Mixed Integer Programming Solvers

Mauriana Pesaresi Seminar Series

Federica Di Pasquale
federica.dipasquale@phd.unipi.it
25/03/2022

Dipartimento di Informatica
Università di Pisa

Outline

Mixed Integer Programming (MIP)

General context: Optimization

Figure 1: Energy Systems

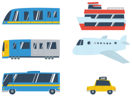


Figure 2: Public transport

Figure 3: Supply Chain

Optimization problem

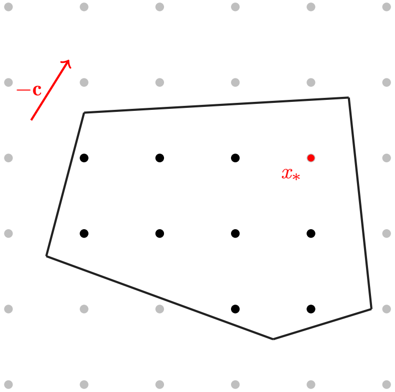
$$(P) \qquad f_* = \min\{\, f(\mathbf{x}) \;:\; \mathbf{x} \in \mathbb{X} \,\}$$

Mixed Integer (Linear) Program

$$
\begin{array}{lll}
\text{(P)} & \min & \mathbf{c}^\intercal \mathbf{x} \\
& s.t. & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\
& & \mathbf{x} \in \mathbb{R}^{n-p} \times \mathbb{Z}^p
\end{array}
$$

- Strong expressive power
- NP-hard

Visualization

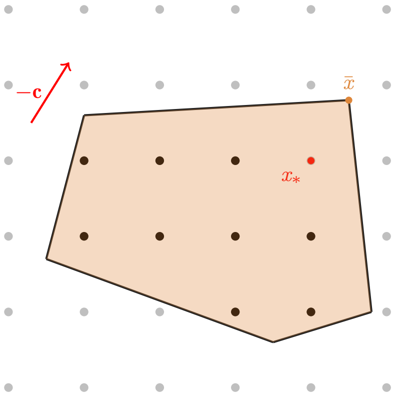


- Convex Polyhedron:

$$\mathcal{P} = \{\, \mathbf{x} \in \mathbb{R}^n \mid \mathbf{A}\mathbf{x} \leq \mathbf{b} \,\}$$

- Integer optimal solution x_*

Visualization

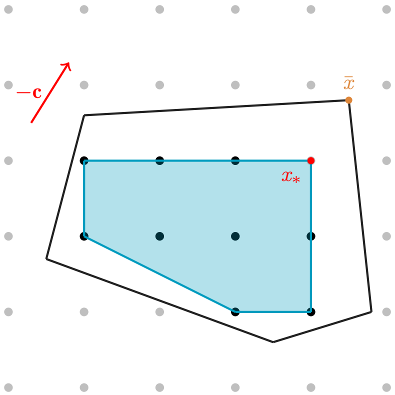


- Convex Polyhedron:

$$\mathcal{P} = \{\, \mathbf{x} \in \mathbb{R}^n \mid \mathbf{A}\mathbf{x} \leq \mathbf{b} \,\}$$

- Integer optimal solution x_*
- LP relaxation solution $\bar{x}$

Visualization



- Convex Polyhedron:

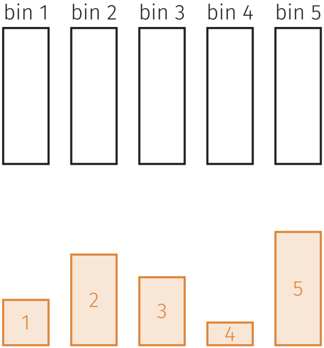
$$\mathcal{P} = \{\, \mathbf{x} \in \mathbb{R}^n \mid \mathbf{A}\mathbf{x} \leq \mathbf{b} \,\}$$

- Integer optimal solution x_*
- LP relaxation solution $\bar{x}$
- Convex-Hull

$$\mathcal{P}_1 = conv(\mathcal{P} \cap \mathbb{Z}^n)$$

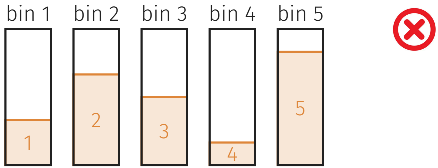
MIP example: The Bin Packing problem

- n bins with capacity c
- n items with weights w_i
- pack the items **minimizing** the number of used bins



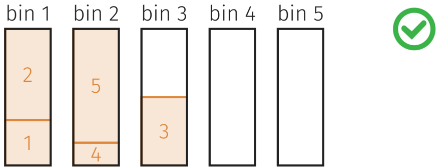
MIP example: The Bin Packing problem

- n bins with capacity c
- n items with weights w_i
- pack the items **minimizing** the number of used bins

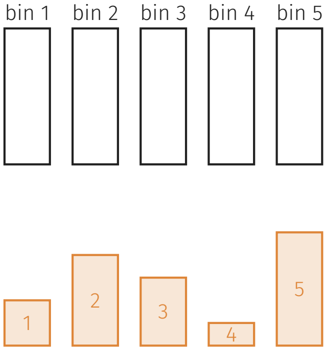


MIP example: The Bin Packing problem

- n bins with capacity c
- n items with weights w_i
- pack the items **minimizing** the number of used bins



MIP example: The Bin Packing problem



$$x_{ij} = \begin{cases} 1 & \text{if item } i \text{ is put into bin } j \\ 0 & \text{otherwise} \end{cases}$$

$$y_j = \begin{cases} 1 & \text{if bin } j \text{ is used} \\ 0 & \text{otherwise} \end{cases}$$

