Ecole: A Gym-like Library for Machine Learning in Combinatorial Optimization Solvers

Maxime Gasse

Deep Learning and Combinatorial Optimization
IPAM Workshops, Feb. 24

Combinatorial Optimization Solvers

Mixed-Integer Linear Program (MILP)

$$
\begin{aligned}
\operatorname*{arg\,min}_{x} \quad & c^\top x \\
\text{subject to} \quad & Ax \leq b, \\
& x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}.
\end{aligned}
$$

- $c \in \mathbb{R}^n$ the objective coefficients
- $A \in \mathbb{R}^{m \times n}$ the constraint coefficient matrix
- $b \in \mathbb{R}^m$ the constraint right-hand-sides
- $p \leq n$ integer variables

Mixed-Integer Linear Program (MILP)

$$\begin{aligned} \arg\min_{x} \quad & c^\top x \\ \text{subject to} \quad & Ax \leq b, \\ & x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}. \end{aligned}$$

- $c \in \mathbb{R}^n$ the objective coefficients
- $A \in \mathbb{R}^{m \times n}$ the constraint coefficient matrix
- $b \in \mathbb{R}^m$ the constraint right-hand-sides
- $p \leq n$ integer variables

A versatile CO modeling tool

NP-hard !

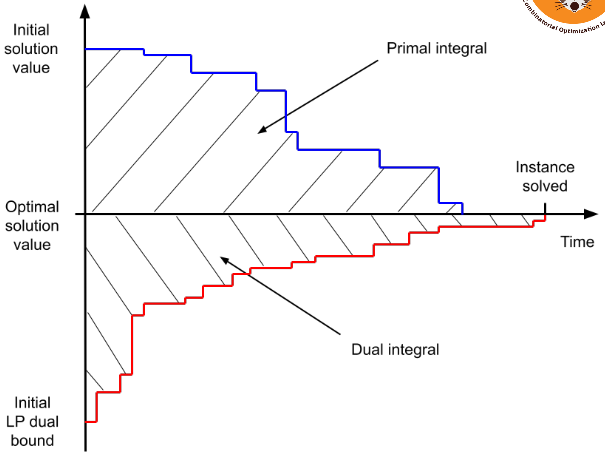
Exact solving ?

The primal side: finding solutions
$\implies$ Upper bound U

The dual side: proving optimality
$\implies$ Lower bound L

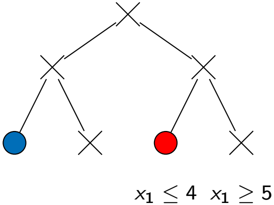
Stopping criterion:

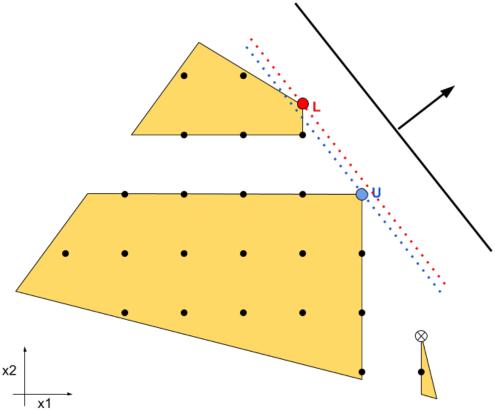
- L = U (optimality certificate)
- L = ∞ (infeasibility certificate)
- L - U < threshold (regret certificate)



Exact algorithms: branch-and-bound, cutting planes, others (application-specific)...

Branch-and-bound recursively decomposes the problem into smaller ones.



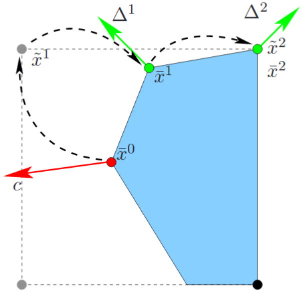


Lower bound (L): minimal among leaf nodes

Upper bound (U): minimal among integral leaf nodes

Decision task: which node to process next ? on which variable(s) to split ?

