Antoine Prouvost, Justin Dumouchelle, Maxime Gasse, Didier Chételat, Andrea Lodi

INFORMS — 24th of October 2021

DATA SCIENCE FOR REAL-TIME DECISION-MAKING

Mila

IVADO

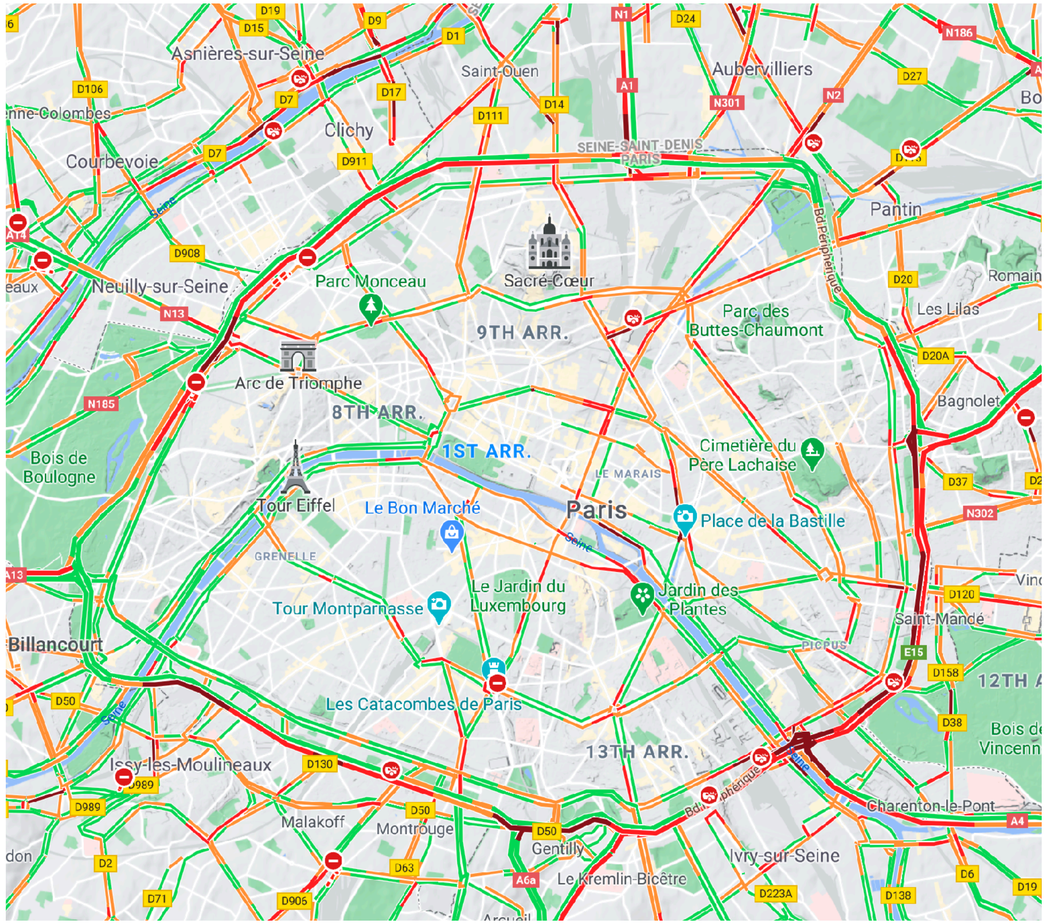
POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

