When MIP solvers roll dice

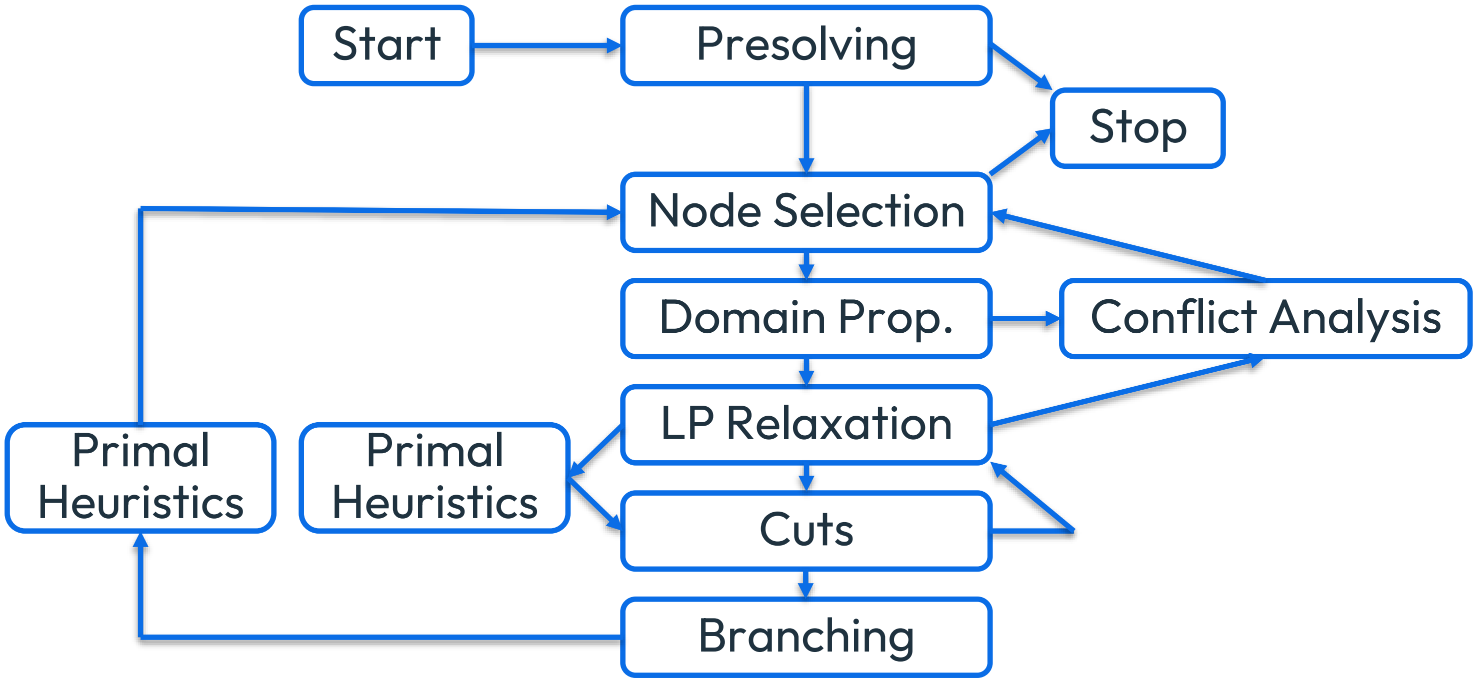
Primal Heuristics

PD Dr. Timo Berthold
Private Lecturer @ TU Berlin
Director, Mixed-Integer Optimization @ FICO