Primal heuristics (generic search routines) are run at the leaf nodes.

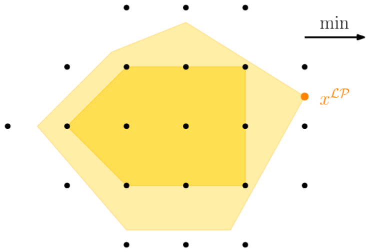


Decision task: which heuristics to run ? When ? (heuristics are costly)

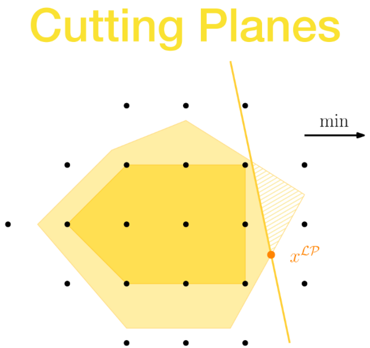
T. Berthold (2006). Primal heuristics for mixed integer programs.

Cuts can be added to the sub-MILPs to tighten the bounds. (Branch-and-cut)

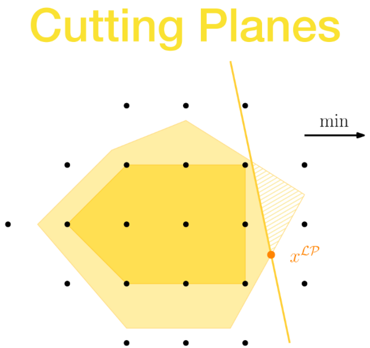
Cutting Planes



Cuts can be added to the sub-MILPs to tighten the bounds. (Branch-and-cut)

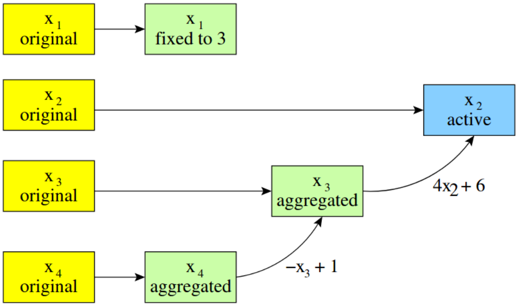


Cuts can be added to the sub-MILPs to tighten the bounds. (Branch-and-cut)



Decision task: which cuts to add to the LP ? Not all cuts are good, some are redundant. Adding too many cuts can lead numerical instabilities.

Preprocessing routines can be run before the solving starts (usually several, sequentially), to simplify and / or tighten the problem formulation.



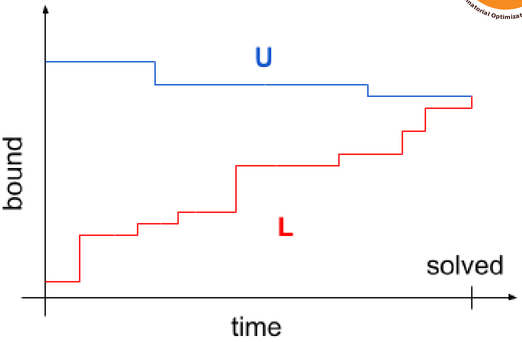
Decision task: which routines to run ? How many times ?

T. Achterberg (2004). SCIP - A Framework to Integrate Constraint and Mixed Integer Programming.

Solver Design: a Complex Control Problem

Many intertwined decisions:

- node selection
- variable selection
- cutting planes
- primal heuristics
- preprocessing
- simplex initialization
- ...



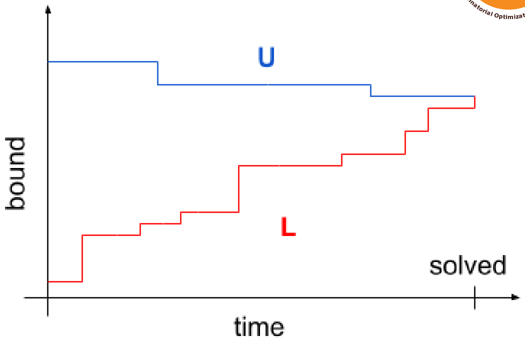
Solver Design: a Complex Control Problem

Many intertwined decisions:

- node selection
- variable selection
- cutting planes
- primal heuristics
- preprocessing
- simplex initialization
- ...