The Bin Packing problem

$$\min \sum_{j=1}^{n} y_j$$

$$s.t. \sum_{i=1}^{n} w_i x_{ij} \leq c y_j \qquad \forall j$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i$$

$$y_j \in \{0,1\} \quad x_{ij} \in \{0,1\}$$

MIP example: The Bin Packing problem - computational results

		n 57	119	239	400
gap [%]	CBC	6.7	7.7	4.8	20.1
	GLPK	6.7	7.7	4.8	20.1
	CPLEX	6.7	7.7	4.8	13.2

Table 1: The Bin Packing problem - Time limit = 1800 sec

State-of-the-art solvers:

- CPLEX: https://www.ibm.com/analytics/cplex-optimizer
- Gurobi: https://www.gurobi.com
- FICO Xpress: https://www.fico.com/en/products/fico-xpress-solver
- Mosek: https://www.mosek.com
- SCIP: https://www.scipopt.org
- CBC: https://github.com/coin-or/Cbc
- GLPK: https://www.gnu.org/software/glpk/
- many others ...

Solving a MIP

Basic algorithms

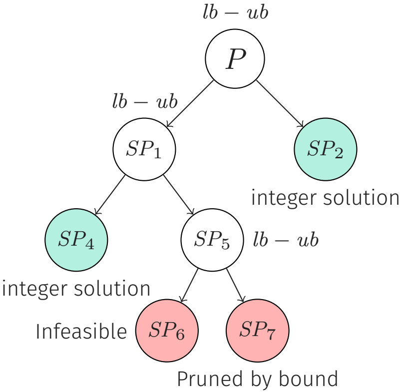
- Branch and Bound
- Cutting Planes
- Branch and Cut

Branch and Bound

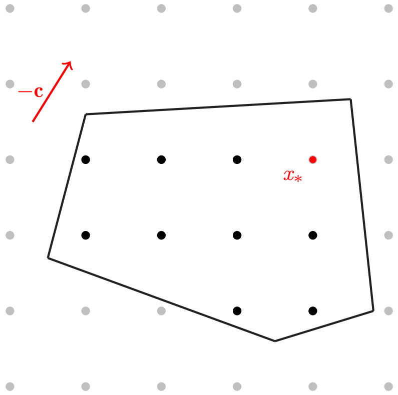
Build a search tree ***implicitly*** *enumerating all the candidate solutions to find an optimal one*

Main ingredients:

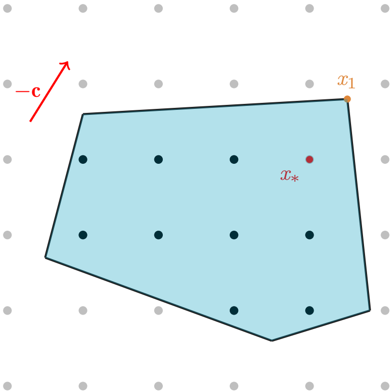
- **Bounding** to compute upper/lower bounds
- **Branching Rules** to create nodes
- **Pruning Rules** to prune a node
- **Search strategy** to visit the search tree