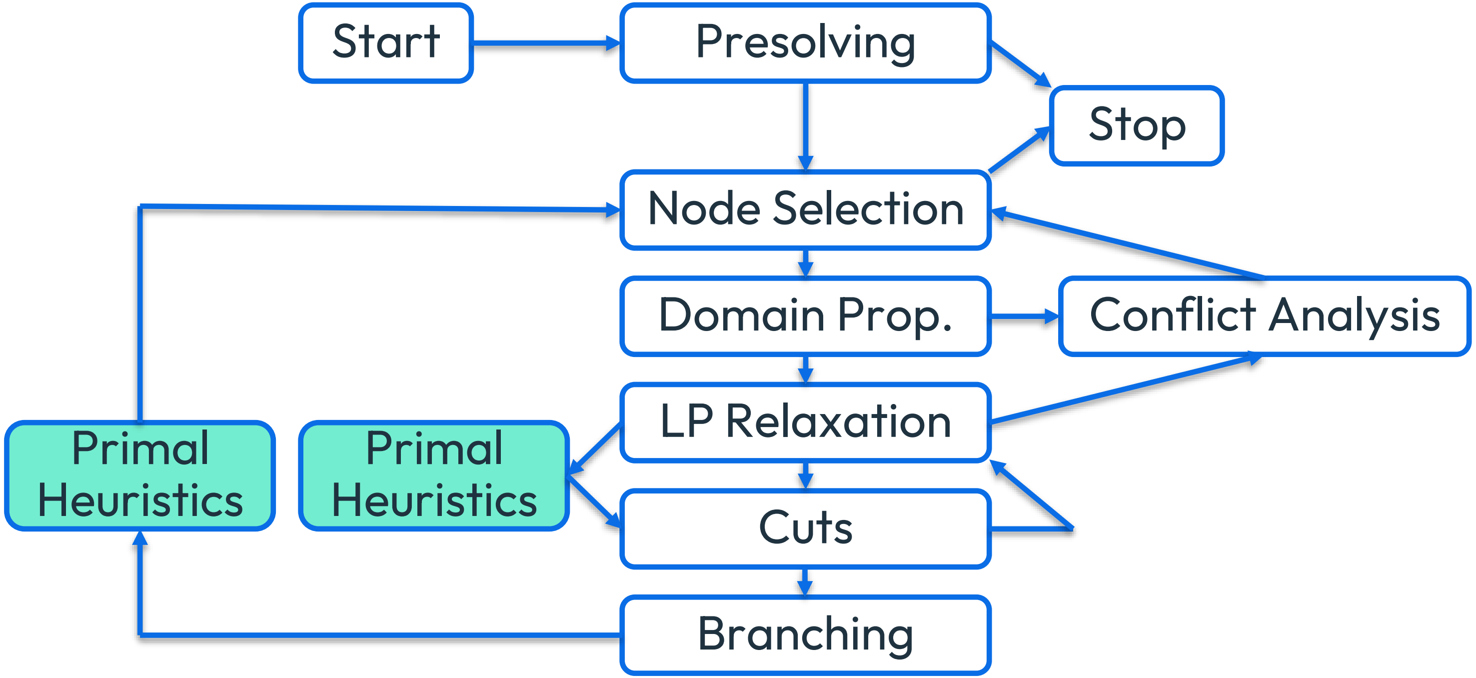
Agenda

1. Heuristics: Idea, Information
2. Rounding, Diving
3. Feasibility Pump
4. LNS
5. Primal-Dual Integral

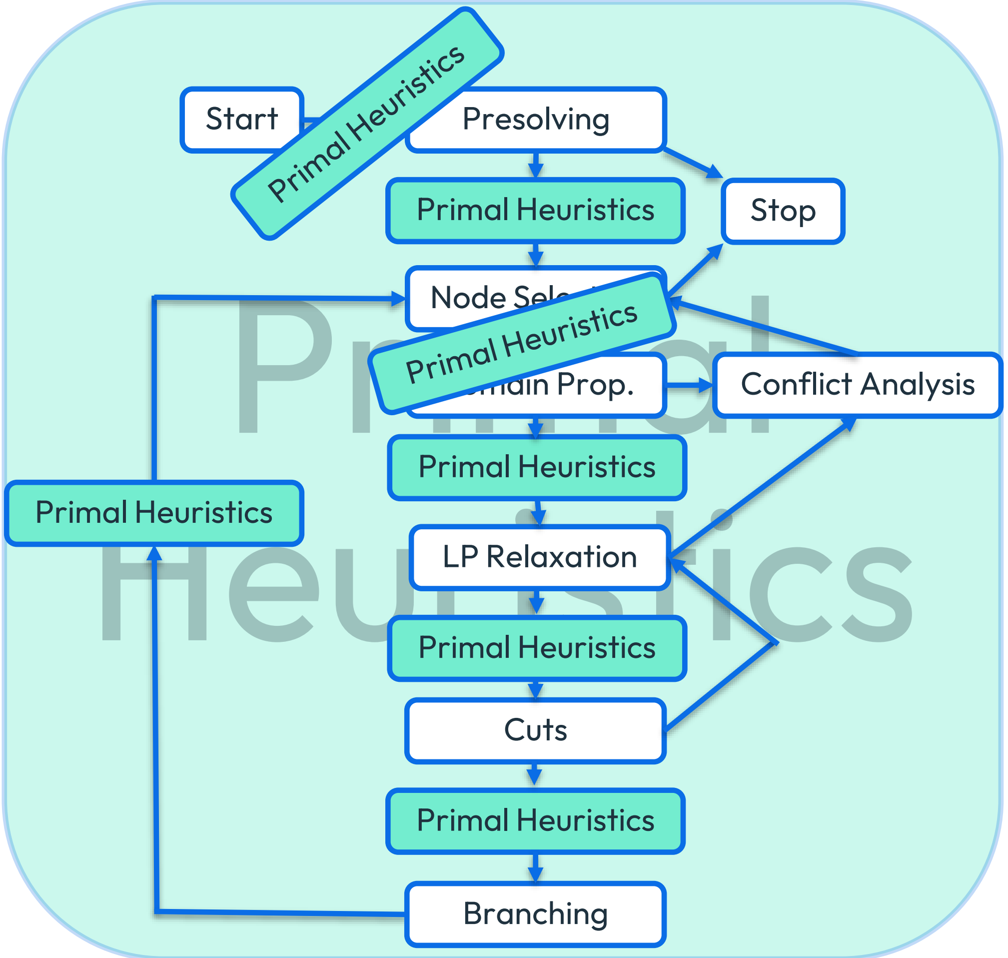
MIP Solver Flowchart



MIP Solver Flowchart



Heuristics Everywhere…

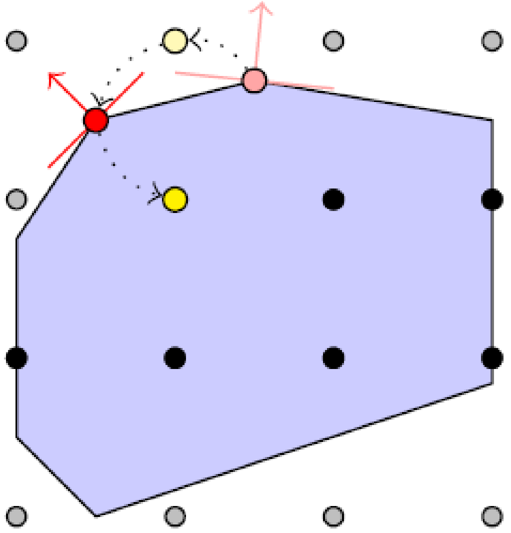


- Heuristics may typically be applied:
  - Before presolving
  - After presolving, before LP
  - After LP, before cut loop
  - During cut loop
  - After node
  - When backtracking
  - **In the background!**
- Heuristics may trigger other heuristics

What Are Primal Heuristics?

- Primal heuristics . . .
  - are incomplete methods which
  - often find good solutions
  - within a reasonable time
  - without any warranty!

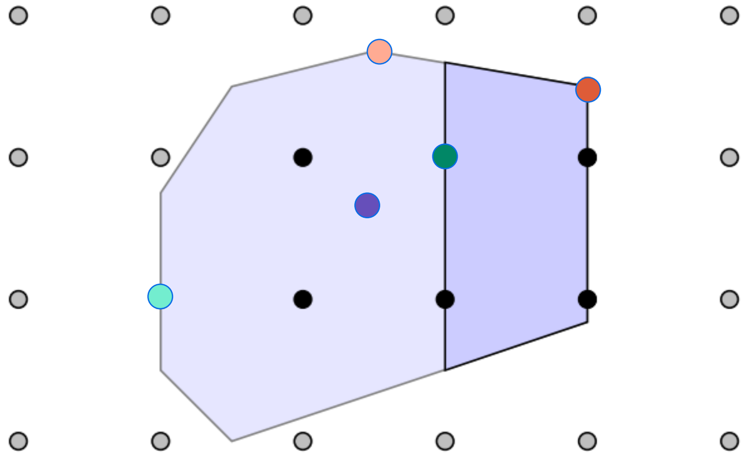
- Why use primal heuristics inside a (complete) MIP solver?
  - to prove the feasibility of the model
  - often nearly optimal solutions suffice in practice
  - feasible solutions guide the remaining search process