Many evaluation metrics:

- B&B tree size
- solving time: reach U=L fast
- primal-dual integral: U - L $\searrow$ fast
- dual integral: L $\nearrow$ fast
- primal integral: U $\searrow$ fast



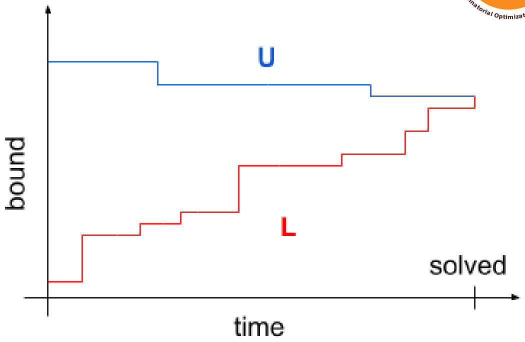
Solver Design: a Complex Control Problem

Many intertwined decisions:

- node selection
- variable selection
- cutting planes
- primal heuristics
- preprocessing
- simplex initialization
- ...

Many evaluation metrics:

- B&B tree size
- solving time: reach U=L fast
- primal-dual integral: U - L ↘ fast
- dual integral: L ↗ fast
- primal integral: U ↘ fast



State of affairs: expert rules + benchmarks.

Ecole: Extensible Combinatorial Optimization Learning Environments

Why Ecole ?

ML4CO: a growing field

Node selection

- [He et al., 2014]
- [Song, Lanka, Zhao, et al., 2018]

Variable selection

- [Khalil, Le Bodic, et al., 2016]
- [Hansknecht et al., 2018]
- [Balcan et al., 2018]
- [Gasse et al., 2019]
- [Gupta et al., 2020]
- [Nair et al., 2020]

Cutting planes selection

- [Baltean-Lugojan et al., 2018]
- [Tang et al., 2019]

Primal heuristic selection

- [Khalil, Dilkina, et al., 2017]
- [Hendel et al., 2018]

Formulation selection

- [Bonami et al., 2018]

Neighborhood search heuristics

- [Ding et al., 2019]
- [Song, Lanka, Yue, et al., 2020]
- [Addanki et al., 2020]

Diving heuristics

- [Song, Lanka, Zhao, et al., 2018]
- [Yilmaz et al., 2020]
- [Nair et al., 2020]

Why Ecole ?

Poor reproducibility in the field

- closed-source solvers
- problem benchmarks
- evaluation metrics

High bar of entry for newcomers

- low-level C/C++ code
- highly technical APIs even for OR experts

Gap between the ML and OR communities

- amputated solvers raise criticism in the OR community
- OR experts employ basic ML models

$\Longrightarrow$ need for a *standard*, *open* platform based on a *state-of-the-art* solver

---

A. Prouvost et al. (2020). Ecole: A Gym-like Library for Machine Learning in Combinatorial Optimization Solvers.

Why Ecole ?

Poor reproducibility in the field

- closed-source solvers
- problem benchmarks
- evaluation metrics

Remove technical obstacles, so that we can focus on the interesting challenges !

High bar of entry for newcomers

- low-level C/C++ code
- highly technical APIs even for OR experts

Gap between the ML and OR communities

- amputated solvers raise criticism in the OR community
- OR experts employ basic ML models

$\Longrightarrow$ need for a *standard*, *open* platform based on a *state-of-the-art* solver

A. Prouvost et al. (2020). Ecole: A Gym-like Library for Machine Learning in Combinatorial Optimization Solvers.

The PO-MDP Formulation

Sequential control problem = **Markov decision process**

State = state of the branch-and-bound process (solver)
Actions = variables, nodes, primal heuristics, cuts, preprocessing routines to select
Episode = solving an instance to completion

$$\tau \sim \underbrace{p_{init}(s_0)}_{\text{initial state}} \prod_{t=0}^{\infty} \underbrace{\pi(a_t|s_t)}_{\text{next action}} \underbrace{p_{trans}(s_{t+1}|a_t, s_t)}_{\text{next state}}$$

