Optimizing Traffic Flow Using sumo-RL: An Integration of SUMO and Reinforcement Learning

SECTION 1: OVERVIEW

SUMO-RL is a Python library that facilitates RL experiments in traffic scenarios. This library integrates SUMO, which is used for simulating traffic scenarios with advanced mechanisms like machine learning and reinforcement learning concepts. The SUMO-RL library supports various RL algorithms, allowing customization and experimentation with different settings to optimize traffic conditions.

The Sumo RL library supports both single-agent and multi-agent setups and comes with predefined observation spaces and action spaces for traffic signal control. All of the predesigned environments are customizable according to the needs of the user and have extensive documentation and tutorials for ease of use.

Dependencies:

SUMO Simulator (from Eclipse SUMO)

Python (3.6 or up)

Python libraries:

1. **sumo-RL**: Provides the RL environment interfacing SUMO with OpenAI Gym.
2. **stable-baselines3**: A set of improved implementations of RL algorithms.

SECTION 2: INTRODUCTION

To simulate the entire scenario, the model integrates four main components: the Sumo environment, the PettingZoo Sumo environment, traffic signal functions, and observation functions.

The environment is modeled using a four-by-four grid network in SUMO. Each intersection is equipped with traffic signals controlled by an RL agent. The state space includes the number of incoming vehicles and waiting times, while the action space consists of possible traffic signal phases. A reinforcement learning algorithm is employed to train the agents, aiming to minimize cumulative vehicle waiting times across the network.

In the simulation model, the agent refers to the traffic signal controller that operates within the SUMO environment and that makes decisions about traffic signal timings to optimize traffic flow based on observed traffic conditions.

SUMO Environment:

The Sumo environment is designed for single-agent RL experiments. It simulates individual traffic intersections where vehicles move through a defined road network and an agent learns to control traffic lights to optimize flow. It also models the physical aspects of traffic flow, including road layouts, vehicle interactions, and traffic dynamics.

Within this environment, only one agent interacts with the simulated traffic, making decisions that affect how vehicles move through intersections.

PettingZoo Environment:

PettingZoo is a Python library designed for multi-agent reinforcement learning (MARL) environments. It provides a standardized interface for agents to interact with an environment that has multiple learning agents.

Multi-agent RL experiments are supported within this environment. Multiple agents control traffic lights at different intersections, collaboratively or competitively optimizing traffic flow.

Traffic Signal Control:

Traffic signal control is the primary application of SUMO-RL.

These functions define how the agent can interact with the traffic signals within the SUMO environment.

1. **Traffic Signal API:** Allows RL agents to control the state and timing of traffic signals. Supports functionalities like phase switching, yellow light durations, and adaptive timing.
2. **Gymnasium API:** If the network only has ONE traffic light, then it can be instantiated with a standard Gymnasium env.
3. **PettingZoo API:** For multi-agent environments, the PettingZoo API is used.

SECTION 3: METHODOLOGY

Role of the Agent in the Simulation:

Decision-Making Entity: The agent is responsible for deciding when and how to change traffic signal phases to optimize traffic flow.

Interaction with the Environment:

1. **Observations**: The agent observes the current traffic conditions through the observation functions.
2. **Actions**: It takes actions by calling traffic signal functions to change the state of traffic lights.
3. **Rewards**: The agent receives rewards based on a defined reward function, such as minimizing cumulative vehicle waiting times or queue lengths.

Learning Process:

Through reinforcement learning algorithms, the agent learns optimal policies over time by interacting with the environment and receiving feedback.

The goal is to improve its decision-making strategy to achieve better traffic optimization.

Single-Agent vs. Multi-Agent Scenarios:

Single-Agent Scenario:

There is one agent controlling traffic signals in the environment, often at a single intersection or a centralized controller for multiple intersections.

The agent's objective is to optimize traffic flow based on global observations.

Multi-Agent Scenario:

Multiple agents are present, each controlling different traffic signals at various intersections.

Agents may operate independently or collaborate to optimize overall traffic flow.

The PettingZoo SUMO environment facilitates this multi-agent interaction.

Observations and Rewards

The effectiveness of RL agents depends heavily on the design of observations and rewards.

1. **Observations**

SUMO-RL offers a variety of observation spaces, such as vehicle counts, queue lengths, and average speeds.

Users can select or customize observations based on the specific requirements of their traffic optimization problem.

2. **Reward Functions**

Reward functions are customizable and can include metrics like reduced waiting time, minimized queue length, or maximized throughput.

Proper reward design is crucial for training effective RL agents.

SECTION 4: OPTIMIZATION AND LEARNING

Learning:

The action space is defined as a vector space of discrete values in delta_time secs which represents the time each traffic agent waits for before switching the signal. Every 'delta_time' seconds, each traffic signal agent can choose the next green phase configuration. Every time a phase change occurs, the next phase is preceded by a yellow phase lasting yellow_time seconds.

The reward is based on minimizing the wait times. The cumulative wait time is calculated for each agent as **the number of vehicles * the sum of individual wait times recorded by each vehicle.**

The default reward function is the change in cumulative vehicle delay

$$r_t = D_{a_t} - D_{a_{t+1}}$$

The formulae for wait time in sumo

The reward is how much the total delay (sum of the waiting times of all approaching vehicles) changed in relation to the previous time step.

Since the action space and the reward function are made of discrete values, the suitable choices of reinforcement learning algorithms are:

i) Deep Q-Network (DQN)

ii) Proximal policy optimization (PPO)

iii) Soft Actor-Critic (SAC) for discrete actions

Optimization:

1. Real-Time Coordination of Traffic Lights

By networking multiple traffic signals, the RL agents can coordinate signal timings across intersections. This ensures smoother traffic flow and reduces traffic in a network. Agents adjust signal phases according to fluctuations in traffic density and patterns.

