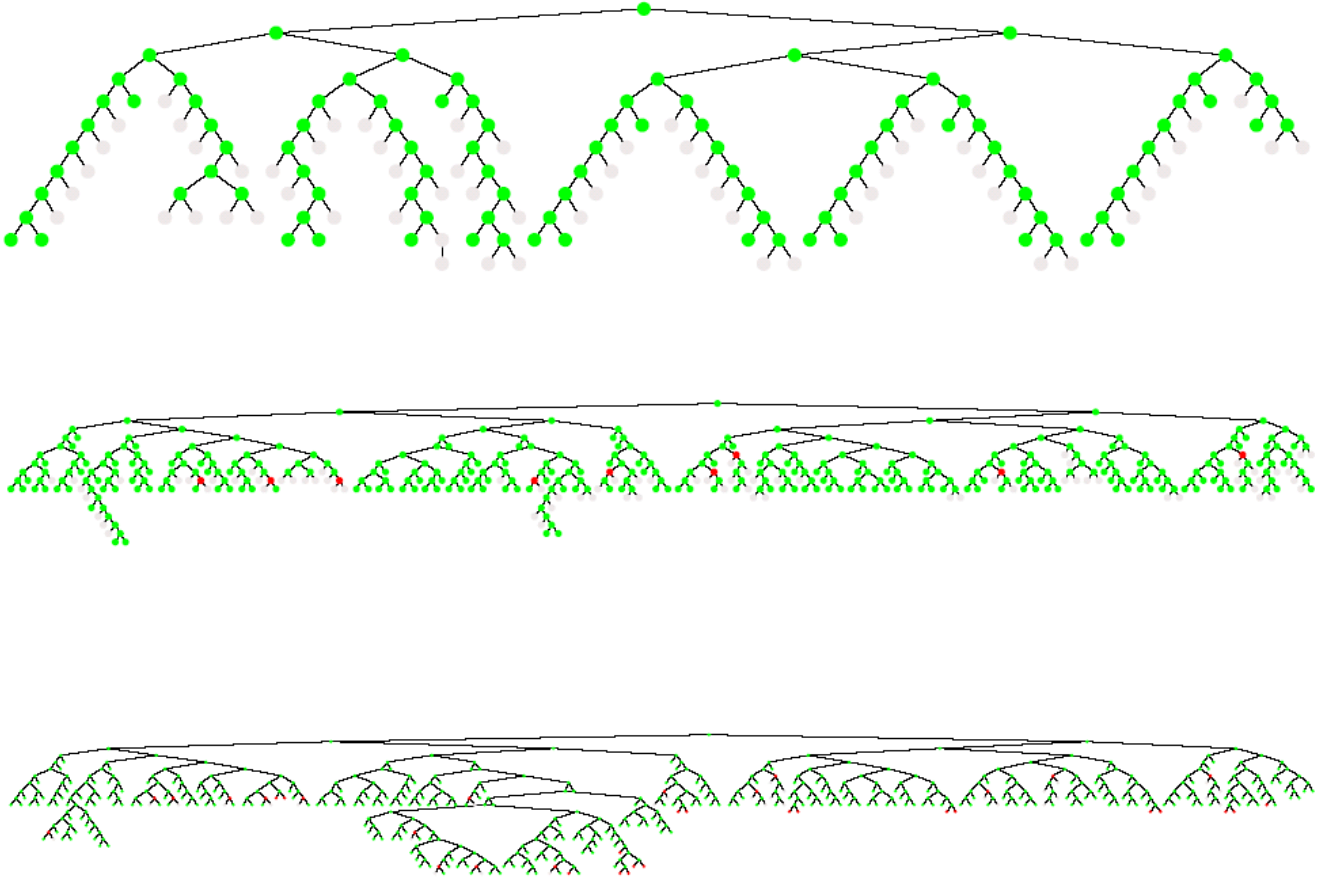


image source: Thorsten Koch

What do typical branching trees look like?

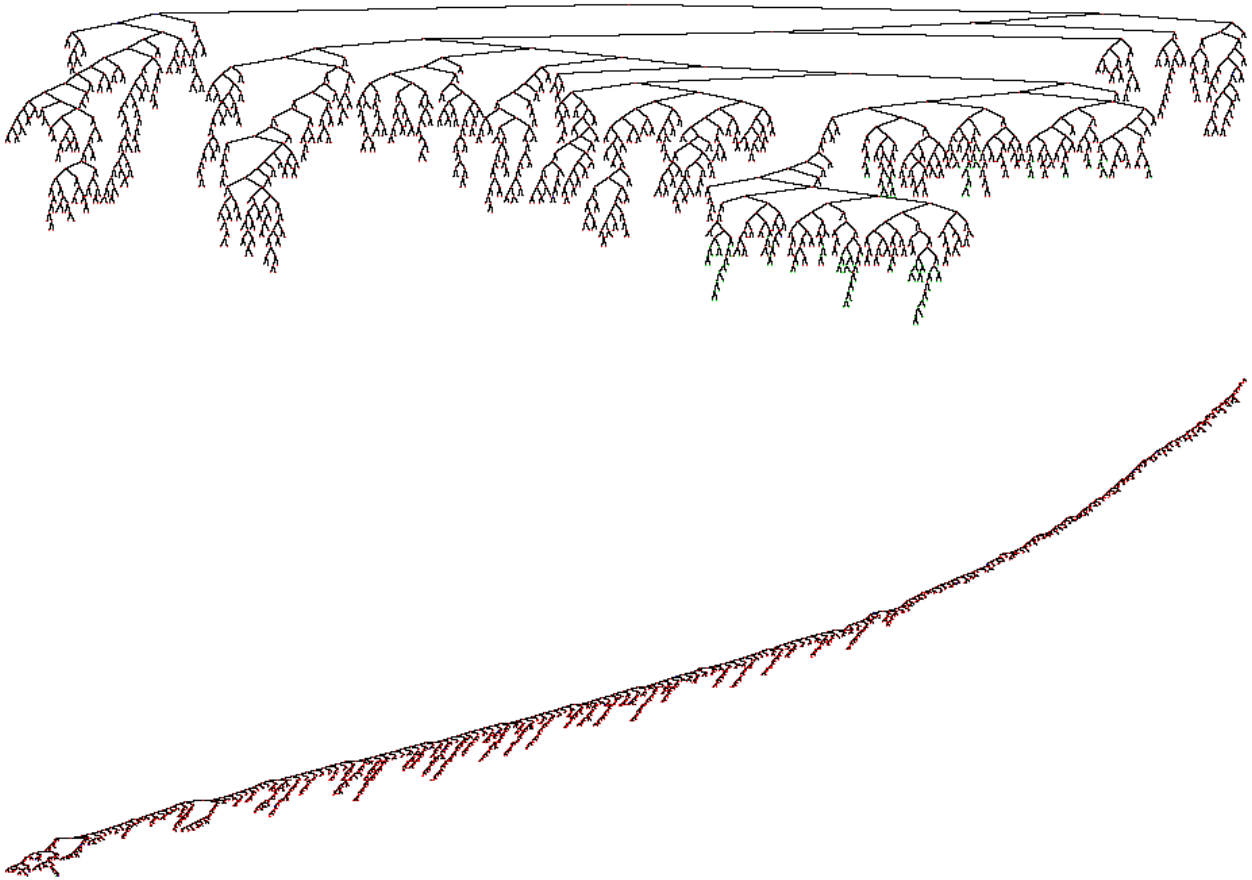


image source: Thorsten Koch

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

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

Thank You!

Timo Berthold
TU Berlin, FICO, MODAL