Heuristics: General Information

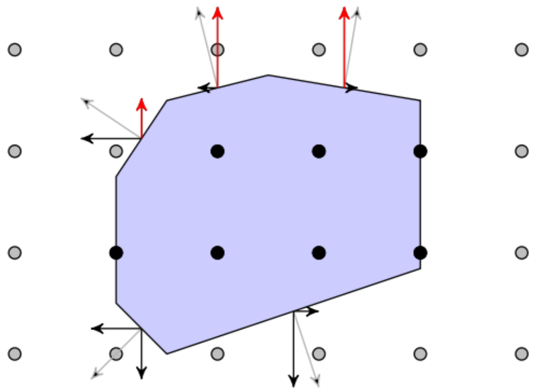
Reference Points

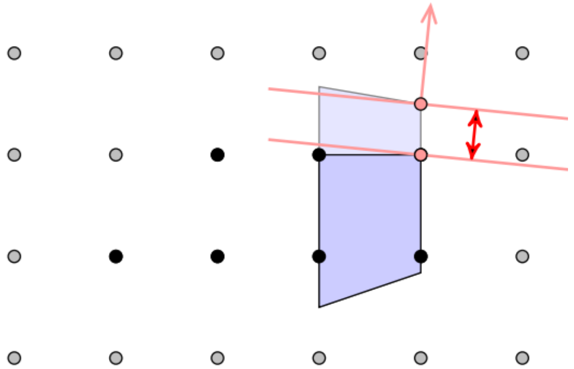
- Current LP optimum
- Current incumbent
- Other feasible solutions
- LP optimum at root node
- Analytic center



Variable Statistics

- Locking numbers
  - Number of potentially violated rows
- Pseudo-costs
  - Average objective change
- Conflict Statistics
  - How often has a variable been involved in proving local infeasibility?



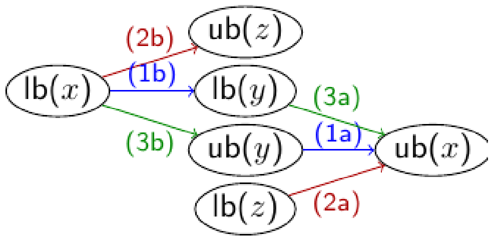


Global Structures

- General structures that are automatically detected during presolving
  - Clique table
  - Flow structures
  - Implication graph
  - Variable bound graph
  - Symmetry information

$$
\begin{aligned}
x - 2y &\leq 3 \quad (1)\\
x + 2z &\leq 2 \quad (2)\\
x + 3y &\leq 6 \quad (3)
\end{aligned}
$$

$$
\begin{array}{lcll}
x & \leq & 2y + 3 & (1a)\\
y & \geq & \frac{1}{2}x - \frac{3}{2} & (1b)\\
x & \leq & 2 - 2z & (2a)\\
z & \leq & 1 - 0.5x & (2b)\\
x & \leq & 6 - 3y & (3a)\\
y & \leq & 2 - \frac{1}{3}x & (3b)
\end{array}
$$



Different Types of Heuristics

- Rounding: Take a (fractional) LP solution, change fractional to integral values without reoptimization
- Diving: simulate a depth-first search (DFS) in the branch-and-bound tree using some special branching rule (i.e. fix variables and reoptimize)
- FP-type: manipulate objective function in order to reduce fractionality, reoptimize
  - FP=Feasibility Pump, sometimes referred to as objective diving
- Local Search: Step-by-step updates of an integer reference vector
- Large Neighborhood Search: fix variables, add constraints, solve resulting subproblem
- Pivoting: manipulate simplex algorithm
- ...

Start Vs. Improvement Heuristics

- Start heuristics
  - Applied early in the search process
  - Often at root node
  - Typically start from LP optimum
  - Ignore incumbent (if one exists)
- Improvement heuristics
  - Require feasible solution
  - Normally at most once for each incumbent
  - Quick improvement directly after incumbent
  - Heavy improvement only after long time without new incumbent

Rounding and Diving

Rounding Heuristics

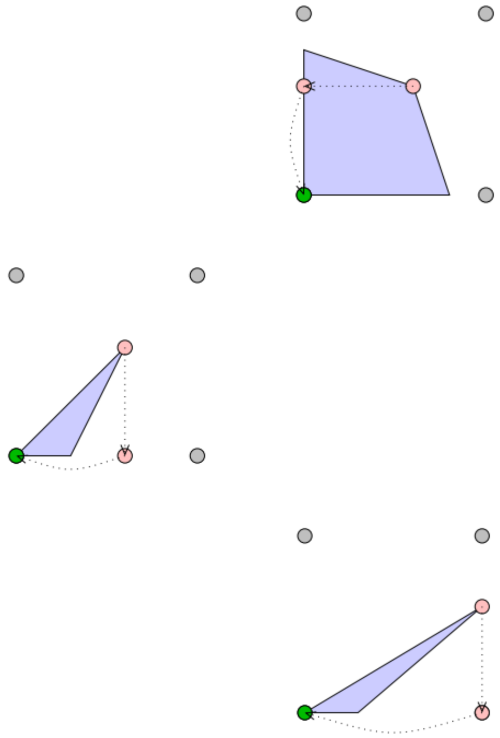
Variable locks as main guide, fast fail strategy

- Simple Rounding: always stays feasible
  - Only applicable when zero locks on all fractionals
- Rounding may violate constraints
  - Round in direction of fewer locks
- Shifting: may unfix integers
  - Again, depending on locks

Other approaches:

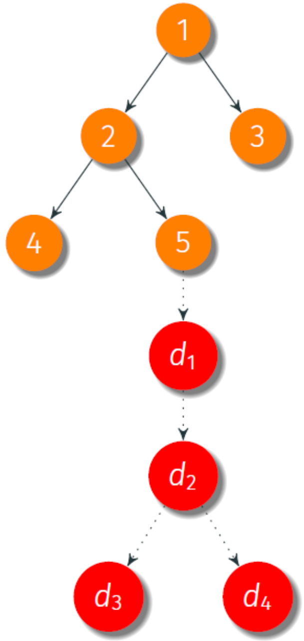
- Random rounding
- Analytic center rounding

