# Workshop No.2 Systems Sciences

Juan Santiago Ramos Ome, Arlo Nicolas Ocampo

May 9, 2025

## 1 Introduction

This workshop delves into the analysis and refinement of an autonomous agent designed for traffic light control, utilizing concepts from dynamical systems. The primary goal is to enhance the agent's ability to adapt to complex traffic scenarios by employing tools such as Markov Decision Processes (MDP) and reinforcement learning, specifically Q-learning. Furthermore, the workshop explores the use of phase portraits to visualize system behavior and emphasizes the importance of stability and convergence in the agent's learning process. The design of advanced feedback mechanisms is also addressed to enable the agent to respond effectively to uncertain and dynamic environments.

## 2 System Dynamics Analysis:

### 2.1 Mathematical/Simulation Model

For the agent to learn to intelligently control traffic lights, it is necessary to represent mathematical the environment and its decisions. Two principal models that we are used for this: the Markov Decision Process (MDP) and the Q-learning algorithm.

The MDP allows the operation of the traffic system to be described as a series of diferent states (how many vehicles there are, the state of the traffic ligth, and actions like changing or maintaning the traffic light phase, and rewards like reducing the waiting time). This helps to define what the agent observes and what it can do.

Then the Q-learning is applied, a reinforcement learning algorithm that allows the agent to learn through trial and error. With the simulation the agent tests different actions, observes the results, and learns which decisions are best for improving vehicular flow. Over time, the agent learns a strategy that optimizes traffic light behavior.

**MDP**

Is a decision mathematical framework for describing decision-making problems in dynamic and uncertain environments, such as traffic.

$$MDP = (S, A, P, R) \tag{1}$$

Where:

- S: Describe the current situation of the environment, in your case, the traffic and traffic lights.
- A: They are the decisions that the agent can make in a given state.
- P: Describes the probability of the system moving from one state to another.
- R: It is a numerical value that represents how good an action was in a given state.

## 2.2 Phase Portraits or Diagrams

Phase portraits are a way to visualize the behavior of a dynamical system in its state space. The "state space" is the set of all possible states that the system can be in. In our traffic control scenario, a "state" could be defined by variables like:

- Number of vehicles before traffic light A
- Number of vehicles between traffic lights A and B
- Number of vehicles after traffic light B
- Average waiting time at each light
- Current phase of each traffic light (red, yellow, green)

And a probable phase portrait for this project could be a portrait where:

- X-axis = Number of vehicles before traffic light A
- Y-axis = Average waiting time at traffic light B

Attractors would represent desirable traffic flow patterns. For example:

- A stable attractor might be a point with "moderate vehicle numbers before A" and "low average waiting time at B." This indicates efficient traffic flow.
- The agent's goal is to guide the system towards these attractor states by making appropriate traffic light decisions.

In traffic control, chaos could manifest as:

- Unexpected surges in traffic that lead to oscillations in waiting times and vehicle numbers.

Swift changes could be present as:

- A long arrow might represent a sudden increase in waiting time due to a traffic incident.
- The agent's actions (e.g., changing traffic light phases) should ideally produce smooth transitions in the phase portrait, avoiding abrupt changes that could disrupt traffic flow.

Phase portraits can show how the system responds to different inputs or conditions.

- Possible scenarios: "high congestion", "low demand", "different arrival rates of vehicles," etc.
- Comparing these phase portraits will reveal how the agent adapts (or fails to adapt) to varying inputs.

Stability is indicated by trajectories that remain within a bounded region of the phase portrait. Convergence is when trajectories approach an attractor over time.

- The agent should promote stability, preventing traffic conditions from spiraling out of control (e.g., unbounded queues).
- Convergence means that the agent learns to consistently guide the system towards optimal traffic flow.

# 3 Feedback Loop Refinement:

## 3.1 Enhanced Control Mechanisms

**Table 1: Additional sensors**

| **sensors** | **What does the sensor do?** | **Data provided by the sensor** |
|---|---|---|
| Speed sensor | Measures the average speed of vehicles in each lane | Average speed (km/h or m/s) |
| Vehicle distance sensor | Estimate traffic congestion in real time | Average distance between vehicles |
| People waiting sensor | how long a person has been waiting | people waiting time |

**Table 2: More granular rewards**

| Reward signal | Type of score reward | Advantage |
|---|---|---|
| Fluency Reward | +1 . crossing vehicles | the number of vehicles crossing increases |
| Penalty for prolonged waiting of people | -1 . waiting people | Prevents the system from ignoring the people |
| You have to cross cars when it's green | -2 points if there is a green light but no vehicle is crossing | Improves efficient use of green time |

## 3.2 Stability and Convergence

Stability and convergence are critical concepts for our traffic control agent. We need to ensure that the agent not only performs well but also does so reliably and consistently.

