Workshop No.1 Systems Sciences Foundations

Juan Santiago Ramos Ome, Arlo Nicolas Ocampo

April 8, 2025

1 Introduction

Develop an autonomous agent capable of learning and adapting to a simulated environment using reinforcement learning. The officer will monitor traffic lights at a single intersection with multiple traffic participants (cars, motorcycles, and pedestrians) or at two consecutive traffic lights on the same main street.

2 System Requirements Document

2.1 Functional Specifications

An autonomous agent will be designed to control and manage two consecutive traffic lights located on the same street, with the goal of optimizing traffic flow with reinforcement learning in the simulation, the agent learns to respond to the traffic lights by observing the traffic conditions, in orden to minimize waiting time and avoid traffic into the two points.

The integration of sensors, actuators, and reward functions which are essential for the interaction between the agent and the environment.

Table 1: Sensors

sensors	**What does the sensor do?**	**Data provided by the sensor**
Camera in the two traffic lights	counts stopped and moving vehicles	number of vehicles before, between and after traffic lights
status timer	time spent in a state (red, yellow, green)	time in seconds
passage camera	counts the vehicles that pass at change the state of the traffic light	average number of vehicles per unit of time
people detector	detects if there are people waiting	yes or no

Table 2: Actuators

Actuators	**what the actuator does**
first traffic light controller (A)	changes the state of the traffic light according to the agent (red, yellow, green)
second traffic light controller (B)	changes the state of the traffic light according to the agent (red, yellow, green))
timer	adjusts the time based on the defined parameters

Table 3: Reward functions

Situation	**Type of score reward**	**why the reward?**
correct flow between traffic lights	+1 point	promotes the correct synchronization of traffic lights
very long waiting times for vehicles (piling up at one or both traffic lights)	-1 or -2	punishes the long wait
accident or vehicle stopped between two traffic lights	-2 or -3	punishment for block the road
crossing time decreases	+ 0.5 or +1	reward for speeding the crossing of people
Unnecessary or very rapid action that affects the crossing of vehicles	-1 point	Very fast or unnecessary light change in vehicles or people

2.2 Use Cases

Use case 1

Title: Optimizing vehicle flow between the two traffic lights
Priority: High
Estimate: 5 Days

User story: As an intelligent traffic control agent, I want to learn to coordinate two consecutive traffic lights, so that vehicle flow on the main road is optimized and waiting time is minimized.

Acceptance Criteria:

- Given the agent is in a simulated environment during training

- When it makes decisions to switch traffic lights
- Then it learns to reduce traffic congestion and improve average travel time

Use case 2

Title: Reducing the average wait time
Priority: Medium
Estimate: 3-4 Days

User story: As an autonomous control system, I want to adjust the traffic light in real time based on traffic variation, so that the system remains efficient during both peak and low traffic hours.

Acceptance Criteria:

- Given traffic conditions vary over time
- When the agent receives updated sensor observations
- Then it modifies to maintain optimal the traffic flow without manual intervention

3 High-Level Architecture

The component diagram and the feedback loops are uploaded in the GitHub repository.

4 Preliminary Implementation Outline

4.1 Potential frameworks

1. Gymnasium:
Gymnasium is a standard framework for defining reinforcement learning environments. The principal advantages of using it in our agent are:

- It will allow to build the traffic light environment with a Python class that defines states, actions, and rewards.
- It allows to observe the environment to track the agent behavior in real time. This will be useful during the agent testing period.
- It is compatible with RL libraries, including Stable-Baselines3.
- It is one of the best for working on experimental projects.

How to apply it to the agent? It will be used to simulate the traffic system with vehicles moving between two control points (traffic light A and traffic light B) with defined transition rules, virtual sensors, and rewards.

2. Stable-Baselines3:
Stable-Baselines3 is an implementation of RL algorithms and provides algorithms that are ready to use like DQN.

- It allows the use of advanced algorithms like DQN, which is ideal for environments with discrete spaces, in this case traffic lights.
- It is an interface that is not complex to use.
- It will allow metrics to be recorded, in this case average waiting time and time spent standing still.
- Rewards can be rewritten without rewriting all the system.

How to apply it to the agent? It will allow to start a DQN agent that learns to change traffic lights based on observations like the number of vehicles and waiting time, and it will allow you to evolve to more complex models if desired.

4.2 Timeline for the transition from basic Q learning to more advanced DQN approaches

Week 1
Problem analysis and system definition

- Scenario analysis: the street with two traffic lights
- Identify the actors: sensors, environment and vehicles
- Review the functional specifications, primarily for the sensors
- Literature review of frameworks and concepts

Objetive: Document the system requirements and describe the agent and the technologies to be used

Week 2
Gymnasium design and implementation

- Define the environment states (number of vehicles on the street, people, and time), and also define the actions (changing the state of the traffic lights)
- Logical structure of the environment, rewards and sensors
- Implement basic methods for basic traffic simulation

Objetive: Design a basic functional Gym environment (Python, Gymnasium, NumPy)

Week 3
Traffic simulation and implement sensor logic

- Vehicle Simulation Programming
- Traffic Light Logic (timers)
- Sensor modeling (counting in zones)

Objetive: Perform the traffic simulation and gather observations (Python, Gymnasium, logical structures)

Week 4
Training with basic Q learning

- Implementation of RL agent with Q table
- Training in multiple scenarios

Objetive: Initial training with Q table and analysis of initial performance

Week 5
Evaluation of Q learning and parameter tuning

- Analysis of the reward function and observe if it is being met
- View data on wait times and decisions

Objetive: Observe and correct the Q agent with clear data

Week 6
Transition to DQN

- Implementation of Stable-Baselines3
- Implementation of DQN agents
- Training and observe if there is room for improvement

Objetive: Implementation of the first DQN model (Gymnasium, State-Baselines3)

Week 7
Evaluation and optimization of the agent DQN

- Review the reward function to ensure it delivers the expected results Objective

Objetive: Have the agent optimized and the reward function stable

Week 8
Compare Q learning and DQN

- Adjust average time, congestion, zone, and shifts
- View and interpret the results

Objetive: Compare performance

Week 9
Documentation and delivery

- Final report
- System architecture diagrams

Objetive: Final delivery with graphics and code

5 References

- Gymnasium documentation. (s. f.). https://gymnasium.farama.org/index.html
- Stable-Baselines3 Docs - Reliable Reinforcement Learning Implementations — Stable Baselines3 2.6.1a0 documentation. (s. f.). https://stable-baselines3.readthedocs.io/en/master/
- Lin, R. (2022, 3 march). Create your own environment using the OpenAI Gym library — GridWorld. Medium. https://reneelin2019.medium.com/create-your-own-environment-using-the-openai-gym-library-gridworld-8ca9f18e00a4
- Vitality Learning. (2024, 15 november). Solving the Taxi Problem Using OpenAI Gym and Reinforcement Learning. Medium. https://vitalitylearning.medium.com/solving-the-taxi-problem-using-openai-gym-and-reinforcement-learning-0317e089b48f
- Tutorial: An Introduction to Reinforcement Learning Using Open AI Gym (s. f.). https://www.gocoder.one/blog/rl-tutorial-with-openai-gym/