For continuous variables: solve final LP



Diving Heuristics

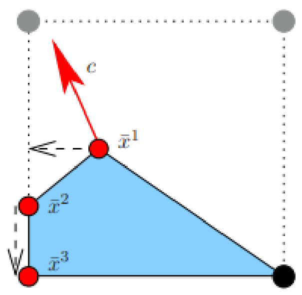
- Simulated tree search
- Pure DFS
  - At most 1-level backtracking
- „One-sided“ Branching rule
- Might fix several variables per branch
- Might skip LP and only propagate at some nodes
- Often applied before MIP solver would backjump in main tree



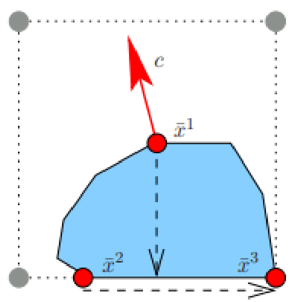
Rule of thumb: Good branching strategies are bad diving strategies

Main Difference: Variable Fixing Strategy

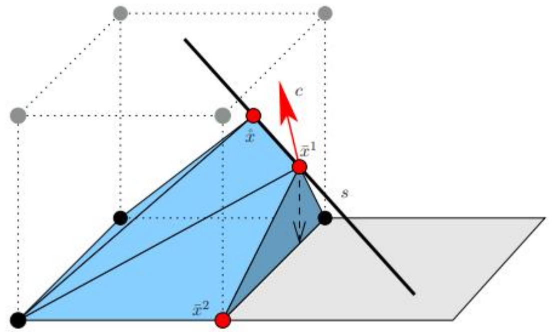
- Fractional diving: Round least fractional
- Guided diving: Round towards reference solution
- Coefficient diving: Round in direction of fewer locks
- Line search diving: Observe development since root LP
- Vector length diving: Fix variables in long constraints



Fractional Diving



Coefficient Diving



Linesearch Diving

Local Search

Local Search

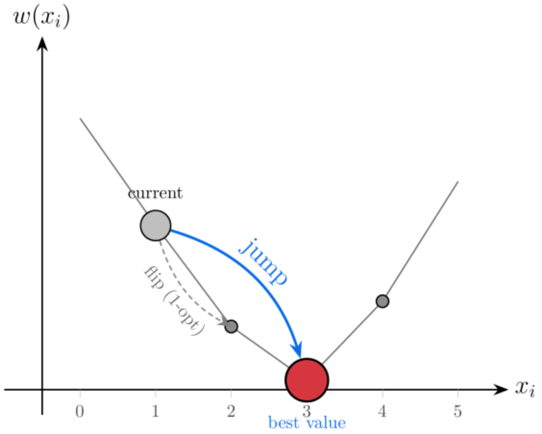
1. Consider reference solution
2. Unfix small number of variables and explicitly try alternative values
   - Efficient for 1-neighborhood
   - Runtime (naively): $O(n^k)$ for k-neighborhood
     - Already for 2-neighborhood, only a subset of candidates can be considered
     - Exploit structures to reduce runtime
       - Example: Lin Kernighan Heuristic
       - Check https://stemlounge.com/animated-algorithms-for-the-traveling-salesman-problem/
3. Accept new solution if it improves incumbent
4. Stop if locally optimal
   - Might allow some suboptimal moves to break out

Feasibility Jump

- Luteberget & Sartor, Mathematical Programming Computation 15, 2023; winner of the MIP 2022 Computational Competition
- Randomized local search:
  - Minimizes total weighted constraint violation
    - Interpreted as a Lagrangian relaxation
  - Jump: sets variable directly to its best value, not just a ±1 flip
  - Only applied to all-integer problems
  - Only used to find a first feasible solution

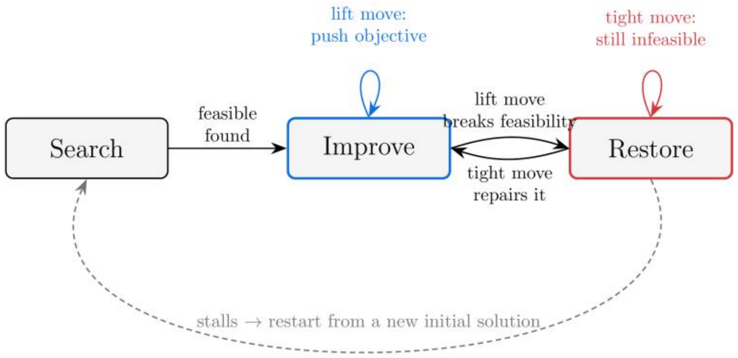
```
initialize all row weights to 1
initialize current solution to 0

while there are violated constraints
   if there are improving moves (flips)
      randomly sample N improving moves
      apply best move in sample
   else
      increase weight for currently violated rows
      perform a random (non-improving) move
```



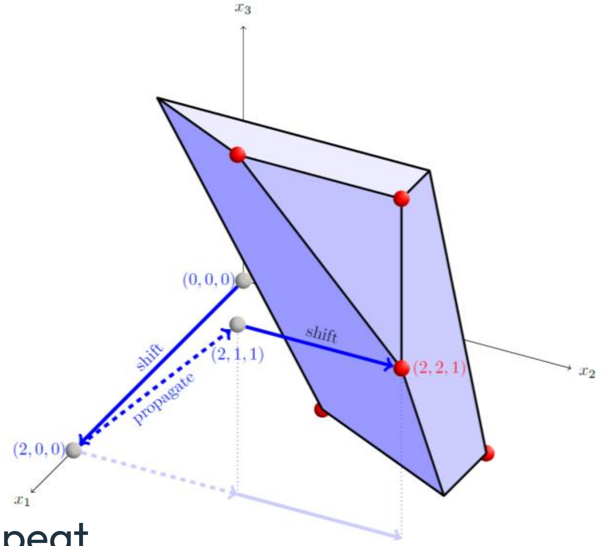
Local-ILP & Local-MIP

- Lin, Cai, Zou & Lin: efficient local search for MIP, Artificial Intelligence 348, 2025
- Three-mode cycle:
  - Search: find a first feasible assignment
  - Improve: lift move pushes the objective
  - Restore: tight move repairs feasibility after a lift move
  - Stalls → restart from a new initial solution
- Same building blocks as Feasibility Jump:
  - Complete assignment, single-variable moves, no LP relaxation, weighted score
  - Local-MIP adds a breakthrough move for native continuous variables