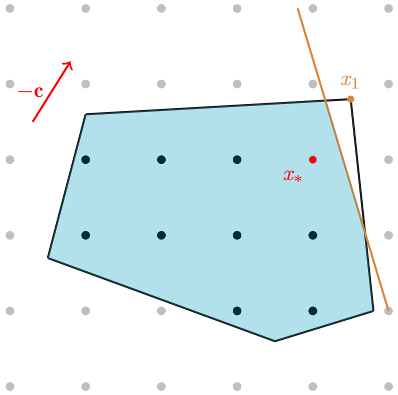
Cutting plane algorithm



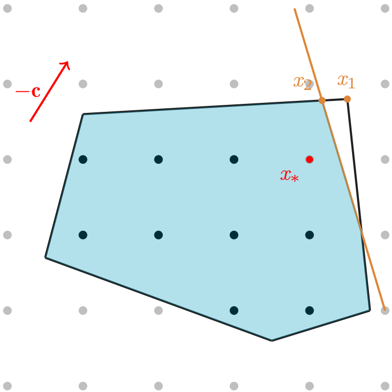
Cutting plane algorithm



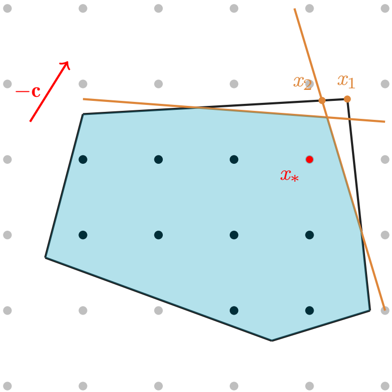
Cutting plane algorithm



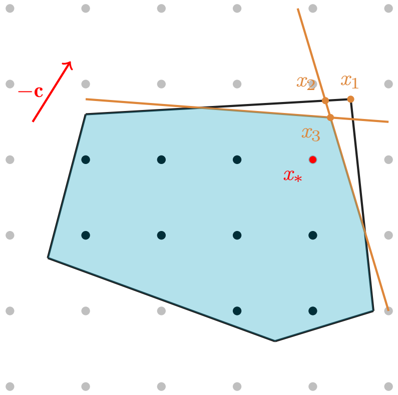
CUTTING PLANE ALGORITHM



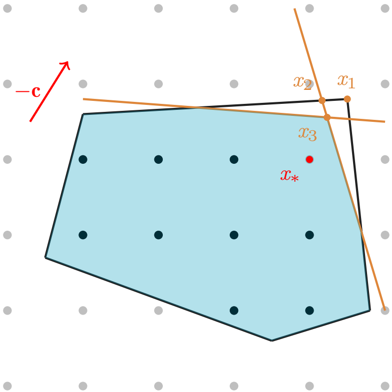
Cutting plane algorithm



Cutting plane algorithm



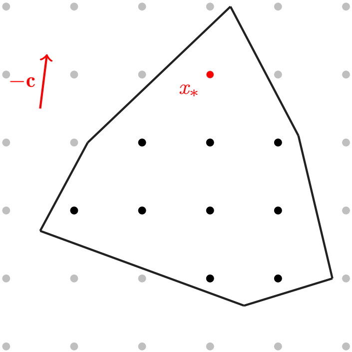
Cutting plane algorithm



The sequence of points $x_1, x_2, \ldots$ converges to x_*

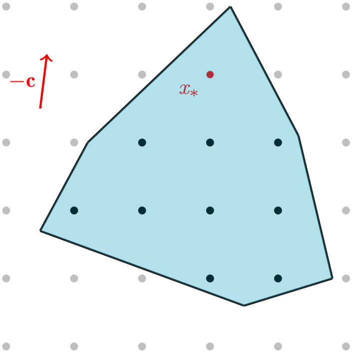
Branch and Cut

- Select Node
- Relaxation
- Cuts
- Pruning Rules
- Branch



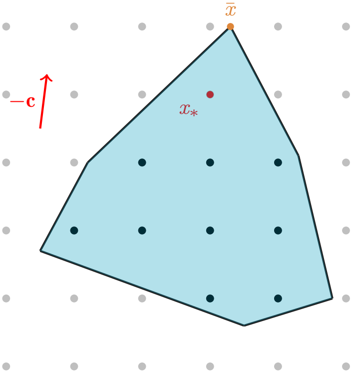
Branch and Cut

- Select Node ←
- Relaxation
- Cuts
- Pruning Rules
- Branch



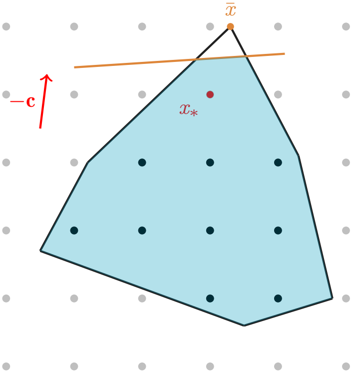
Branch and Cut

- Select Node
- Relaxation $\leftarrow$
- Cuts
- Pruning Rules
- Branch



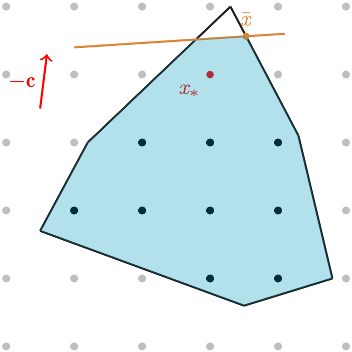
Branch and Cut

- Select Node
- Relaxation
- Cuts $\leftarrow$
- Pruning Rules
- Branch



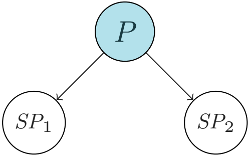
Branch and Cut

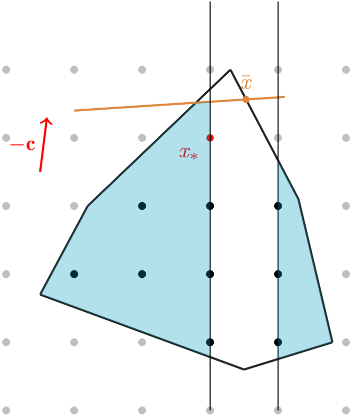
- Select Node
- Relaxation ←
- Cuts
- Pruning Rules
- Branch



Branch and Cut

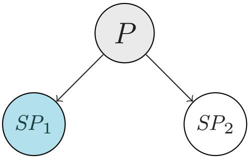
- Select Node
- Relaxation
- Cuts
- Pruning Rules
- Branch ←

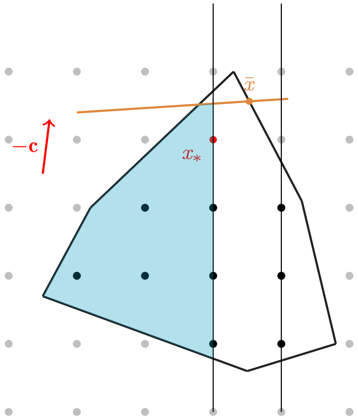




Branch and Cut

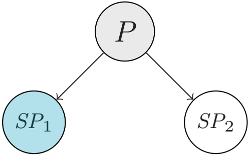
- Select Node ←
- Relaxation
- Cuts
- Pruning Rules
- Branch

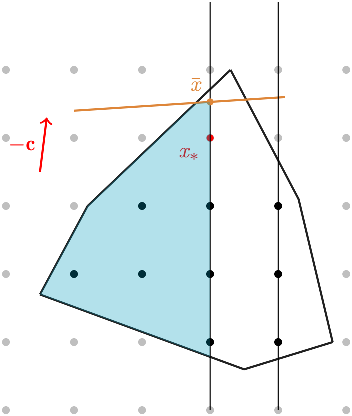




Branch and Cut

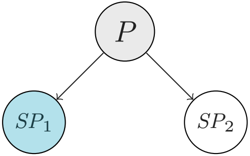
- Select Node
- Relaxation $\leftarrow$
- Cuts
- Pruning Rules
- Branch

