CSE 546 - Reinforcement Learning

Deep Reinforcement Learning for Intelligent Traffic Signal Control:

An Adaptive and Scalable Approach

REPORT

Team 21: Final Project Checkpoint 1

Environment Setup

SUMO (Simulation of Urban MObility), an open source, highly portable, microscopic and continuous multi-modal traffic simulation package designed to handle large networks. The suite includes a variety of tools that assist in tasks like network import, route calculations, traffic scenario visualization, and emission calculations. The XML configuration(twoone.net.xml) defines the road network, including the layout, traffic light logic, and connections between lanes and edges.

Sumo Components used to build Traffic Lights Simulator

- **Network elements (Netedit)**: The XML describes the traffic network including edges, lanes, junctions, and connections, which are foundational elements within SUMO.
- **sumo-gui**: The graphical user interface of SUMO is used, allowing interactive simulation and visualization of the traffic network.
- **Traffic Light Logic (tlLogic)**: Defined in the XML to control traffic lights at junctions.
- **Simulation Control**: The use of "TraCI" in our script indicates remote control over the simulation, adjusting traffic light phases in response to the environment's state.

Network Configuration

- **Edges**: Represents roads in the network, each with unique ids.
- **Lanes**: Subsections of edges with attributes such as speed limit and length.
- **Junctions**: Intersections where lanes meet, some of which are controlled by traffic lights (id="0") and others that operate on priority rules (id="1", id="2", etc.).
- **Traffic Lights**: Controlled by tlLogic elements with defined phases and durations.
- **Connections**: Define the permissible movements between lanes at junctions, including right-of-way rules.

Traffic Simulation

- **Vehicle Movement**: Vehicles are simulated within the network, with their behaviors governed by road rules and traffic lights.
- **Traffic Data Collection**: Real-time data such as the number of vehicles departed, arrived, and average speeds are collected for analysis.
- **Traffic Light Control**: The traffic lights' phases are managed, possibly by an AI agent, to optimize traffic flow.

Functionality of DQN (Deep Q-Network)

DQN is a reinforcement learning algorithm used to solve decision-making problems. It uses a neural network to approximate the optimal action-value function, which tells us what the expected return (or total accumulated reward) is, given a state and an action.

Components of DQN

- **State Space**: All possible situations that the agent can encounter. In traffic simulation, this could include the number of vehicles waiting at a light, waiting times, etc.
- **Action Space**: All the possible actions the agent can take. For traffic lights, this would be the available phases.
- **Reward Signal**: A numerical value given to the agent after it performs an action. It's designed to guide the agent—penalizing bad traffic patterns and rewarding smooth flow.

DQN Working Mechanism

- **Observation**: The agent observes the current state of the environment (in this case, the traffic conditions).
- **Decision**: The agent decides on an action based on the observed state. It can either explore a new action with a probability of epsilon (random choice) or exploit the best-known action by choosing the one with the highest expected reward.
- **Action Execution**: The chosen action is applied to the traffic lights, and the simulation advances one step.
- **Reward Assessment**: After performing the action, the agent receives a reward based on the new state of the environment (e.g., the reduction in waiting time).

- **Learning**: The agent uses the reward to update its neural network, improving its action-value function approximation.

Experience Replay

The DQN agent stores its experiences in a replay memory, which contains a history of state, action, reward, and next state transitions. The agent randomly samples from this memory to learn, making the learning process more stable and efficient.