The PO-MDP Formulation

Sequential control problem = **Markov decision process**

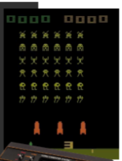
State = state of the branch-and-bound process (solver)
Actions = variables, nodes, primal heuristics, cuts, preprocessing routines to select
Episode = solving an instance to completion

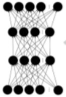
$$\tau \sim \underbrace{p_{init}(s_0)}_{\text{initial state}} \prod_{t=0}^{\infty} \underbrace{\pi(a_t|s_t)}_{\text{next action}} \underbrace{p_{trans}(s_{t+1}|a_t, s_t)}_{\text{next state}}$$

PO-MDP: state s ∈ $\mathcal{S}$ $\rightarrow$ observation o ∈ $\mathcal{O}$

OpenAI Gym API

```
import gym


env = gym.make("SpaceInvaders-v0")

for episode in range(1000):
    observation, done = env.reset(), False

    while not done:
        action = policy(observation)
        observation, reward, done, info = env.step(action)
```





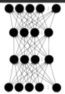
Ecole API

```python
import ecole

env = ecole.environment.Branching()

for episode in range(1000):
    obs, action_set, reward, done = env.reset("path/to/problem")

    while not done:
        action = policy(observation, action_set)
        obs, action_set, reward, done, info = env.step(action)
```

Solved multiple times

Solver statistics

Branching candidates



Number of nodes

Instance solved

Branching variable

Ecole features

Open: BSD-3 license

Easy: plug-and-play Python interface, one-line install via conda

Fast: full C++/PyBind11 implementation, thread-safe

Extensible: expand the library in C++ and/or Python via PySCIPOpt

Modular: compose from existing rewards, observations, and environments

```
env = ecole.environment.Branching(
    reward_function=(LpIterations()**2 - 3* NNodes()),
    observation_function=NodeBipartite(),
    scip_params={"presolving/maxrestarts": 2},
)
```

What's in Ecole now ?

Environments:

- Configuring: tune solver parameters (bandit)
- Branching: B&B variable selection

Rewards:

- Solving Time
- NNodes (B&B tree size)
- LP Iterations

Observations:

- Node Bipartite [Gasse et al., 2019]
- Khalil2016 [Khalil, Le Bodic, et al., 2016]
- Strong Branching Scores
- Pseudocosts

Instance Generators:

- Minimum Set Covering [Balas et al., 1980]
- Combinatorial Auction [Leyton-Brown et al., 2000]
- Capacitated Facility Location [Cornuejols et al., 1991]
- Maximum Independent Set [Bergman et al., 2016]

Go check `https://doc.ecole.ai` now !

Conclusions

Ecole exposes key control problems arising in **exact** CO solvers

- simple **Gym-like API** for learning
- modern open-source solver **SCIP**
- standard **benchmarks**, **metrics** and **feature sets** for reproducibility

What next

- new environments: learning to cut, local search
- new reward functions: primal/dual integral
- real-world instance collections
- ML4CO competition based on Ecole

Ecole: A Gym-like Library for Machine Learning in Combinatorial Optimization Solvers

Thank you!

Ecole contributors


A. Prouvost


M. Gasse


D. Chételat


J. Dumouchelle


L. Scavuzzo

A note on NP-Hardness

No Free Lunch Theorems for Optimization [Wolpert et al., 1997]:

> *[...] for any algorithm, any elevated performance over one class of problems is offset by performance over another class.*

General-purpose solvers ? Gurobi, IBM CPLEX, FICO Xpress, SCIP...