Fix-And-Propagate Heuristics

- Work a partial integer reference point, possibly infeasible
- Applies several rounds of greedy fixings and bound propagations to improve the solution
- Columns are typically scored using some combination of objective value and row violations
- In one round:
  - Select an unfixed column
  - Fix the column to the value that minimizes the score
  - Propagate bound implications from the fixing
    - Reverse fixing if infeasible
  - Continue until all integer variables are fixed
- Update scoring weights, randomize parts of the solution and repeat
- Repair moves to break out of infeasibility



Quiz Time

- Diving heuristics
  a) Simulate a depth-first search
  b) Manipulate the simplex algorithm
  c) Change the objective function
- Fix-and-propagate heuristics typically
  a) Solve a sub-MIP
  b) Solve a series of LPs
  c) Do not solve LPs or auxiliary MIPs
- Primal heuristics
  a) Typically have an approximation ratio
  b) Always improve the primal bound
  c) Might terminate unsuccessfully

Quiz Time

- Diving heuristics
    - **a) Simulate a depth-first search**
    - b) Manipulate the simplex algorithm
    - c) Change the objective function
- Fix-and-propagate heuristics typically
    - a) Solve a sub-MIP
    - b) Solve a series of LPs
    - c) Do not solve LPs or auxiliary MIPs
- Primal heuristics
    - a) Typically have an approximation ratio
    - b) Always improve the primal bound
    - c) Might terminate unsuccessfully

Quiz Time

- Diving heuristics
  a) Simulate a depth-first search
  b) Manipulate the simplex algorithm
  c) Change the objective function
- Fix-and-propagate heuristics typically
  a) Solve a sub-MIP
  b) Solve a series of LPs
  c) Do not solve LPs or auxiliary MIPs
- Primal heuristics
  a) Typically have an approximation ratio
  b) Always improve the primal bound
  c) Might terminate unsuccessfully

Quiz Time

- Diving heuristics
  a) **Simulate a depth-first search**
  b) Manipulate the simplex algorithm
  c) Change the objective function
- Fix-and-propagate heuristics typically
  a) Solve a sub-MIP
  b) Solve a series of LPs
  c) **Do not solve LPs or auxiliary MIPs**
- Primal heuristics
  a) Typically have an approximation ratio
  b) Always improve the primal bound
  c) **Might terminate unsuccessfully**

Large Neighborhood Search

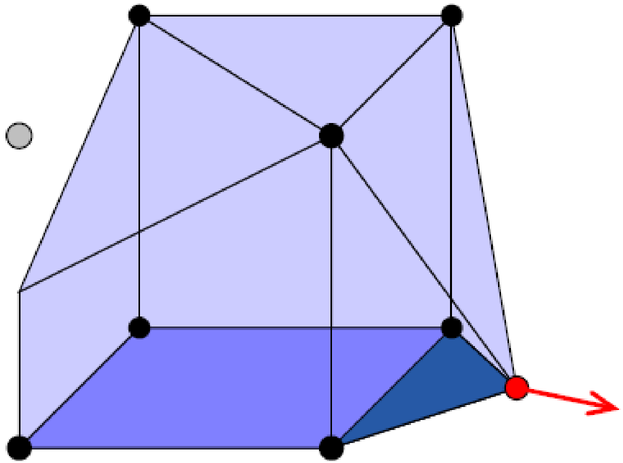
Large Neighborhood Search

1. Consider one or several reference solutions
2. Create an auxiliary problem that is too hard to solve by enumeration („large“)
   - Feasible set is a subset of original
3. According to some neighborhood definition:
   - Fix variables
   - Add constraints
   - Change the objective
4. Perform a partial solve
   - Might use LNS inside LNS

RENS (Berthold 2014)

Idea: Search vicinity of a relaxation solution

1. Reference point:
   $\bar{x} \leftarrow$ LP optimum
2. Fix all integral variables:
   $x_i \leftarrow \bar{x}_i \quad \forall i\colon \bar{x}_i \in \mathbb{Z}$
3. Reduce domain of fractional variables:
   $x_i \in \{\lfloor \bar{x}_i \rfloor, \lceil \bar{x}_i \rceil\}$
4. Solve the resulting sub-MIP



Start heuristic, does not need a feasible solution!

RINS (Danna et Al 2005)

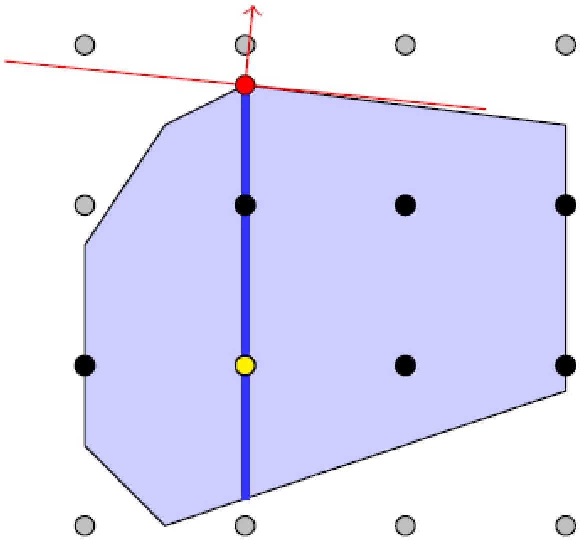
Idea: Search common vicinity relaxation solution and incumbent

- Close to LP optimum: high quality
- Close to incumbent: feasible

1. Reference points:
   $\bar{x} \leftarrow$ LP optimum
   $\tilde{x} \leftarrow$ incumbent

2. Fix coinciding variables:
   $x_i \leftarrow \bar{x}_i \;\; \forall i: \bar{x}_i = \tilde{x}_i$

3. Solve the resulting sub-MIP



Most common sub-MIP heuristic?