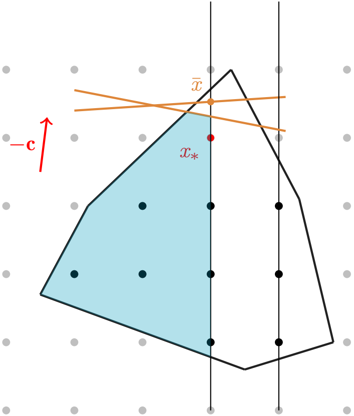




Branch and Cut

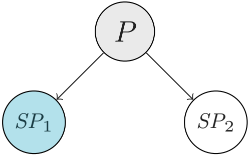
- Select Node
- Relaxation
- Cuts ←
- Pruning Rules
- Branch

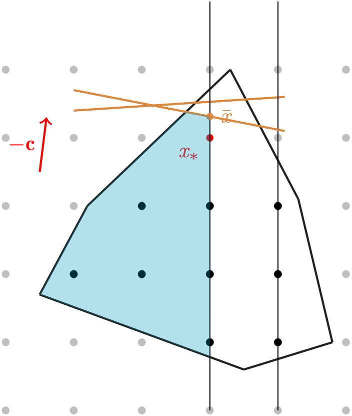




Branch and Cut

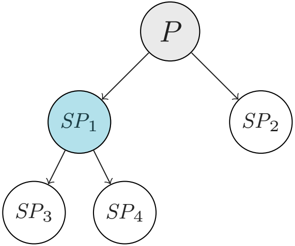
- Select Node
- Relaxation ←
- Cuts
- Pruning Rules
- Branch

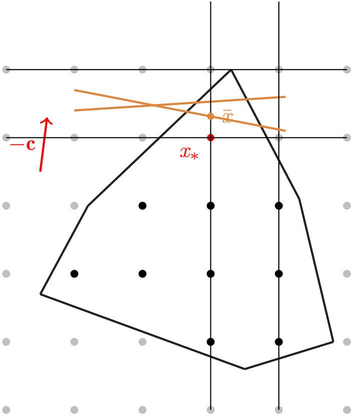




Branch and Cut

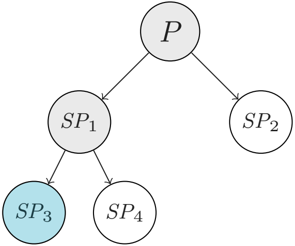
- Select Node
- Relaxation
- Cuts
- Pruning Rules
- Branch $\leftarrow$

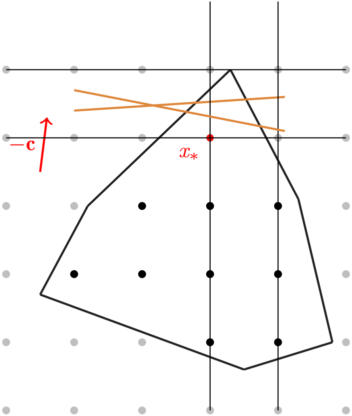




Branch and Cut

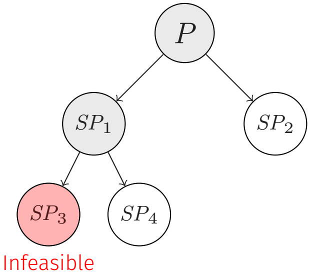
- Select Node ←
- Relaxation
- Cuts
- Pruning Rules
- Branch

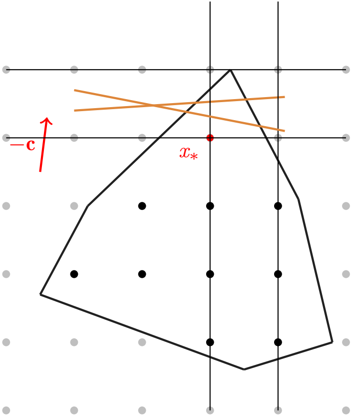




Branch and Cut

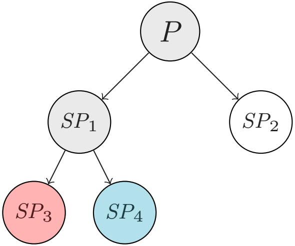
- Select Node
- Relaxation
- Cuts
- Pruning Rules ←
- Branch

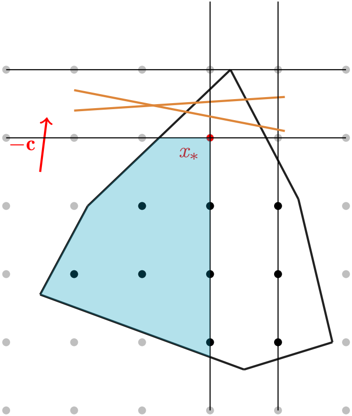




Branch and Cut

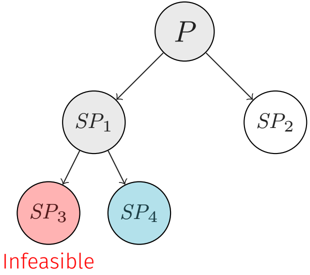
- Select Node $\leftarrow$
- Relaxation
- Cuts
- Pruning Rules
- Branch

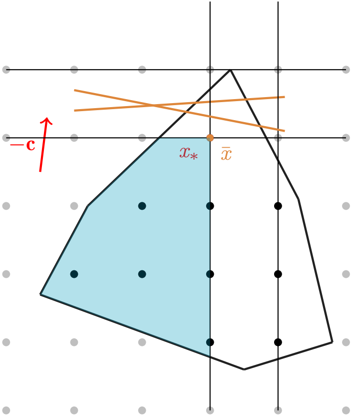




Branch and Cut

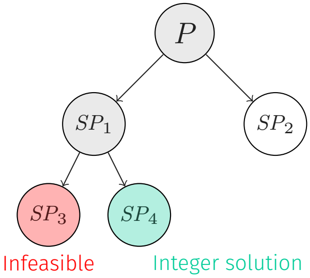
- Select Node
- Relaxation ←
- Cuts
- Pruning Rules
- Branch