2. Rush Hour Traffic Management

During peak hours, the RL system can implement strategies such as longer green phases for dominant traffic flows or adjusting cycle lengths to accommodate increased vehicle volumes. The system can learn how to redirect traffic to less busy routes.

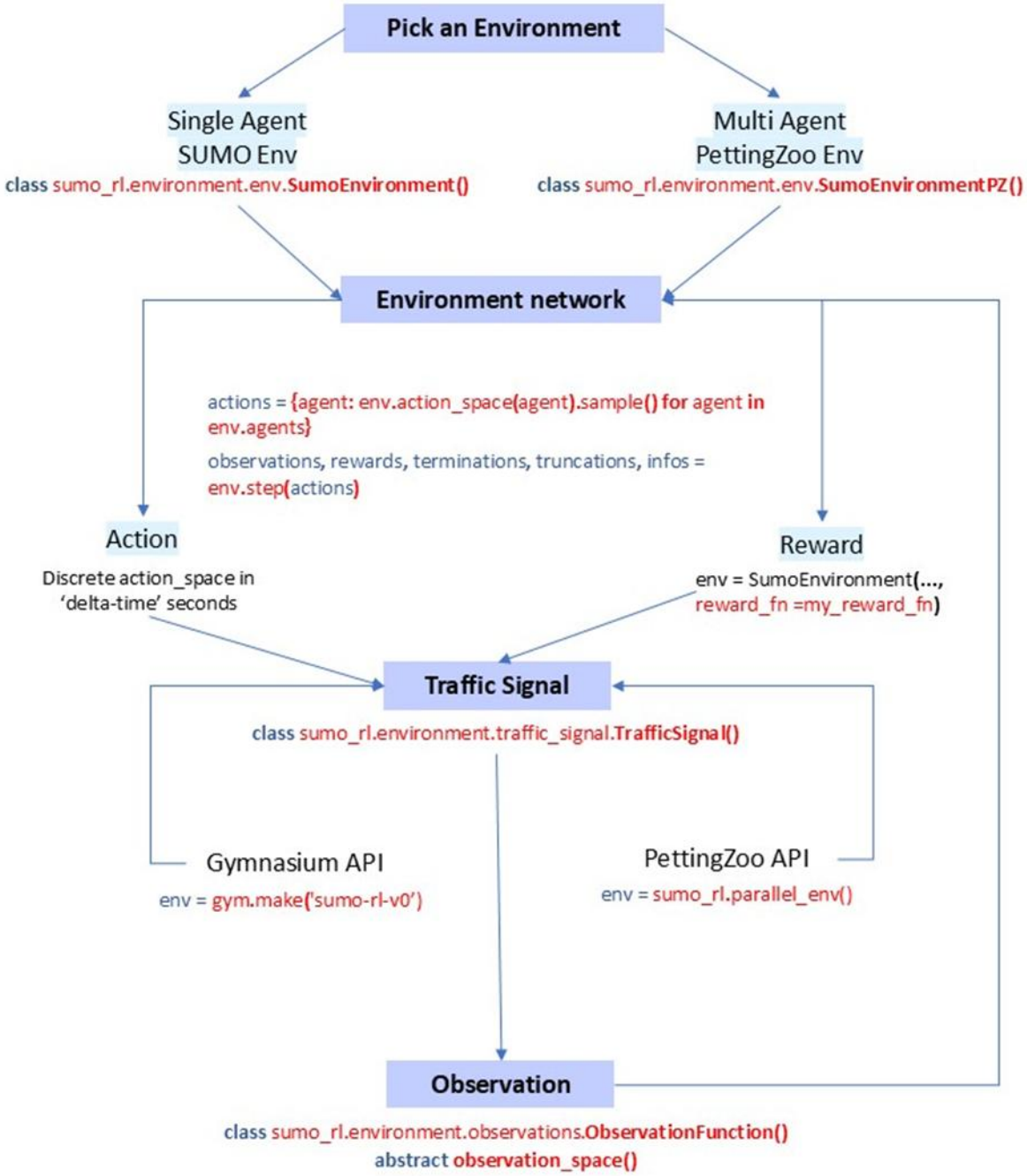
3. Emission Monitoring and Environmental Impact

Traffic signals can monitor estimated emissions by considering factors like idling time and stop-and-go movements. When abnormally high levels of emissions are detected, the system can trigger warnings. The RL agents can incorporate emission levels into their reward functions, promoting actions that reduce overall emissions.

4. Route Optimization

Using simulations, the system can identify optimal routes that reduce wait times for vehicles and travel times for commuters.

SECTION 5: BLOCK DIAGRAM



Author: Shazia Parween.

References:

Lopez, P. A., et al. (2018). *Microscopic Traffic Simulation using SUMO*. Proceedings of the 21st IEEE International Conference on Intelligent Transportation Systems (ITSC). https://sumo.dlr.de/docs/

Gawron, C. (1998). *An iterative algorithm to determine the dynamic user equilibrium in a traffic simulation model*. International Journal of Modern Physics C, 9(03), 393–407. DOI: 10.1142/S0129183198000303

Krause, S., et al. (2021). *sumo-rl: Reinforcement Learning environments for traffic signal control using SUMO*. GitHub repository. https://github.com/LucasAlegre/sumo-rl

Stable-Baselines3 Contributors (2020). *Stable Baselines3: Reliable implementations of reinforcement learning algorithms in PyTorch*. https://github.com/DLR-RM/stable-baselines3

Terry, J. K., et al. (2021). *PettingZoo: Gym for Multi-Agent Reinforcement Learning*. https://www.pettingzoo.ml

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press. http://incompleteideas.net/book/the-book-2nd.html