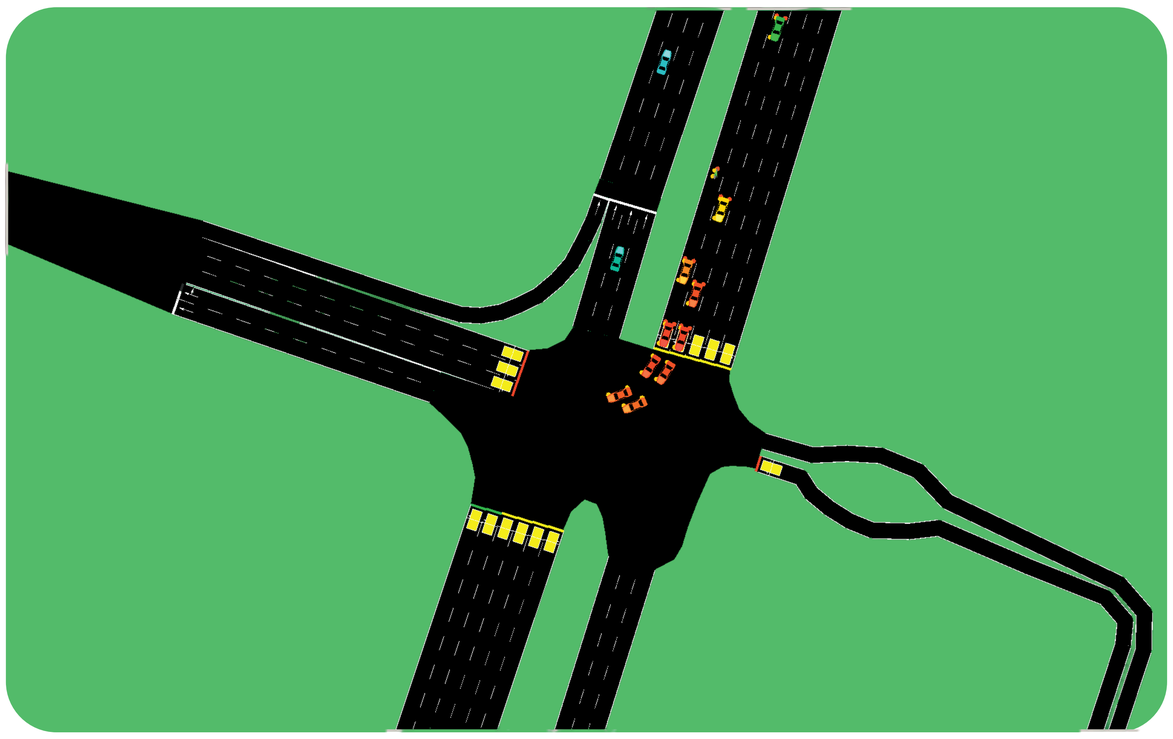
SUMOGYM: A FRAMEWORK FOR PERFORMING TRAFFIC CONTROL USING REINFORCMENT LEARNING

CONTRIBUTERS

Cobus Louw, Louwrens Labuschagne and Tiffany Woodley

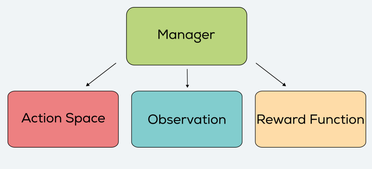


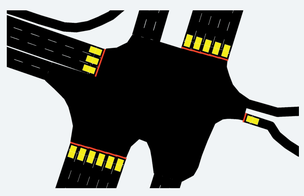
SumoGym is a third party Open AI Gym wrapper for SUMO simulator. SumoGym simplifies the training of RL agents in this traffic simulation environment. SumoGym inherits from the Env class from the OpenAI gym package. It therefore behaves like any other Gym environment. This enables easy application of existing RL algorithms on problems in this environment. Typical usage of the SumoGym environment can be seen below:

```python
import gym
import sumogym

env = gym.make('SumoGym-Technopark-v0', render=True)
done = False
obs = env.reset()

while not done:
    action = env.action_space.sample()
    obs, reward, done, _ =  env.step(action)
env.close()
```

WRAPPERS

SumoGym follows the Open AI Gym philosophy. Traditional *Gym wrappers* can be used to transform observations and rewards before they are returned to the agent, as well as to transform actions before they are performed in the environment. Additionally, *SumoGym wrappers* are provided. SumoGym includes a *Manager* that allows multiple wrappers to be applied to the environment. These wrappers allow you to completely alter the observations and reward functions of the environment. These wrappers are primarily built on TraCI, a Python package that communicates directly with the SUMO simulation environment. TraCI provides access to a wide range of metrics to the observation and reward functions. For example, if an environment's default observation is lane queue length, the relevant wrapper can be used to exchange that with induction loop readings. It is also possible to obtain a combined observation of multiple metrics. Custom wrappers can also be easily created.





