ECOLE

Extensible Combinatorial Optimization Learning Environments

What is Ecole?

Ecole exposes several key decision tasks arising in general-purpose combinatorial optimization solvers as control problems over Markov decision processes. Its interface mimics the popular OpenAI Gym library and is both extensible and intuitive to use.

Why should I use it?

- Fast ⚡
- Extensible 🧩
- Well tested 🔍
- Good defaults 🌈
- Python (no GIL) & C++ 💻
- Available on Conda 🚀
- Well documented 📖

Who develops Ecole?

Antoine Prouvost
Justin Dumouchelle
Lara Scavuzzo
Maxime Gasse
Didier Chételat
Andrea Lodi

ds4dm
ds4dm/ecole
www.ecole.ai
doc.ecole.ai

NeurIPS LMCA workshop 12th of December 2020

DATA SCIENCE FOR REAL-TIME DECISION-MAKING · Mila · IVADO · POLYTECHNIQUE MONTREAL UNIVERSITÉ D'INGÉNIERIE

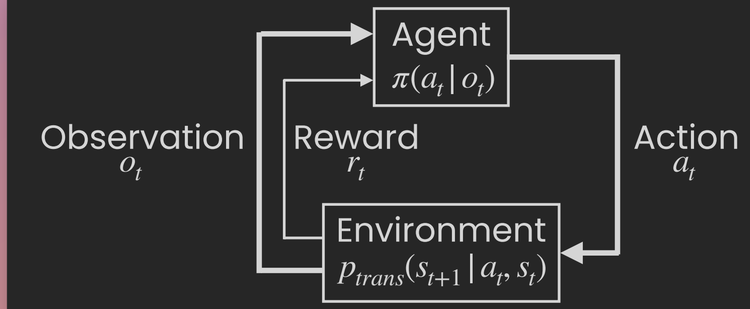
How do I use it?

```python
import ecole

env = ecole.environment.Branching(
    observation_function=ecole.observation.NodeBipartite(),
    reward_function=-1.5 * ecole.reward.NNodes() ** 2,
)

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

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

I want my own environment

```python
from ecole.environment import BranchingDynamics
from pyscipopt.scip import PY_SCIP_PARAMSETTING


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)
```

Sounds like a MDP!



I want to extract my own data

```python
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
```

Just give me the Math

- State space $\mathcal{S}$ and action space $\mathcal{A}$
- Initial state distribution $p_{init} : \mathcal{S} \to \mathbb{R}_{\geq 0}$
- State transition distribution $p_{trans} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_{\geq 0}$
- Reward function $R : \mathcal{S} \to \mathbb{R}$
- Observation state $\mathcal{O}$ and function. $O : \mathcal{S} \to \mathcal{O}$ (POMDP)

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

Where do I get problem instances?

```python
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")
```