How MILP get solved

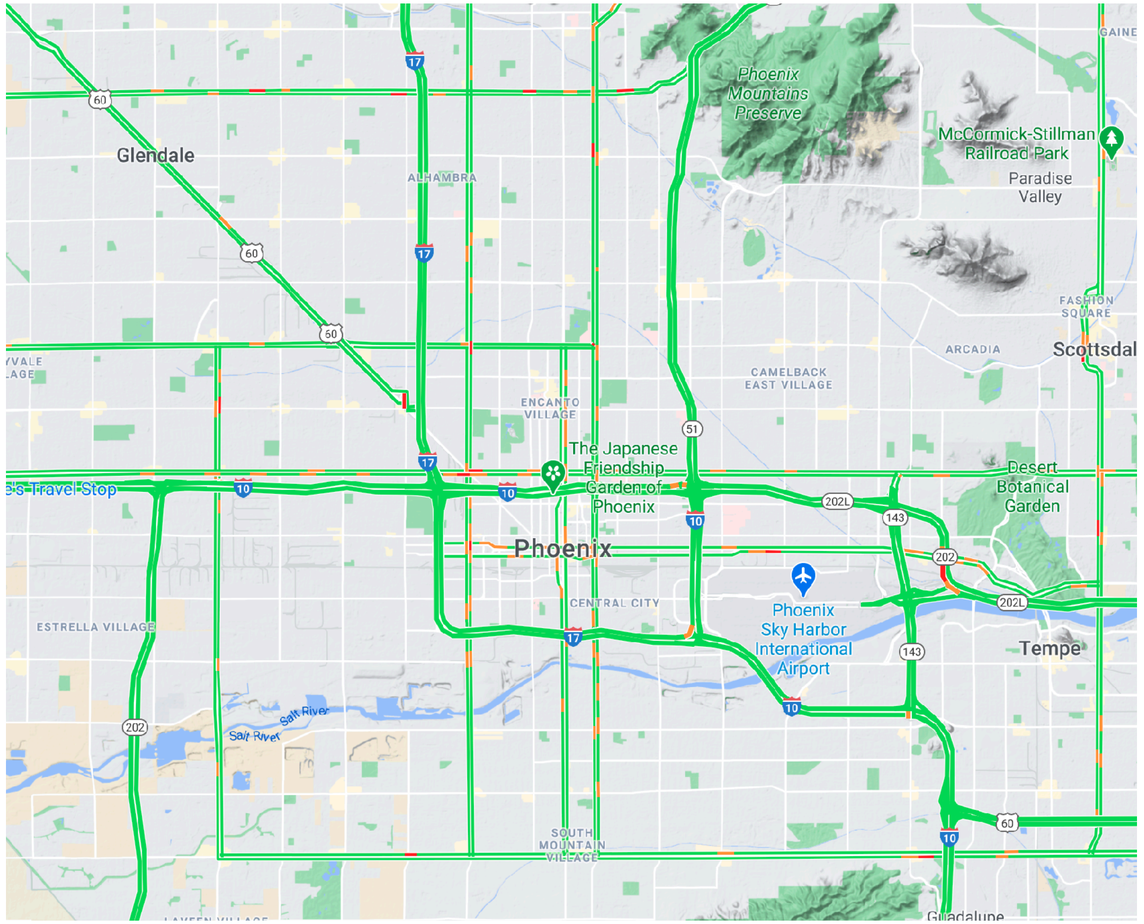
Handcrafted Heuristics

- Not focusing on all problems to beat $\mathcal{NP}$-hardness
- Weak form of **learning**
- Trained through researchers **trials and errors**
- On datasets such as MipLib

Problem Distribution



Paris



Phoenix

Learned Heuristics

- **Automatically** can be **specialized**
- Scale to more complex tasks (unlike rule based systems)
- Can span a large heuristic space

Bengio, Lodi, Prouvost. "*Machine learning for combinatorial optimization: a methodological tour d'horizon*." European Journal of Operational Research (2020).

Why Ecole?

Research Code

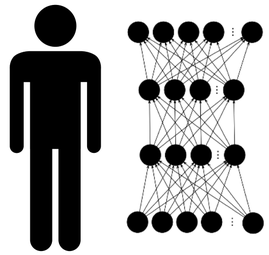
- Need to iterate fast
- Not tested
- Not optimized for speed
- Not modular
- Hard to install
- Often not reproducible

Introducing Ecole

- Intuitive API
- Python and C++
- Fast
- Well tested
- Easy to install
- Good defaults
- Extensible (through SCIP and PySCIPOpt)
- Ease comparison