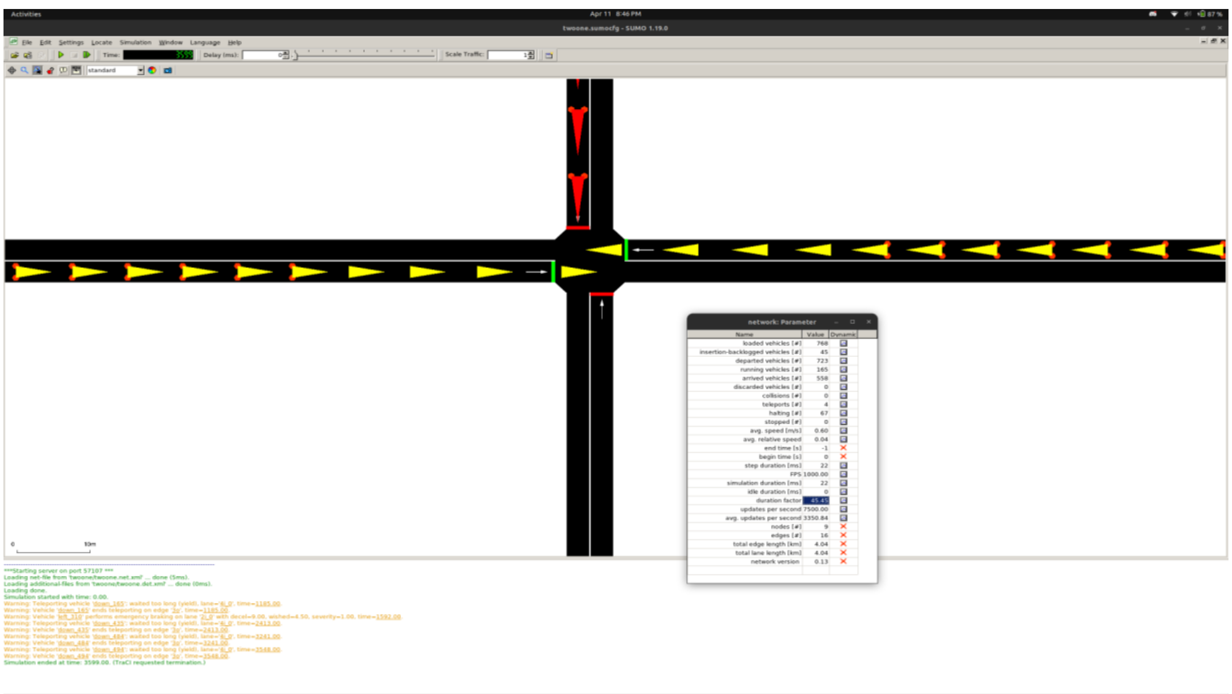
Epsilon-Greedy Strategy

The agent employs an epsilon-greedy strategy where it initially explores the action space randomly but gradually exploits more as it learns about the environment. The epsilon value decreases over time, reducing the probability of random actions and increasing the reliance on learned behaviors.

Model Updates

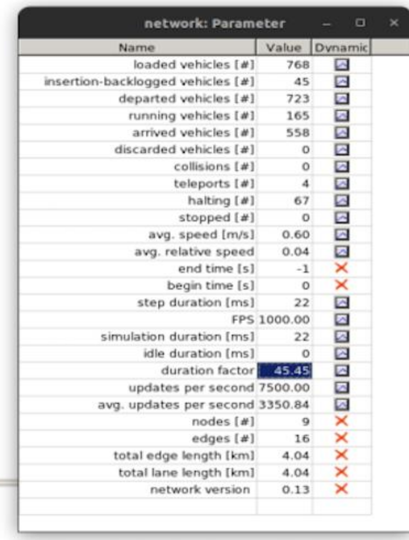
The agent updates its neural network model by predicting the expected rewards and adjusting the model weights using backpropagation to minimize the difference between predicted and actual rewards, thereby improving its policy over time.

Traffic Lights Simulator after training with DQN



Traffic Lights Simulator using DQN - Network Parameters Observation

- The network parameters from the SUMO traffic light simulator indicate that 768 vehicles were loaded into the simulation, with 45 experiencing backlogged insertion, meaning they could not enter the network immediately.
- A total of 723 vehicles departed, and 558 arrived at their destinations, while 165 are still running, or in-transit. There have been no recorded vehicle discards or collisions. Four teleportations occurred, which typically means that some vehicles were instantly moved due to issues like being stuck in traffic.
- Average speeds are low at 0.60 m/s, suggesting heavy traffic or congested conditions.
- The network itself consists of 16 edges (individual street segments) totaling 4.04 km in length. The simulation step duration is set at 22 ms with an average of 3350.84 updates per second, indicating a highly detailed simulation timeframe. The 'duration factor' is notably high at 45.45, which may be used for scaling the simulation's time or speed.


network: Parameter

Name	Value	Dynamic
loaded vehicles [#]	768	
insertion-backlogged vehicles [#]	45	
departed vehicles [#]	723	
running vehicles [#]	165	
arrived vehicles [#]	558	
discarded vehicles [#]	0	
collisions [#]	0	
teleports [#]	4	
halting [#]	67	
stopped [#]	0	
avg. speed [m/s]	0.60	
avg. relative speed	0.04	
end time [s]	-1	✕
begin time [s]	0	✕
step duration [ms]	22	
FPS	1000.00	
simulation duration [ms]	22	
idle duration [ms]	0	
duration factor	45.45	
updates per second	7500.00	
avg. updates per second	3350.84	
nodes [#]	9	✕
edges [#]	16	✕
total edge length [km]	4.04	✕
total lane length [km]	4.04	✕
network version	0.13	✕

Traffic Lights Simulator Running Console – DQN

```
***Starting server on port 57107 ***
Loading net-file from 'twoone/twoone.net.xml' ... done (5ms).
Loading additional-files from 'twoone/twoone.det.xml' ... done (0ms).
Loading done.
Simulation started with time: 0.00.
Warning: Teleporting vehicle 'down_165'; waited too long (yield), lane='4i_0', time=1185.00.
Warning: Vehicle 'down_165' ends teleporting on edge '3o', time=1185.00.
Warning: Vehicle 'left_310' performs emergency braking on lane '2i_0' with decel=9.00, wished=4.50, severity=1.00, time=1592.00.
Warning: Teleporting vehicle 'down_435'; waited too long (yield), lane='4i_0', time=2413.00.
Warning: Vehicle 'down_435' ends teleporting on edge '3o', time=2413.00.
Warning: Teleporting vehicle 'down_484'; waited too long (yield), lane='4i_0', time=3241.00.
Warning: Vehicle 'down_484' ends teleporting on edge '3o', time=3241.00.
Warning: Teleporting vehicle 'down_494'; waited too long (yield), lane='4i_0', time=3548.00.
Warning: Vehicle 'down_494' ends teleporting on edge '3o', time=3548.00.
Simulation ended at time: 3599.00. (TraCI requested termination.)

'twoone/twoone.sumocfg' loaded.
```

Conclusion

In this SUMO simulation environment, the DQN agent is tasked with optimizing traffic light controls to improve traffic flow. It learns from the simulation through trial and error, using neural networks and experience replay to develop a policy that reduces vehicle waiting times and enhances overall traffic conditions.