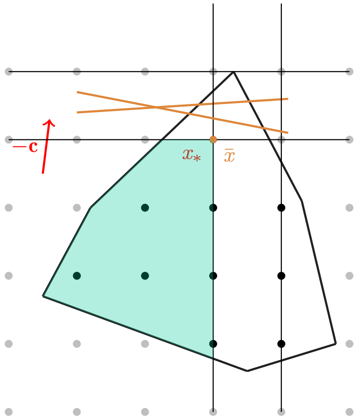




Branch and Cut

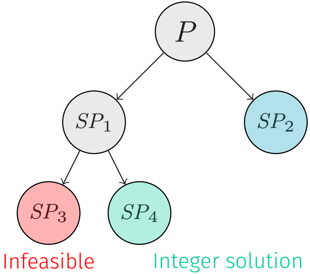
- Select Node
- Relaxation
- Cuts
- Pruning Rules ←
- Branch

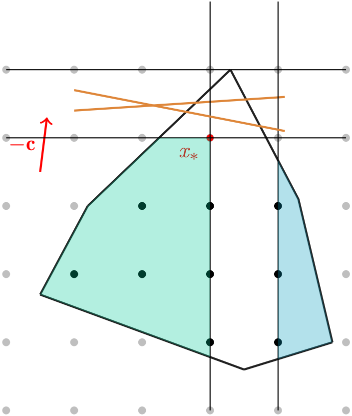




Branch and Cut

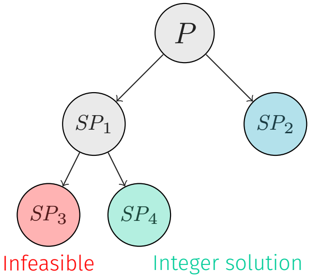
- Select Node ←
- Relaxation
- Cuts
- Pruning Rules
- Branch

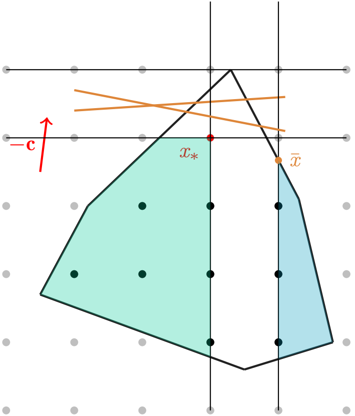




Branch and Cut

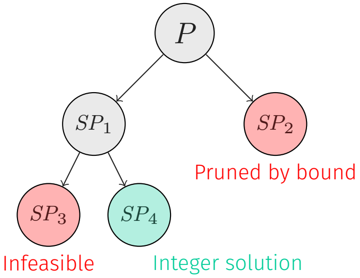
- Select Node
- Relaxation ←
- Cuts
- Pruning Rules
- Branch

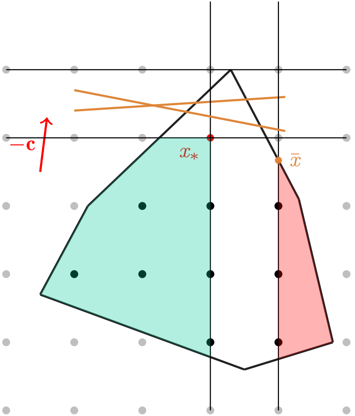




Branch and Cut

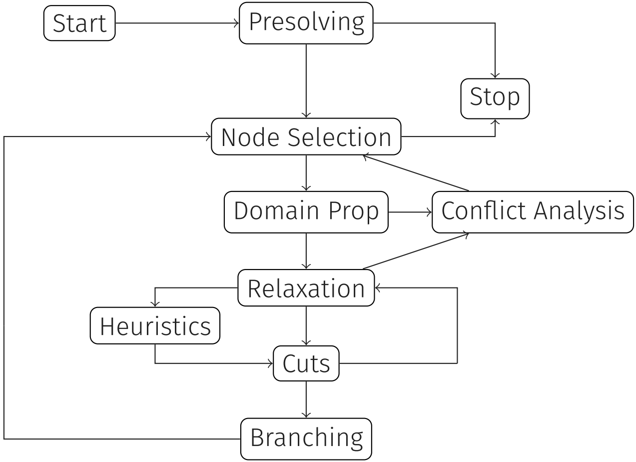
- Select Node
- Relaxation
- Cuts
- Pruning Rules ←
- Branch





MIP solvers

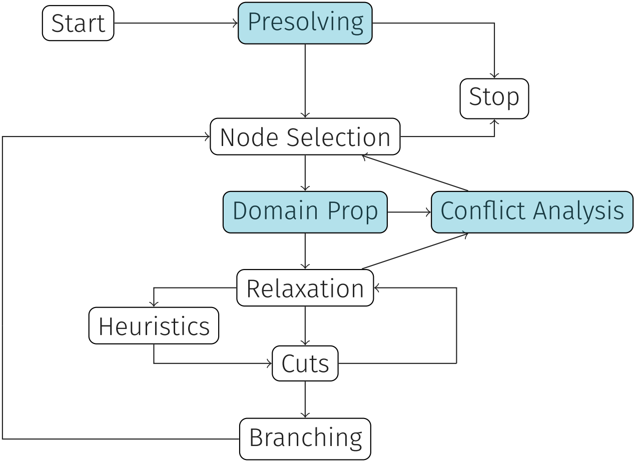
MIP solver: Flowchart



http://co-at-work.zib.de

Tobias Achterberg. “Constraint Integer Programming”. PhD thesis. Universität Berlin, June 12, 2007