A note on NP-Hardness

No Free Lunch Theorems for Optimization [Wolpert et al., 1997]:

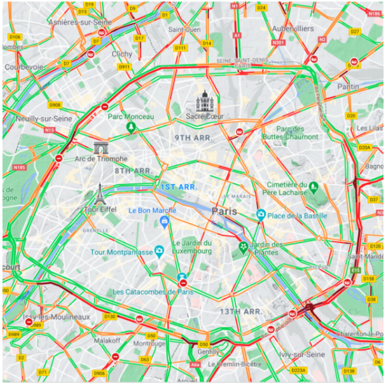
> *[...] for any algorithm, any elevated performance over one class of problems is offset by performance over another class.*

~~General-purpose~~ solvers ? Gurobi, IBM CPLEX, FICO Xpress, SCIP...

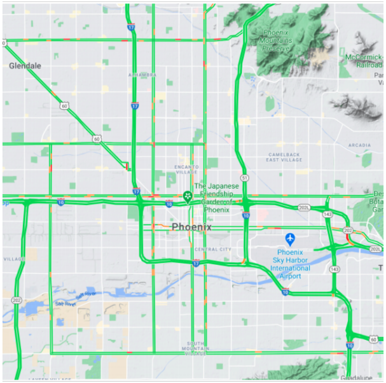
$\Longrightarrow$ Tailored to their own (internal) industrial benchmark.

A note on NP-Hardness

What about specific problem distributions ? What is the best solver for *me* ?



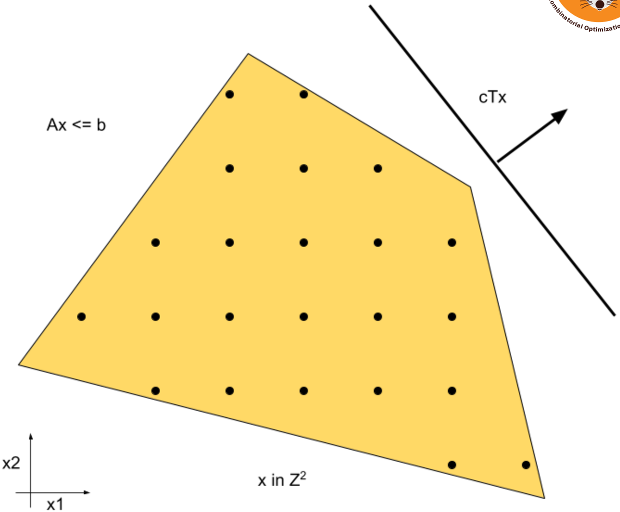
Paris



Phoenix, AZ

Mixed-Integer Linear Program (MILP)

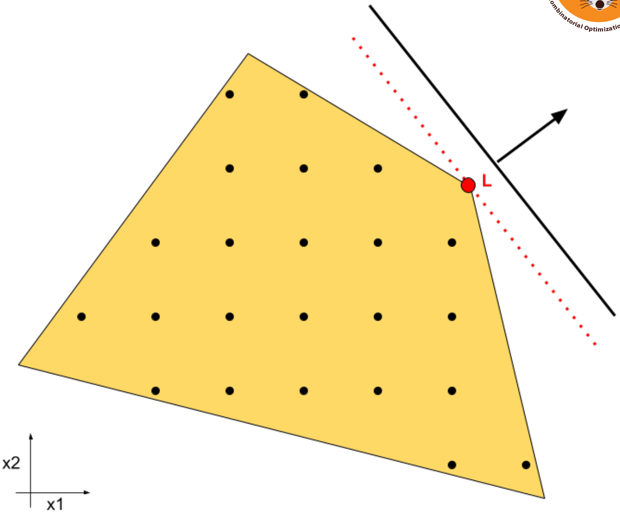
$$
\begin{aligned}
x^\star = \arg\min_{x} \quad & c^\top x \\
\text{subject to} \quad & Ax \leq b, \\
& x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}
\end{aligned}
$$



~~Mixed-Integer~~ Linear Program (LP)

$$
\begin{aligned}
\hat{x}^\star = \arg\min_{x} \quad & c^\top x \\
\text{subject to} \quad & Ax \le b, \\
& l \le x \le u, \\
& x \in \mathbb{R}^n.
\end{aligned}
$$

Efficient algorithms (e.g., simplex).