Learning Environments

OpenAI Gym



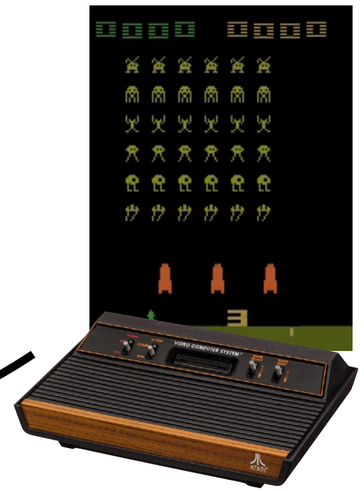
Agent
$\pi(a|s)$

State
S_{t+1}

Reward
R_{t+1}

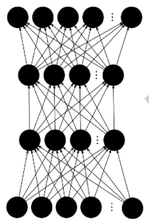
Action
A_t

Environment
$p(s', r|a, s)$

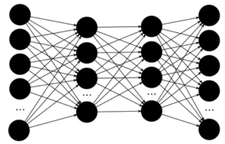


OpenAI Gym

```
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)
```



Branching with Ecole



Agent
$\pi(a|s)$

Solver statistics

State
S_{t+1}

Reward
R_{t+1}

Action
A_t

Branching variable

Number of nodes

Environment
$p(s', r|a, s)$

Solver

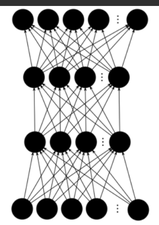
Branching with Ecole

```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



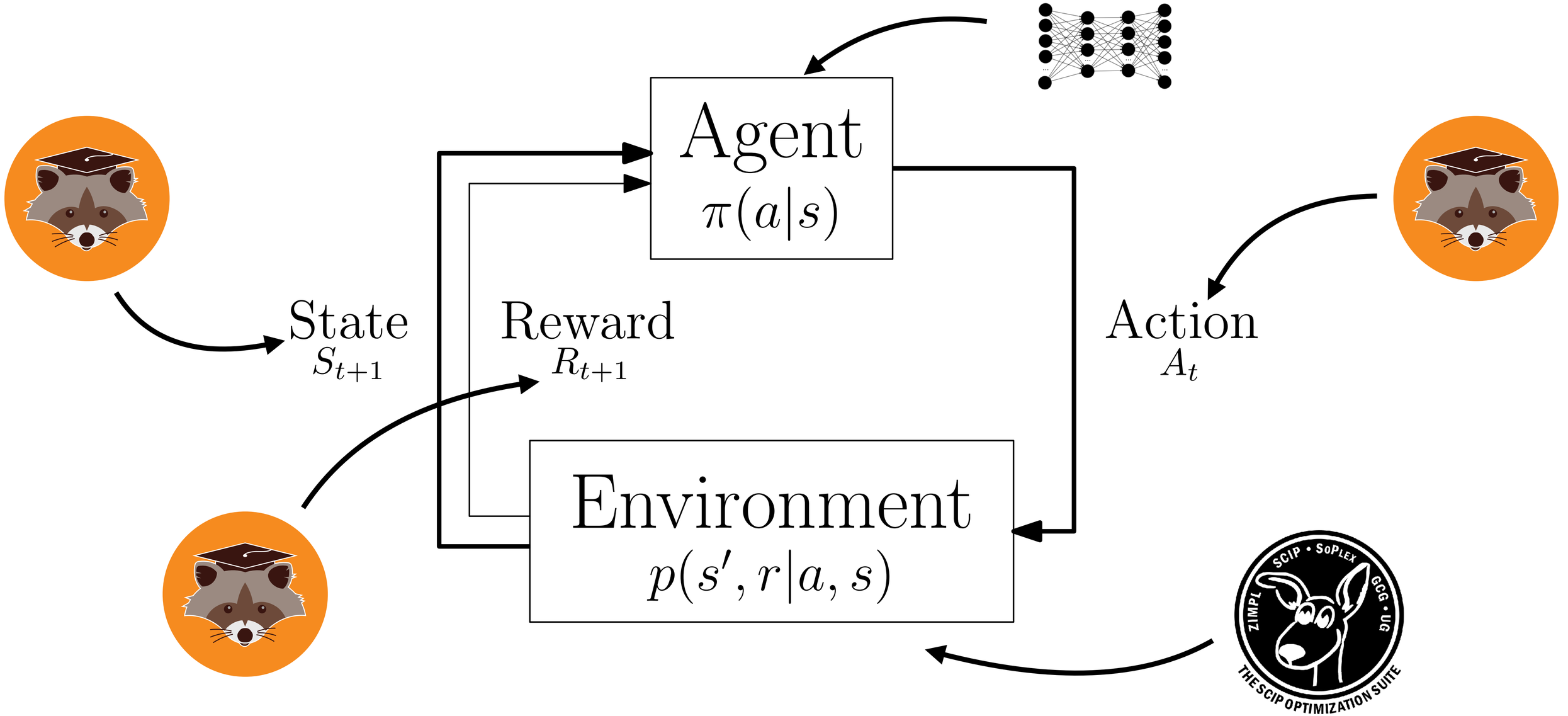
Number of nodes

Instance solved

Branching variable

Extensibility

Highly configurable



Easily swap and define

- Rewards
  - LP Iterations
  - Number of Nodes
  - Solving Time
  - Primal / Dual Integral
- Observation (States)
  - Milp Bipartite
  - Gasse et al. “Exact combinatorial optimization with graph convolutional neural networks”, 2019
  - Hutter et al. “Sequential model-based optimization for general algorithm configuration.”, 2011
  - Khalil et al. “Learning to branch in mixed integer programming.”, 2016
- Any additional information
- Environments
  - Branching variable selection
  - Parameter selection
  - Primal search

Composition

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

New Reward Function