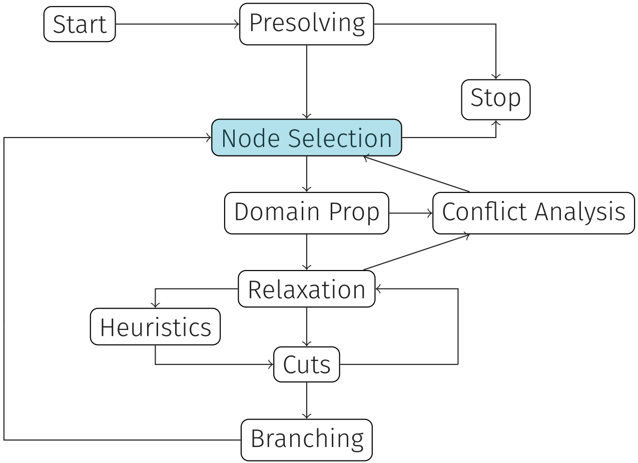
MIP solver: Flowchart



`http://co-at-work.zib.de`

Tobias Achterberg. “Constraint Integer Programming”. PhD thesis. Universität Berlin, June 12, 2007

MIP solver: Flowchart



http://co-at-work.zib.de

Tobias Achterberg. “Constraint Integer Programming”. PhD thesis. Universität Berlin, June 12, 2007

Node selection strategies

Two goals:

Improve the lower bound

Better lower bounds usually close to the root node

Best first search = select a leaf with the current smallest lower bound

- it leads to a minimal number of nodes to be processed

Improve the upper bound

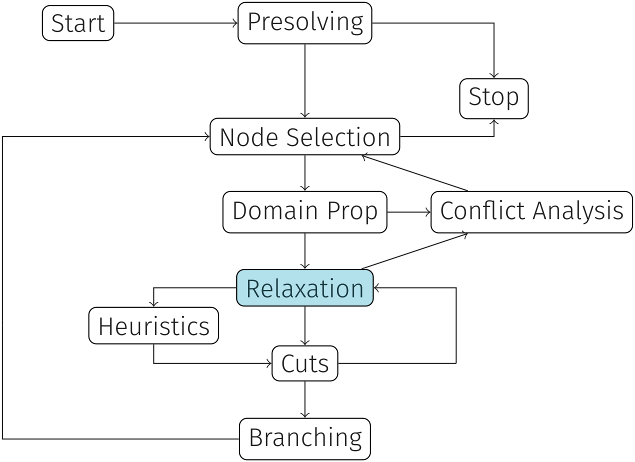
Feasible solutions usually found very deep in the search tree

Depth first search = select child of the current node

- Best approach to identify feasible solutions
- Reoptimization more easily to implement
- Smaller memory consumption

Hybrid approaches: e.g. **Best first search with plunging**

MIP solver: Flowchart



http://co-at-work.zib.de

Tobias Achterberg. "Constraint Integer Programming". PhD thesis. Universität Berlin, June 12, 2007

Relaxation: Lagrangian relaxation I

Given (P) with the following structure:

$$(P) \quad \min\{\, \mathbf{c}^\intercal \mathbf{x} : \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{C}\mathbf{x} \leq \mathbf{d}, x_i \in \mathbb{Z} \,\} \tag{1}$$

$\mathbf{A}\mathbf{x} \leq \mathbf{b} \equiv$ "complicating" constraints. Lagrangian Relaxation:

$$(P_{\boldsymbol{\lambda}}) \quad c^*_{\boldsymbol{\lambda}} = \min\{\, \mathbf{c}^\intercal \mathbf{x} + \boldsymbol{\lambda}^\intercal (\mathbf{b} - \mathbf{A}\mathbf{x}) : \mathbf{C}\mathbf{x} \leq \mathbf{d}, x_i \in \mathbb{Z}\} \tag{2}$$

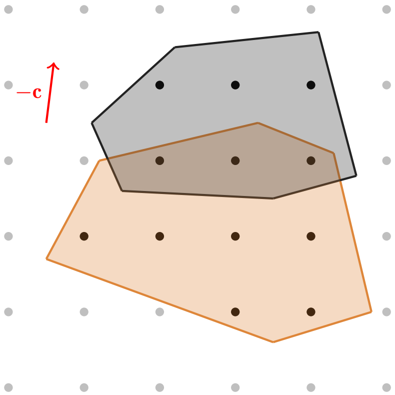
$\boldsymbol{\lambda} \geq 0 \equiv$ (Lagrangian Multipliers)

Best bound for $\boldsymbol{\lambda}$ solution of the Lagrangian Dual Problem

$$(D) \quad \max\{\, \mathcal{L}(\boldsymbol{\lambda}) \equiv c^*_{\boldsymbol{\lambda}} : \boldsymbol{\lambda} \geq 0 \,\} \tag{3}$$

$\mathcal{L}(\boldsymbol{\lambda}) \equiv$ Lagrangian Function.