Local Branching (Fischetti&Lodi 2003)

Idea: Search vicinity, induced by 1-norm, of incumbent

- Soft rounding
- Might require auxiliary variables

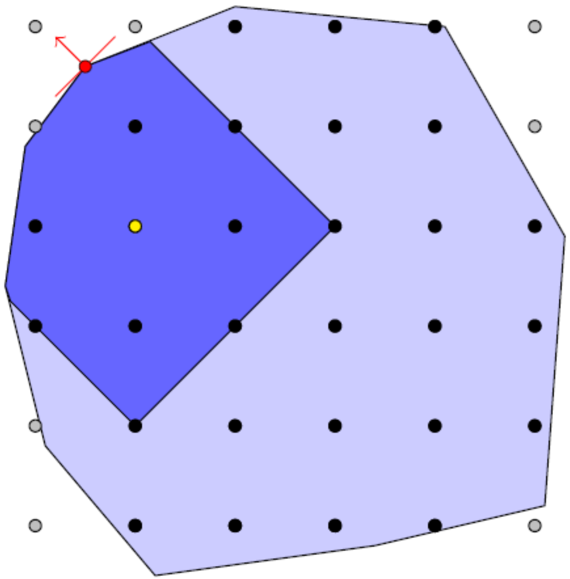
1. Reference points:

   $\tilde{x} \leftarrow$ incumbent

2. Impose Local Branching Constraint:

   $$\Delta(x, \tilde{x}) = \sum \left| x_j - \tilde{x}_j \right| \leq k$$

3. Solve the resulting sub-MIP



Originally suggested as a branching strategy

Crossover (Rothberg 2007)

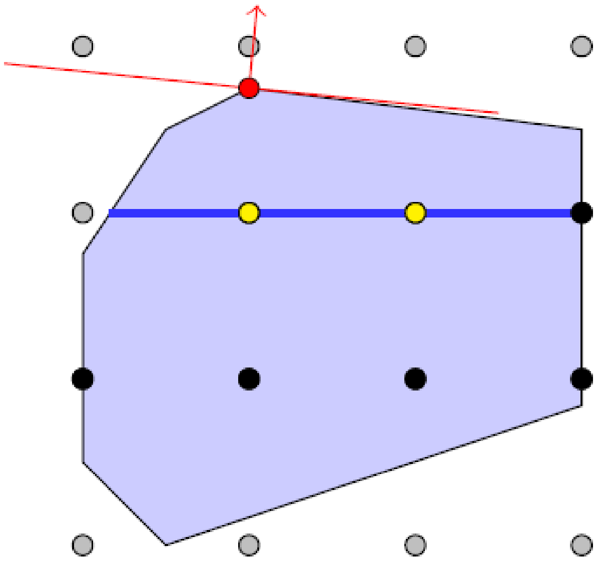
Idea: Search vicinity of several feasible solutions

- Detect implicit conditions for feasibility
- Might be self-fulfilling prophecy

1. Reference points:
   $\tilde{X} \leftarrow$ set of solutions

2. Fix all agreeing variables:
   $x_i \leftarrow \tilde{x}_i^1 \;\; \forall i: \tilde{x}_i^j = \tilde{x}_i^k \;\; \forall \tilde{x}^j, \tilde{x}^k \in \tilde{X}$

3. Solve the resulting sub-MIP



Originally part of a genetic algorithm, with a randomized Mutation LNS heuristic

Changing the Objective

- Drop it
  - Zero Objective or Hail Mary Heuristic
  - Might allow for many additional fixings
- Inverse it
- Use Local Branching constraint as objective (Fischetti&Monaci 2014)
  - Try to optimize towards a reference point
  - Reference point can be an „almost“ feasible solution
- Use Analytic Center as objective (Berthold et al 2018)
  - Indicates the direction into which a variable is likely to move toward feasibility
  - Particularly interesting for variables that are likely to be 1 in a binary problem

Graph Induced Neighborhood Search (Maher et al. 2017)

- Fix all variables outside the „constraint neighborhoods“ of one or several central variables
- Consider Variable-constraint graph:

$$G_A := (V = \{v_1, \ldots, v_n\}, W = \{w_1, \ldots, w_m\}, E = \{(v_i, w_j) \in V \times W : a_{ij} \neq 0\})$$

- k-neighborhood of a variable s:

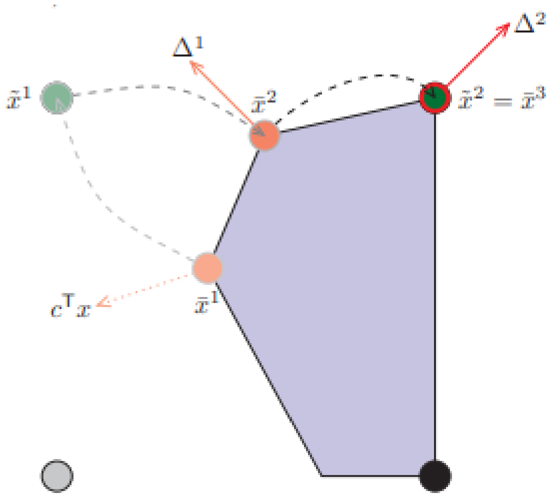
$$N_k(s) := \{t \in V : d(s,t) \leq 2k\}$$

- Fix all variables outside the k-neighborhood
- Choose maximum k to stay above a minimum fixing rate
- Alternatively, can be applied on top of other fixing rules to reach a target fixing rate
  - Fix all variables INSIDE k-neighborhood

Feasibility Pump

Basic Feasibility Pump (Fischetti et Al 2005)

1. Solve original LP
2. Round LP optimum
3. If feasible:
   - Stop!
4. If cycle:
   - Perturb
5. Change objective: $\Delta(x, \tilde{x}) = \sum |x_j - \tilde{x}_j|$
6. Solve LP (project)
7. Goto 2



