from fastapi import FastAPI

from fastapi.middleware.cors import CORSMiddleware

import traci

import uvicorn

import threading

import time

from agent import DQNAgent

from simulation import get\_state

CONFIG\_FILE = "cross3ltl.sumocfg"

lanes = ["1fi\_0", "2fi\_0", "3fi\_0", "4fi\_0"]

app = FastAPI()

app.add\_middleware(

CORSMiddleware,

allow\_origins=["\*"], allow\_methods=["\*"], allow\_headers=["\*"],

)

agent = DQNAgent(state\_size=5, action\_size=2)

metrics = {"step": 0, "queue\_lengths": [0, 0, 0, 0], "wait\_time": 0}

MIN\_GREEN = 10

MAX\_GREEN = 30

def run\_simulation():

global metrics

traci.start(["sumo-gui", "-c", CONFIG\_FILE])

# start phase = lane with most vehicles

queue\_lengths = [traci.lane.getLastStepHaltingNumber(l) for l in lanes]

best\_lane = queue\_lengths.index(max(queue\_lengths))

current\_phase = best\_lane \* 2

traci.trafficlight.setPhase("0", current\_phase)

traci.trafficlight.setPhaseDuration("0", MIN\_GREEN)

time\_in\_phase = 0

prev\_wait = 0

for step in range(200):

traci.simulationStep()

time\_in\_phase += 1

state = get\_state()

action = agent.act(state)

# queue info

queue\_lengths = [traci.lane.getLastStepHaltingNumber(l) for l in lanes]

max\_queue = max(queue\_lengths)

current\_queue = queue\_lengths[current\_phase // 2]

can\_switch = False

# ---- SWITCH LOGIC ----

# 1. Never give green to empty lane if others waiting

if current\_queue == 0 and max\_queue > 0:

can\_switch = True

# 2. Force switch if max green reached

elif time\_in\_phase >= MAX\_GREEN:

can\_switch = True

# 3. After min green, if another lane has more cars → switch

elif time\_in\_phase >= MIN\_GREEN and current\_queue < max\_queue:

can\_switch = True

# ----------------------

if can\_switch:

current\_phase = (

(current\_phase + 2)

% len(traci.trafficlight.getCompleteRedYellowGreenDefinition("0")[0].phases)

)

traci.trafficlight.setPhase("0", current\_phase)

traci.trafficlight.setPhaseDuration("0", MIN\_GREEN)

time\_in\_phase = 0

# reward & training

total\_stopped = sum(traci.lane.getLastStepHaltingNumber(l) for l in lanes)

reward = prev\_wait - total\_stopped

prev\_wait = total\_stopped

next\_state = get\_state()

agent.train(state, action, reward, next\_state, done=False)

# update shared metrics

metrics = {

"step": step,

"queue\_lengths": queue\_lengths,

"wait\_time": total\_stopped,

"phase": current\_phase,

"time\_in\_phase": time\_in\_phase,

}

time.sleep(0.5)

traci.close()

@app.get("/metrics")

def get\_metrics():

return metrics

if \_\_name\_\_ == "\_\_main\_\_":

sim\_thread = threading.Thread(target=run\_simulation, daemon=True)

sim\_thread.start()

uvicorn.run(app, host="0.0.0.0", port=8000)