Relaxation: Lagrangian relaxation II

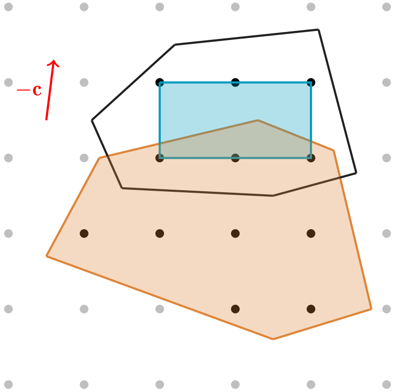


- Given (P):

  $$\min \{ \mathbf{c}^\intercal \mathbf{x} \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{C}\mathbf{x} \leq \mathbf{d}, x_i \in \mathbb{Z} \}$$

- LP Relaxation
  Orange + *Black*

Relaxation: Lagrangian relaxation II



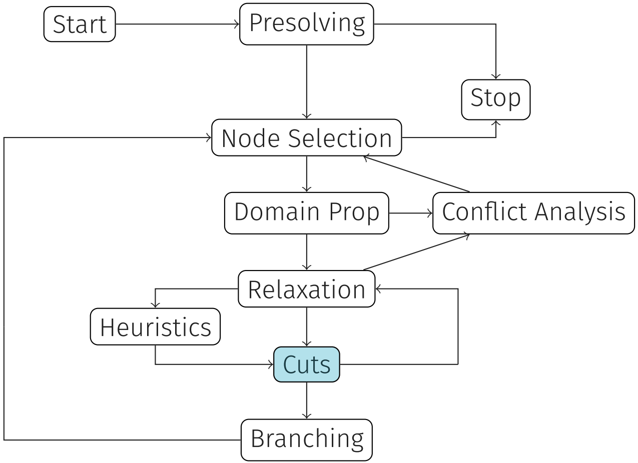
- Given (P):

  $\min\{\, \mathbf{c}^\mathsf{T}\mathbf{x} \mid \mathbf{A}\mathbf{x} \le \mathbf{b}, \mathbf{C}\mathbf{x} \le \mathbf{d}, x_i \in \mathbb{Z} \,\}$

- LP Relaxation
  Orange + *Black*

- Lagrangian Relaxation
  Orange + *Blue*

MIP solver: Flowchart



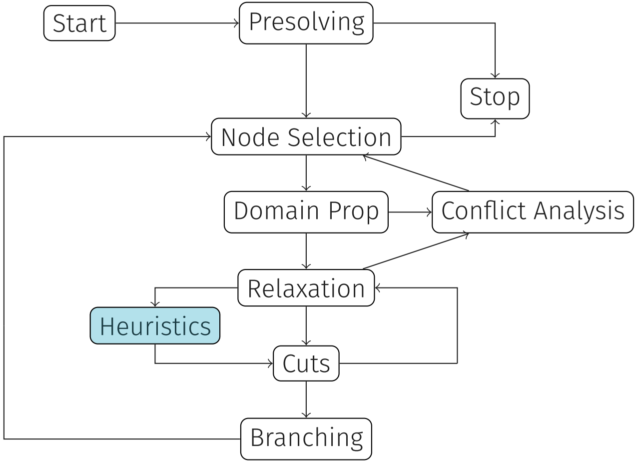
http://co-at-work.zib.de
Tobias Achterberg. "Constraint Integer Programming". PhD thesis. Universität Berlin, June 12, 2007

Cut generation

Three ”cathegories” of cuts:

- **General** cuts, e.g. Gomory cuts
- Cuts that depend on the structure of the problem, e.g. Knapsack Cover cuts
- **Problem-specific** cuts

MIP solver: Flowchart



http://co-at-work.zib.de
Tobias Achterberg. "Constraint Integer Programming". PhD thesis. Universität Berlin, June 12, 2007

Heuristics: Feasibility Pump

Primal Heuristics to find feasible solution.

Example: Feasibility Pump

```
input  : MIP ≡ min{c^T x : x ∈ P, x_j integer ∀j ∈ I}
output: a feasible MIP solution x* (if found)
x* = arg min{c^T x : x ∈ P}
while not termination condition do
    if x* is integer then return x*
    x̃ = Round (x*)
    if cycle detected then Perturb (x̃)
    x* = LinearProj (x̃)
end
```

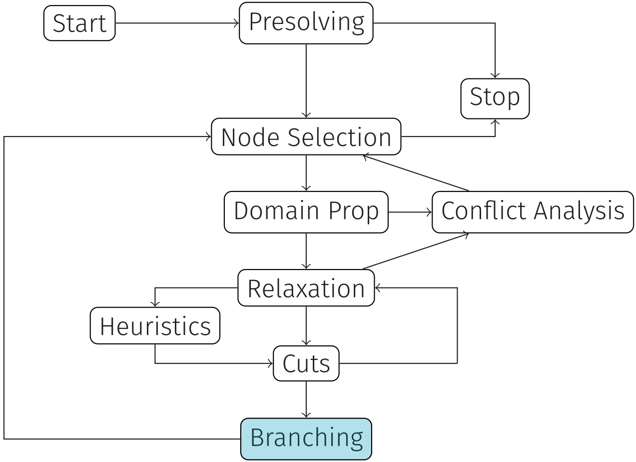
Figure 2: Feasibility Pump—the basic scheme

where **LinearProj** is:

$$x^* = argmin\{\Delta(x, \tilde{x}) \ : \ x \in \mathcal{P}\} \qquad with \qquad \Delta(x, \tilde{x}) = \sum_{j \in I} |x_j - \tilde{x}| \tag{4}$$

Matteo Fischetti, Andrea Lodi, and Fred Glover. "The feasibility pump". In: *Mathematical Programming* 104.1 (Sept. 2005), pp. 91–104. ISSN: 0025-5610, 1436-4646. DOI: 10.1007/s10107-004-0570-3. URL: http://link.springer.com/10.1007/s10107-004-0570-3

MIP solver: Flowchart



---

http://co-at-work.zib.de

Tobias Achterberg. "Constraint Integer Programming". PhD thesis. Universität Berlin, June 12, 2007

Branching rules I

Given a node (SP_i) how to branch? how many children?