```
class NNodeInitLPIterations:

    def before_reset(self, model):
        self.last_iterations = 0.0

    def extract(self, model, done):
        new_iterations = model.as_pyscipopt().getNNodeInitLPIterations()
        diff_iterations = new_iterations - self.last_iterations
        self.last_iterations = new_iterations
        return diff_iterations
```

New Environment Dynamics

```
class SimpleBranchingDynamics(BranchingDynamics):

    def reset_dynamics(self, model):
        # Share memory with Ecole model
        pyscipopt_model = model.as_pyscipopt()

        pyscipopt_model.setPresolve(PY_SCIP_PARAMSETTING.OFF)
        pyscipopt_model.setSeparating(PY_SCIP_PARAMSETTING.OFF)

        # Let the parent class do the rest
        return super().reset_dynamics(model)

    def step_dynamics(self, model):
        ...
```

Instance Generators

```
from ecole.instance import SetCoverGenerator

generator = SetCoverGenerator(n_rows=100, n_cols=200)

for i in range(50):
    instance = next(generator)

    instance.write_problem("some-folder/set-cover-{i:04}.lp")
```

- Set Cover
- Capacitated Facility Location
- Independent Set
- Combinatorial Auction

Ecole In Math

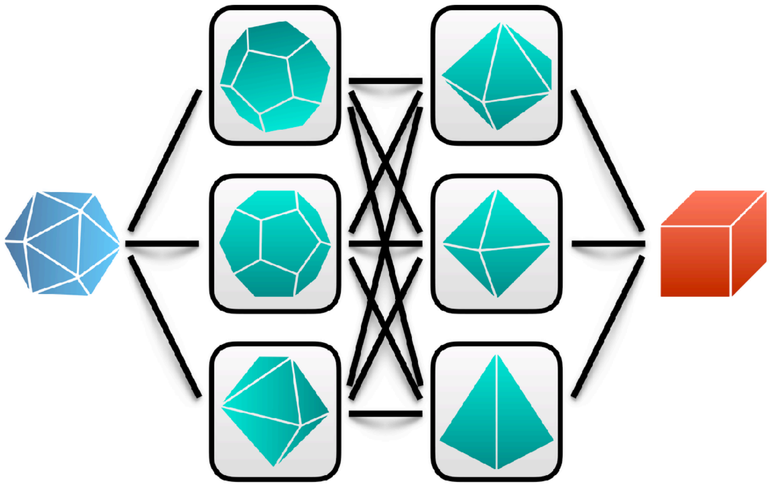
- Instance space $\mathscr{I}$
- State space $\mathscr{S}$ and action space $\mathscr{A}$
- Instance distribution $p_{inst} : \mathscr{I} \to \mathbb{R}_{\geq 0}$
- Conditional initial state distribution $p_{init} : \mathscr{S} \times \mathscr{I} \to \mathbb{R}_{\geq 0}$
- State transition distribution $p_{trans} : \mathscr{S} \times \mathscr{A} \times \mathscr{S} \to \mathbb{R}_{\geq 0}$
- Reward function $R : \mathscr{S} \to \mathbb{R}$
- Observation space $\mathscr{O}$ and function $O : \mathscr{S} \to \mathscr{O}$ (POMDP)
- User policy π

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

NEURAL INFORMATION PROCESSING SYSTEMS

Machine Learning for Combinatorial Optimization

——COMPETITION 2021——



Simon Bowly
Chris Cameron
Quentin Cappart
Jonas Charfreitag
Laurent Charlin
Didier Chételat
Justin Dumouchelle
Maxime Gasse
Ambros Gleixner
Aleksandr M. Kazachkov
Elias B. Khalil
Pawel Lichocki
Andrea Lodi
Miles Lubin
Chris J. Maddison
Christopher Morris
Dimitri J. Papageorgiou
Augustin Parjadis
Sebastian Pokutta
Antoine Prouvost
Lara Scavuzzo
Giulia Zarpellon

https://www.ecole.ai/2021/ml4co-competition/

ds4dm

ds4dm/ecole

www.ecole.ai

doc.ecole.ai