- **Stability Criteria**:
  - **Bounded Behavior**: The system should exhibit bounded behavior, meaning that key metrics like vehicle queue lengths and waiting times remain within acceptable limits. The agent should prevent the system from reaching extreme states like gridlock.
  - **Resilience**: The agent should be resilient to disturbances, such as sudden changes in traffic flow or sensor noise, and maintain stable performance.
- **Convergence**:
  - **Performance Improvement**: Convergence refers to the agent's ability to improve its performance over time. In our case, this means that through learning, the agent should progressively reduce congestion and waiting times.
  - **Steady State**: Ideally, the agent's performance should converge to a steady state, where further learning does not significantly change its effectiveness.
- **Evaluation**:
  - **Theoretical Evaluation**: We will use the underlying mathematical framework (Markov Decision Process) and reinforcement learning principles (Q-learning, DQN) to provide a theoretical basis for stability and convergence.
  - **Practical Evaluation**:
    * **Simulation Analysis**: We will analyze simulation results to measure stability and convergence. This involves tracking metrics like average waiting time, vehicle throughput, and queue lengths over extended simulation periods.

    * **Scenario Testing**: Testing the agent under various scenarios (high/low traffic, incidents) will further validate its stability.

# 4 Iterative Design Outline:

## 4.1 Update to Project Plan

To effectively incorporate dynamic systems concepts and achieve more advanced agent behavior, the project plan needs to be updated with the following:

- **Enhanced Data Structures**:
  - **Time Series Data**: We need to handle time series data to capture the temporal nature of traffic flow. This includes storing sequences of traffic conditions, agent actions, and performance metrics.
  - **Dynamic State Representation**: Instead of simple static state representations, we may need more complex structures (e.g., graphs) to represent the dynamic relationships between different parts of the traffic network.
- **Advanced Algorithms**:
  - **Recurrent Neural Networks (RNNs)**: To capture temporal dependencies in traffic patterns, we can explore using RNNs or LSTM networks.
  - **Kalman Filtering**: Implement Kalman filters to estimate traffic flow parameters and predict future states, enhancing the agent's ability to adapt to uncertainty.
  - **Exploration-Exploitation Strategies**: Implement more sophisticated exploration strategies in reinforcement learning to balance exploring new actions and exploiting known good ones.
- **Frameworks and Tools**:
  - **Deep Learning Frameworks**: Integrate deep learning frameworks like TensorFlow or PyTorch to support more complex models like RNNs and DQN variants.
  - **Traffic Simulation Tools**: Consider using more advanced traffic simulation tools (e.g., SUMO) to create highly realistic and complex traffic scenarios.
- **Additional Planning**:
  - **Computational Resources**: Account for the increased computational demands of more complex models and simulations.
  - **Validation**: Emphasize rigorous validation techniques to ensure the agent's robustness and generalization ability.

- **Feedback Loop Design**: Refine the design of feedback loops to enable the agent to adapt more rapidly to changing conditions.

**Table 4: Simulation parameters**

| **Parameter** | **Description** | **Example** |
|---|---|---|
| Simulation duration | Total time of training | 30 min to 1 hour |
| Decision making | How many seconds the agent act | Every 30 seconds |
| times of change of states | Minimum time to stay green/red light | 30 seconds |

**Scenario variations** : To test the adaptability of the program, it will be create scenarios such as:

- High congestion, to see if the agent invites traffic jams
- Low demand for vehicles, to see if the agent optimizes the use of the traffic light
- Weather conditions, see if the agent adapts phases for safety

# References


[1] Gymnasium documentation. (n.d.). `https://gymnasium.farama.org/index.html`.

[2] Lin, R. Create your own environment using the OpenAI Gym library Grid World. Medium. `https://reneelin2019.medium.com/create-your-own-environment-using-the-openai-gym-library-gridworld-8ca9f18e00a4`.

[3] M. Newman, *Networks*. Oxford university press, 2018.

[4] S. H. Strogatz, *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. CRC press, 2018.

[5] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.