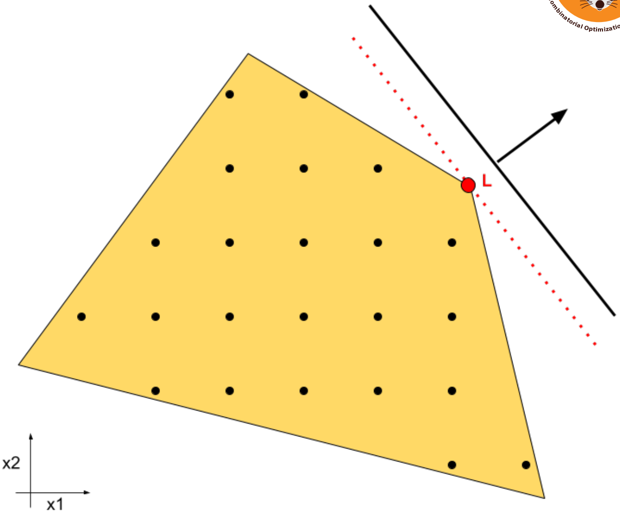


~~Mixed Integer~~ Linear Program (LP)

$$\begin{aligned}
\hat{x}^\star = \arg\min_x \quad & c^\top x \\
\text{subject to} \quad & Ax \leq b, \\
& l \leq x \leq u, \\
& x \in \mathbb{R}^n.
\end{aligned}$$

Efficient algorithms (e.g., simplex).

Lower bound to the original MILP



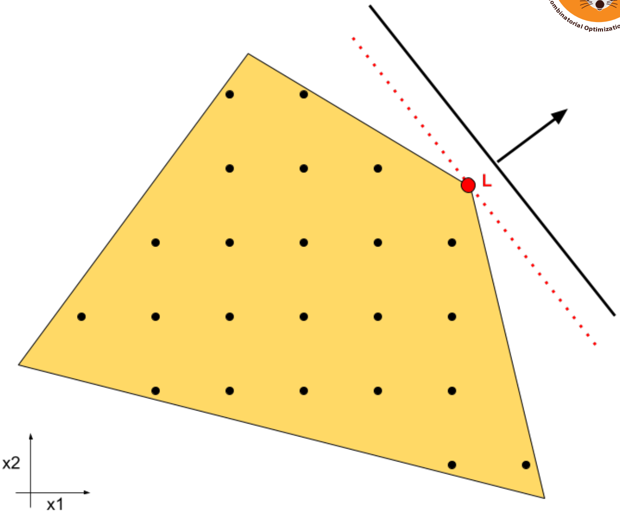
~~Mixed Integer~~ Linear Program (LP)

$$\begin{aligned}
\hat{x}^\star = \arg\min_{x} \quad & c^\top x \\
\text{subject to} \quad & Ax \le b, \\
& l \le x \le u, \\
& x \in \mathbb{R}^n.
\end{aligned}$$

Efficient algorithms (e.g., simplex).

Lower bound to the original MILP

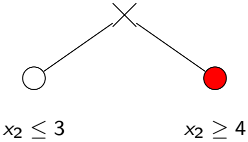
$\hat{x}^\star \in \mathbb{Z}^p \times \mathbb{R}^{n-p}$ (lucky)
$\rightarrow$ problem solved