Main Variants

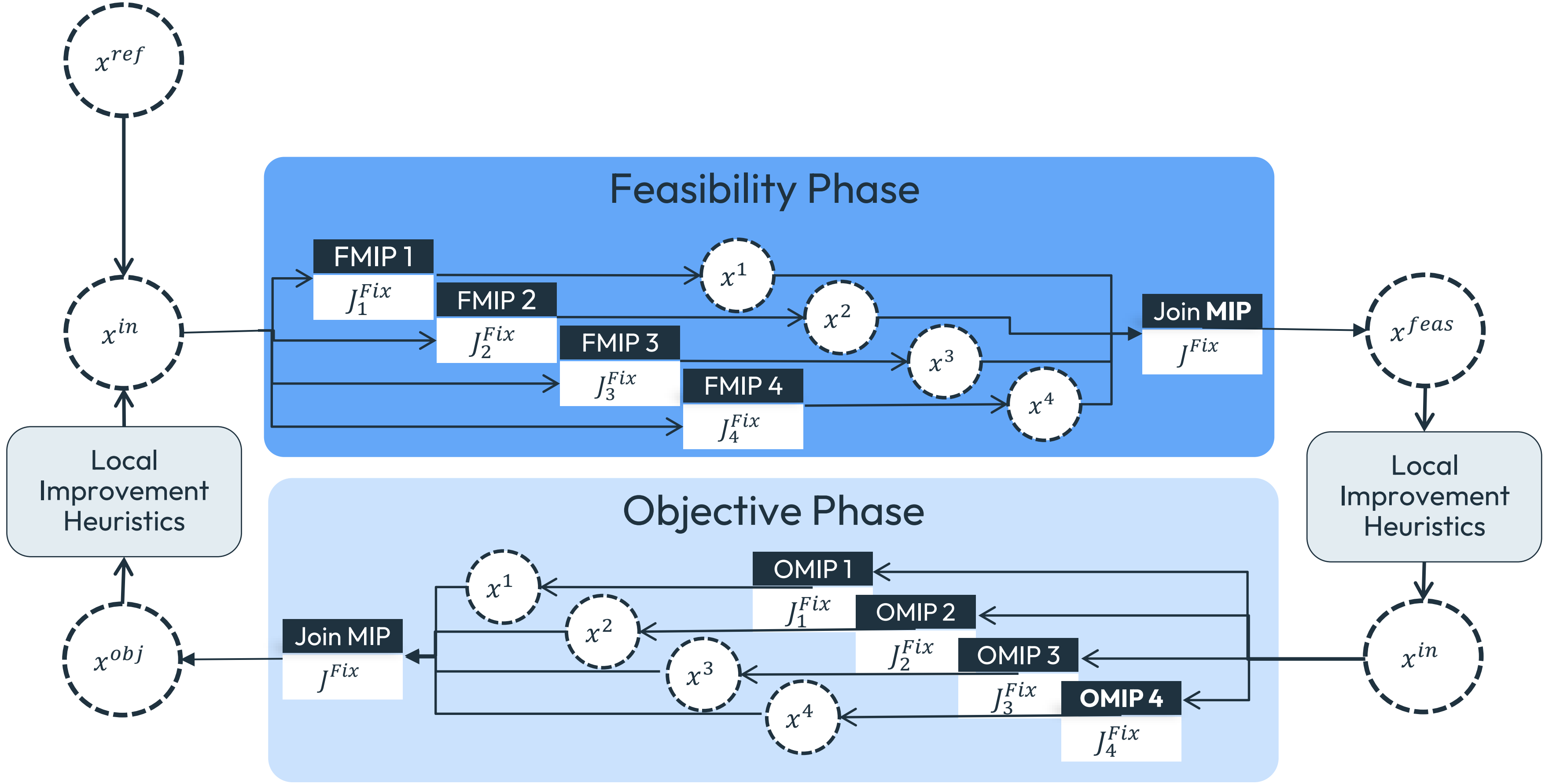
- Improved feasibility pump (Bertacco et al 2007)
  - Uses auxiliary variables to model distance function on general integers

$$\Delta(x,\tilde{x}) := \sum_{j\in\mathcal{I}:\ \tilde{x}_j=l_j}(x_j-l_j) + \sum_{j:\ \tilde{x}_j=u_j}(u_j-x_j) + \sum_{j\in\mathcal{I}:\ l_j<\tilde{x}_j<u_j} d_j \quad \text{s.t.} \quad \mathrm{d}_j \geq x_j-\tilde{x}_j,\ \ \mathrm{d}_j \geq \tilde{x}_j-x_j$$

- Objective feasibility pump (Achterberg & Berthold 2007)
  - Convex combination of original objective and distance function
    - $\widetilde{\Delta} = (1-\alpha)\Delta(\mathrm{x}) + \alpha c^T x$, with $\alpha \in [0,1]$, typically $\alpha \in [0.95,0.99]$
    - Better (Mexi et al. 2023): scales both objectives by their norms
  - Algorithm can recover from cycles
- Feasibility Pump 2.0 (Fischetti & Salvagnin 2009)
  - Applies propagation after each rounding
  - Uses specific propagators for special linear constraints
  - fewer rounding steps, “more feasible”

Parallel (Pre-)Root Heuristics

Parallel Search Heuristics



Key Algorithmic Ingredients

- Portfolio heuristic with focus on finding solutions before the initial LP solve
- Starting solutions come from a solution pool filled, e.g., by quick sequential heuristics (before LP), or, if there is none, from a random initialization. It does not require an LP relaxation solution
- Solves a series of feasibility and objective MIPs as suggested in Alternating Criteria Search
- Both phases solve single-threaded sub-MIPs in parallel to improve the phase objective, namely feasibility or objective function
- At the end of each phase, another sub-MIP is solved that combines all solutions in a crossover fashion
- Diversification is key and is mainly achieved through randomization
- Fixing strategy based on row violations, diversifying the sets across the sub-MIPs
- Propagation and constructive heuristics such as Shift-and-Propagate or Feasibility Jump or Local Search for MIP are employed regularly in parallel to further decrease violations

T. Berthold & G. Hendel: *Shift-and-Propagate*, Journal of Heuristics 21 (2015)

B. Luteberget & G. Sartor: *Feasibility Jump: an LP-free Lagrangian MIP heuristic,* Mathematical Programming Computation 15 (2023)

L. Munguia et. al.: *Alternating criteria search: a parallel large neighborhood search algorithm for mixed integer programs,* Computational Optimization and Applications 69 (1) (2018)