Branching on variables

Let $\bar{\mathbf{x}}$ a solution of the relaxation of (SP_i).

$$\bar{\mathbf{x}} \text{ not feasible} \implies \underbrace{C = \{\, i \in \mathcal{I} \mid \bar{x}_i \notin \mathbb{Z} \,\}}_{candidates} \neq \emptyset \tag{5}$$

Select $i \in C$ and create two sub-problems by adding the trivial inequalities:

$$x_i \leq \lfloor \bar{x}_i \rfloor \qquad and \qquad x_i \geq \lceil \bar{x}_i \rceil \tag{6}$$

How to select the variable?

For each x_i with $i \in C$ compute a **score** s_i. Select $i \in C$ with $s_i = \max_{j \in X}\{s_j\}$

Tobias Achterberg, Thorsten Koch, and Alexander Martin. “Branching rules revisited”. In: *Operations Research Letters* 33.1 (Jan. 2005), pp. 42–54. ISSN: 01676377. DOI: 10.1016/j.orl.2004.04.002

Branching rules II

Several strategies depending on how the **score** is computed:

- Most/Least infeasible Branching
- Pseudocost Branching
- Strong Branching
- Hybrid Strong/Pseudocost Branching
- Pseudocost Branching with Strong Branching initialization
- Reliability Branching
- Inference Branching
- Hybrid Reliability/Inference Branching

Tobias Achterberg, Thorsten Koch, and Alexander Martin. "Branching rules revisited". In: *Operations Research Letters* 33.1 (Jan. 2005), pp. 42–54. ISSN: 01676377. DOI: 10.1016/j.orl.2004.04.002

Research Directions

Research Directions

- **Machine learning** techniques
- **Parallel Programming**
- **Decomposition** techniques

Machine Learning techniques

A lot of decisions in a MIP solver are based on heuristics

ML models to speed-up heuristics computation

In this context, we want the overall algorithm to be exact

Several examples:

- Approximating **Strong Branching**
- Approximating optimal **Node Selection**
- Learning when to run **Heuristics**
- Approximating best algorithmic parameters configuration

Still rarely incorporated in a state-of-the-art MIP Solver!

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. "Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon". In: *arXiv:1811.06128 [cs, stat]* (Mar. 12, 2020). arXiv: `1811.06128`. URL: `http://arxiv.org/abs/1811.06128`

Parallel MIP solvers

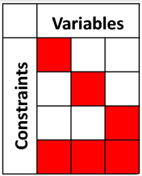
Tree search algorithms seem easy to parallelize.

However, specific challenges in the case of a MIP solver:

- Disprorportionate amount of time processing root node and shallowest nodes
- Dynamic construction of the tree and highly unbalanced
- Order in which the nodes are processed difficult to replicate in parallel
- Huge amount of information to share:
  - Bounds
  - Nodes
  - Solutions
  - Pseudocosts estimates
  - Valid inequalities
  - ...
- Determinism

Ted Ralphs et al. "Parallel Solvers for Mixed Integer Linear Optimization". In: *Handbook of Parallel Constraint Reasoning*. Ed. by Youssef Hamadi and Lakhdar Sais. Cham: Springer International Publishing, 2018, pp. 283–336. ISBN: 978-3-319-63515-6 978-3-319-63516-3. DOI: 10.1007/978-3-319-63516-3_8

Decomposition techniques



If the problem has the appropriate structure $\Longrightarrow$ **decomposition** can be very effective

Many challenges:

- State-of-the-art techniques tailored for LP-based branch and bound
- Difficult to recognize the appropriate structure
- Specialized solvers
- Reoptimization

Antonio Frangioni. "About Lagrangian Methods in Integer Optimization". In: *Annals of Operations Research* 139.1 (Oct. 2005), pp. 163–193. ISSN: 0254-5330, 1572-9338. DOI: 10.1007/s10479-005-3447-9

Thank you for your attention!

References

[1] Tobias Achterberg. “Constraint Integer Programming”. PhD thesis. Universität Berlin, June 12, 2007.

[2] Tobias Achterberg, Thorsten Koch, and Alexander Martin. “Branching rules revisited”. In: *Operations Research Letters* 33.1 (Jan. 2005), pp. 42–54. ISSN: 01676377. DOI: 10.1016/j.orl.2004.04.002.

[3] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. “Machine Learning for Combinatorial Optimization: a Methodological Tour d’Horizon”. In: *arXiv:1811.06128 [cs, stat]* (Mar. 12, 2020). arXiv: 1811.06128. URL: http://arxiv.org/abs/1811.06128.

[4] Matteo Fischetti, Andrea Lodi, and Fred Glover. “The feasibility pump”. In: *Mathematical Programming* 104.1 (Sept. 2005), pp. 91–104. ISSN: 0025-5610, 1436-4646. DOI: 10.1007/s10107-004-0570-3. URL: http://link.springer.com/10.1007/s10107-004-0570-3.

[5] Antonio Frangioni. “About Lagrangian Methods in Integer Optimization”. In: *Annals of Operations Research* 139.1 (Oct. 2005), pp. 163–193. ISSN: 0254-5330, 1572-9338. DOI: 10.1007/s10479-005-3447-9.

[6] Ted Ralphs et al. “Parallel Solvers for Mixed Integer Linear Optimization”. In: *Handbook of Parallel Constraint Reasoning*. Ed. by Youssef Hamadi and Lakhdar Sais. Cham: Springer International Publishing, 2018, pp. 283–336. ISBN: 978-3-319-63515-6 978-3-319-63516-3. DOI: 10.1007/978-3-319-63516-3_8.