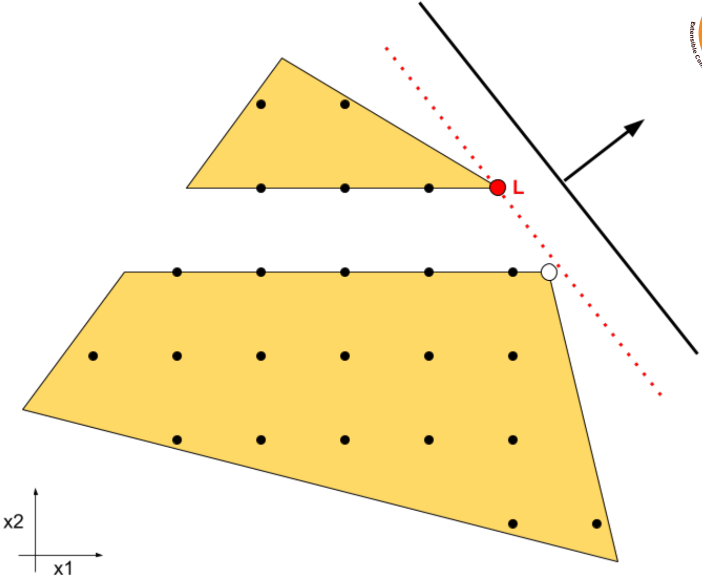


Branch-and-bound

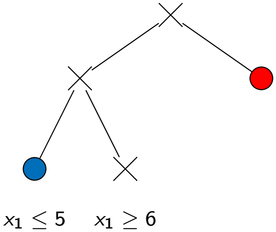
Recursively: pick a fractional variable and partition the LP

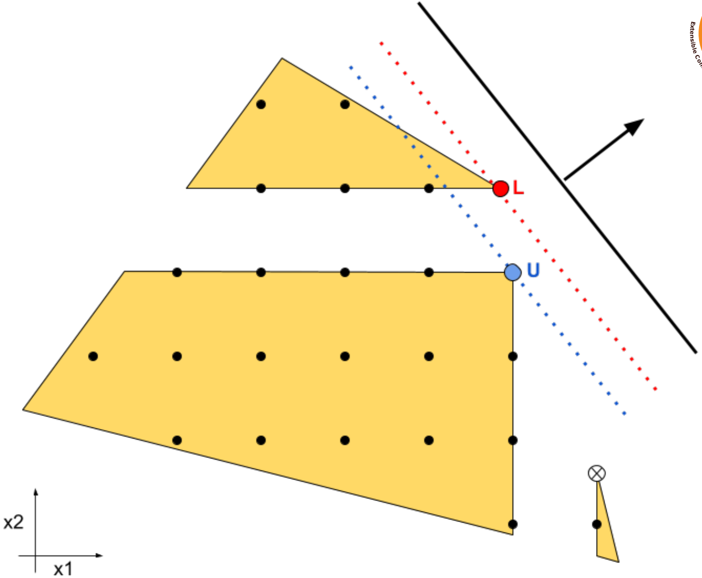
Example: $\hat{x}_i^\star = 3.62 \notin \mathbb{Z} \implies x_i \leq 3 = \lfloor \hat{x}_i^\star \rfloor \quad \vee \quad x_i \geq 4 = \lceil \hat{x}_i^\star \rceil$.





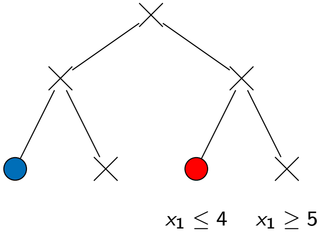
Lower bound (L): minimal among leaf nodes