Parallel Background Tasks

Concept

- At the root node of the tree search, by default MIP solvers can run several "auxiliary" tasks as background jobs in parallel to the root cut loop
  - Background tasks are primarily used for finding heuristic solutions (e.g., to run expensive heuristics), but also to improve the dual bound of the overall solution process
  - Information found by background tasks is (deterministically) shared with the main foreground task
- The solver selects the tasks to run depending on number of available threads and on problem characteristics
- E.g., for Xpress, auxiliary tasks that can be run as background jobs are:
  - Fast branch-and-bound search with cutting planes disabled
  - Local search exploring a neighborhood centered around the node LP solution
  - Feasibility Jump heuristic **[Luteberget and Sartor, MPC 2023]**
  - Fix-Propagate-Repair heuristic **[Salvagnin, Roberti and Fischetti, MPC 2025]**
  - 0-objective heuristic

Measuring the Impact of Primal Heuristics

Performance Measures

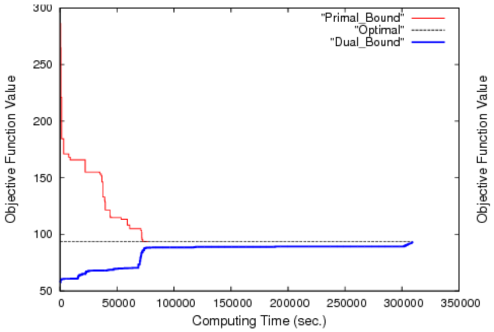
How to measure the added value of a primal heuristic?

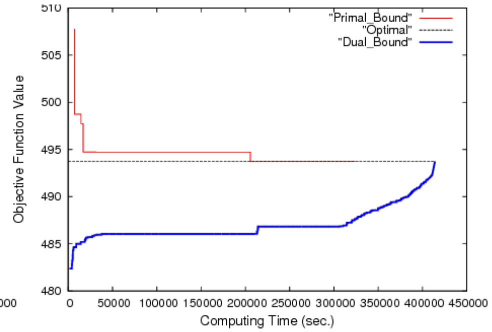
- time to optimality, number of branch-and-bound nodes
  - very much depends on dual bound
- time to first solution
  - disregards solution quality
- time to best solution
  - nearly optimal solution might be found long before
- Some combination of all of those?

Primal Integral / Primal-Dual Integral (Berthold 2013)

$$\gamma^p(\tilde{x}) := \begin{cases} 0, & \textit{if } c^\top x^\star = c^\top \tilde{x} = 0, \\ 1, & \textit{if } c^\top x^\star \cdot c^\top \tilde{x} < 0, \\ \frac{|c^\top x^\star - c^\top \tilde{x}|}{\max\{|c^\top x^\star|,\ |c^\top \tilde{x}|\}}, & \textit{otherwise.} \end{cases}$$

$$P(T) := \int_{t=0}^{T} \gamma^p(t)\, dt = \sum_{i=1}^{I} \gamma^p(t_{i-1}) \cdot (t_i - t_{i-1})$$



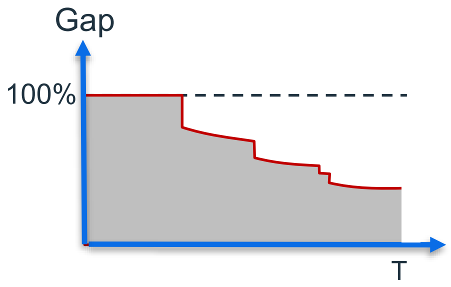


Primal-Dual Integral (PDI):

- favors finding good solutions quickly
- considers each update of incumbent (and the best bound)
- gives you expected solution quality assuming unknown termination time

Heuristic Emphasis

- Textbook: The beauty of MIP is that we can solve problems to proven optimality
- Practice: Often, this takes unbearably long
  - Still, MIP solvers provide us with a proof of quality, the gap
  - Often, solving models to a small gap is good enough
    - And the only viable option within restrictive time limits



- Heuristic Emphasis mode (HEUREMPHASIS)
  - Focus on reducing the gap in the beginning of a MIP search
    - More precisely: minimizes the Primal-Dual Integral (PDI)
  - Typically at the expense of a longer time to optimality
  - Targets problems out of scope for solving to optimality, but when finding good solutions early matters

Heuristic Emphasis: Technology

- Many neighborhood search heuristics at the root
- Additional non-default heuristics
- More frequent heuristics in the tree
- Deactivate some techniques specifically tailored for easy problems
- Background heuristics in parallel to root-node search

Quiz Time

- LNS stands for
  a) Large neighborhood search
  b) Local neighborhood search
  c) Local native search
- What is the idea of Crossover?
  a) Fix variables that coincide in incumbent and LP optimum
  b) Add one constraint for each fractional variable
  c) Fix variables that coincide in a set of integer solutions
- The primal dual integral
  a) Considers each update of the incumbent
  b) Depends on the number of processed nodes
  c) Measures the time to find a first solution

Quiz Time

- LNS stands for
  a) Large neighborhood search
  b) Local neighborhood search
  c) Local native search
- What is the idea of Crossover?
  a) Fix variables that coincide in incumbent and LP optimum
  b) Add one constraint for each fractional variable
  c) Fix variables that coincide in a set of integer solutions
- The primal dual integral
  a) Considers each update of the incumbent
  b) Depends on the number of processed nodes
  c) Measures the time to find a first solution

Quiz Time

- LNS stands for
  a) Large neighborhood search
  b) Local neighborhood search
  c) Local native search
- What is the idea of Crossover?
  a) Fix variables that coincide in incumbent and LP optimum
  b) Add one constraint for each fractional variable
  c) Fix variables that coincide in a set of integer solutions
- The primal dual integral
  a) Considers each update of the incumbent
  b) Depends on the number of processed nodes
  c) Measures the time to find a first solution

Quiz Time

- LNS stands for
  a) **Large neighborhood search**
  b) Local neighborhood search
  c) Local native search
- What is the idea of Crossover?
  a) Fix variables that coincide in incumbent and LP optimum
  b) Add one constraint for each fractional variable
  c) **Fix variables that coincide in a set of integer solutions**
- The primal dual integral
  a) **Considers each update of the incumbent**
  b) Depends on the number of processed nodes
  c) Measures the time to find a first solution

Thank You!

PD Dr. Timo Berthold
Private Lecturer @ TU Berlin
Director, Mixed-Integer Optimization @ FICO