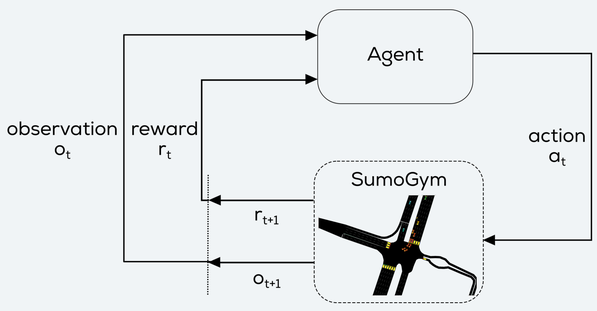
Induction loop layout

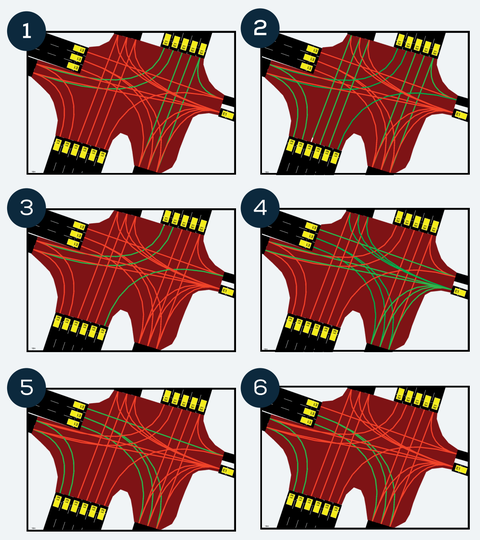
Induction loop readings

TRAFFIC LIGHTS

SumoGym supports multiple traffic light(s) programs. These can be used as a benchmark or to test alongside ML-based traffic light agents. All of these traffic lights are easily configurable using YAML files. Traffic lights included in SumoGym:

- Gym -based
- Fixed-time
- Delay-based vehicle actuated (VA)
- Gap-based VA

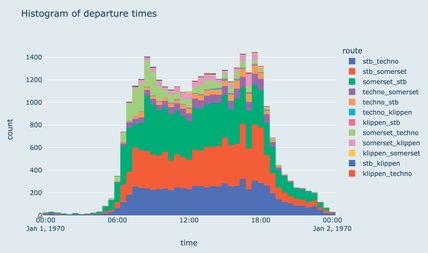




Traffic light stages

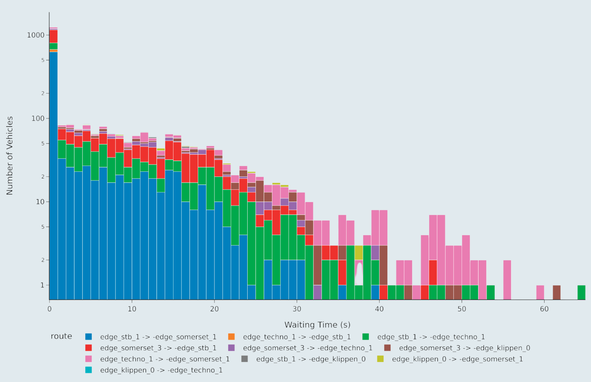
TRAFFIC

Training is often divided into discrete episodes in Reinforcement Learning (RL). In our application, it is desirable to train robust agents that can deal with a wide range of traffic patterns. As a result, we decided to include a traffic generator that samples traffic from a variety of traffic distributions in each training episode. This allows the user to specify the distributions for the various routes and traffic patterns. During training, the traffic generator generates probabilistic traffic distributions to encourage the agents to explore. SumoGym also has the option of generating deterministic traffic. This allows the user to specify the departure time, route, vehicle type, etc for each vehicle in order to generate identical traffic volumes. The performance of various models can be compared by testing on identical traffic patterns.



OUTPUT

After a simulation has been completed, SUMO provides a vast array of output and results. We have added support for SUMO's outputs, which can be configured via a YAML file. The SUMO outputs are then converted to *CSV*, with *feather* as an alternative. Additionally, some plots are generated and saved in your working directory to provide a quick overview of your experiment. SumoGym also supports Weights and Biases logging, allowing you to monitor agents in real time.



Weights & Biases

PETTINGZOO

A PettingZoo wrapper is provided by SumoGym. This wrapper can be used to convert the environment to a PettingZoo environment. This allows Multi-agent Reinforcement Learning (MARL) algorithms to be implemented to optimise networks with multiple traffic lights.

```python
from sumogym import SumoGym, SumoGymPZ

env = gym.make('SumoGym-DualIntersection-v0', render=True)
env = SumoGymPZ(env)
done = False
obs= env.reset()

for agent in env.agent_iter():
  observation, reward, done, info = env.last()
  action = policy(observation, agent)
  env.step(action)
```

CONCLUSION

SumoGym provides a general framework for traffic signal control. The modularity of this environment is a major advantage. Having the ability to use different wrappers, makes the environment highly customizable. SumoGym is still in its early stages of development, but we see a wide range of applications for it and believe it can become the standard environment for implementing RL on traffic signal control.

ByteFuse