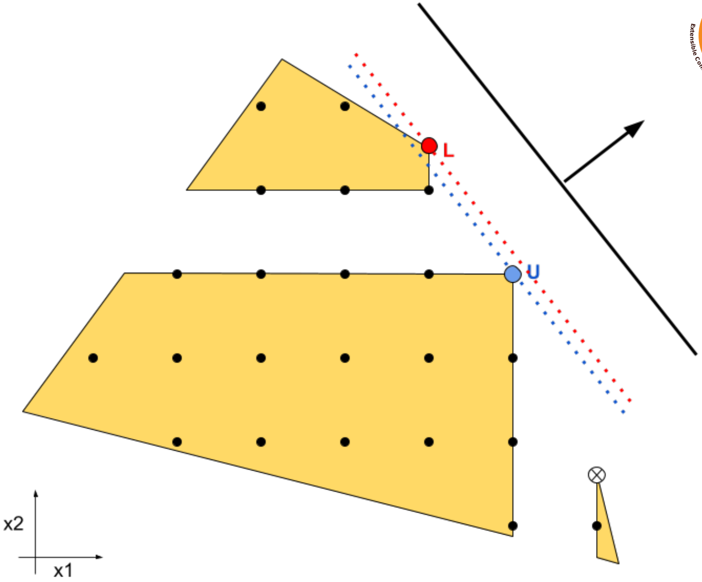




Lower bound (L): minimal among leaf nodes

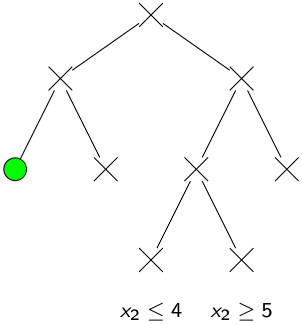
Upper bound (U): minimal among integral leaf nodes





Lower bound (L): minimal among leaf nodes

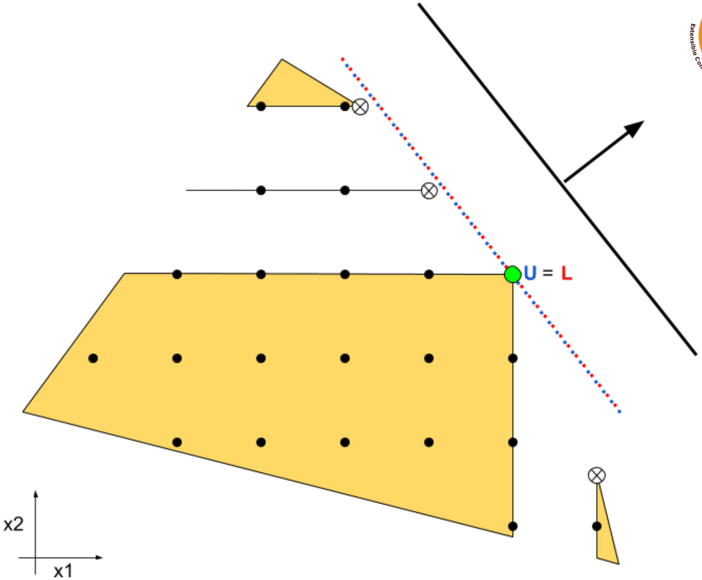
Upper bound (U): minimal among integral leaf nodes



Lower bound (L): minimal among leaf nodes

Upper bound (U): minimal among integral leaf nodes

Problem solved !



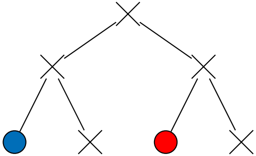
Branch-and-bound

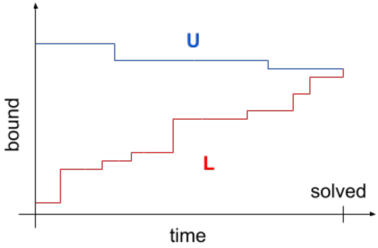
Sequential decisions:

- select an open leaf
- select a fractional variable
- select an open leaf
- select a fractional variable
- ...

Stopping criterion:

- L = U (optimality certificate)
- L = ∞ (infeasibility certificate)
- L - U < threshold (early stopping)

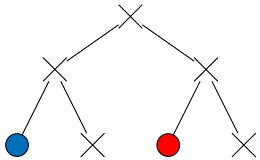


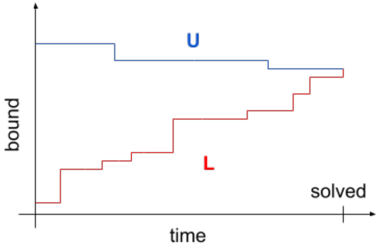


Branch-and-bound

Sequential decisions:

- select an open leaf
- select a fractional variable
- select an open leaf
- select a fractional variable
- ...





Stopping criterion:

- L = U (optimality certificate)
- L = ∞ (infeasibility certificate)
- L - U < threshold (early stopping)

To speed up things, other stuff also happens at each leaf (= sub-MILP).