Multiagent Deep Reinforcement Learning for Traffic Light Control

By Azlaan Mustafa Samad

Advisor: Frans A. Oliehoek
Asst. Professor
Department of Intelligent Systems
Delft University of Technology.

Advisor: Prof. Kees Vuik
Professor of Numerical Analysis
Director of TU Delft Institute for Computational Science and Engineering.

TU Delft

Motivation

- Traffic jams are a part of everyday problem for commuters.
- Cause of Pollution(air and noise), Health issues, accidents and monetary losses etc.

Facts:
The cost of traffic congestion in the EU is large, estimated to be 1% of the EU's GDP[1].

Long commuters are associated with higher weight, lower fitness levels, and higher blood pressure[2] (2012 study by Washington University.)

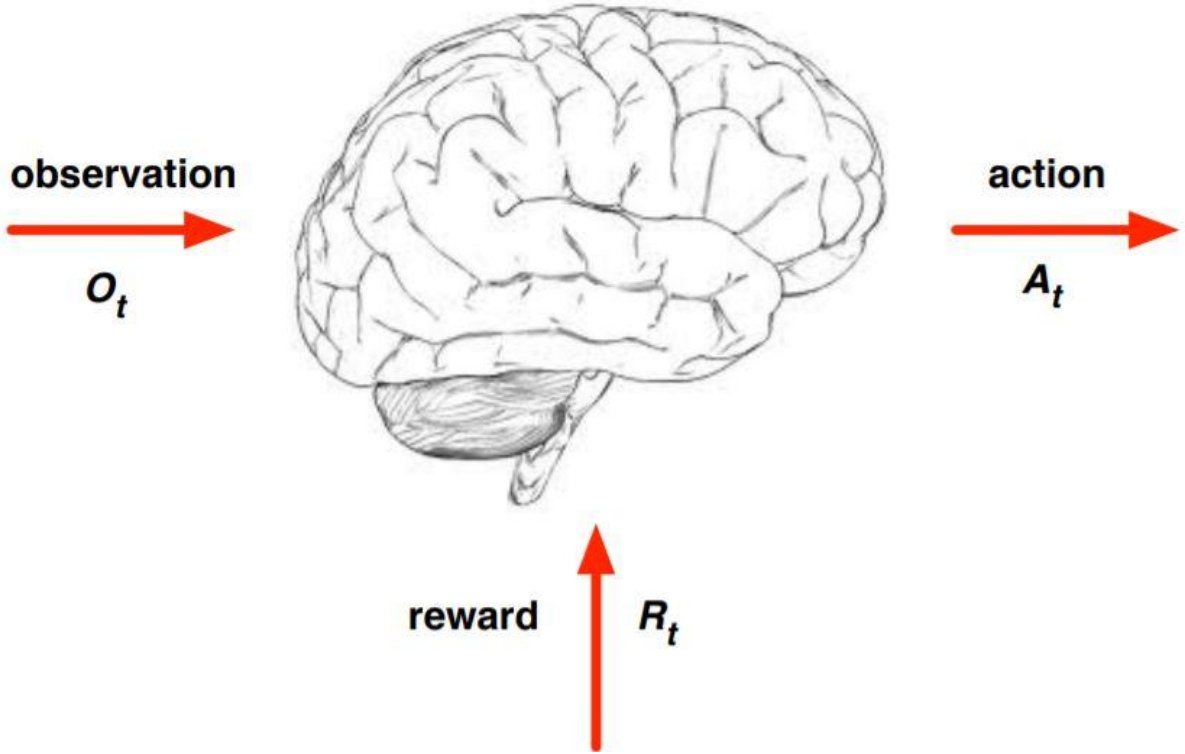
TU Delft

Current Research

- Use of Predefined traffic light cycles.
- Use of Wireless Sensors Networks.
- TRANSYT: optimisation process that adjusts the signal timings with the ultimate aim of reducing the Performance Index to a minimum.
- SCOOT(Split Cycle and Offset Optimisation Technique): uses data from vehicle detectors and optimises traffic signal settings to reduce vehicle delays and stops.
- Use of Deep Reinforcement Learning

Reinforcement Learning

- There is no supervisor, only a reward signal.
- Feedback is delayed, not instantaneous.
- Time really matters (sequential, non i.i.d data).
- Agent's actions affect the subsequent data it receives.


observation
O_t
action
A_t
reward
R_t

Rewards

- A reward R_t is a scalar feedback signal.
- Indicates how well agent is doing at step t.
- The agent's goal is to maximise cumulative reward.

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

$$G_t \doteq R_{t+1} + \gamma G_{t+1}$$

where $\gamma \in [0,1]$ is the discounted term.

Major Components of an RL Agent

An RL agent may include one or more of these components:

- Policy: Agent's behaviour function.
- Value function: How good is each state and/or action.
- Model: Agent's representation of the environment.

Policy

A policy is the Agent's behaviour.

It is a mapping from state to action.

Eg.

Deterministic policy: $a = \pi(s)$
Stochastic policy: $\pi(a|s) = P[A_t = a|S_t = s]$

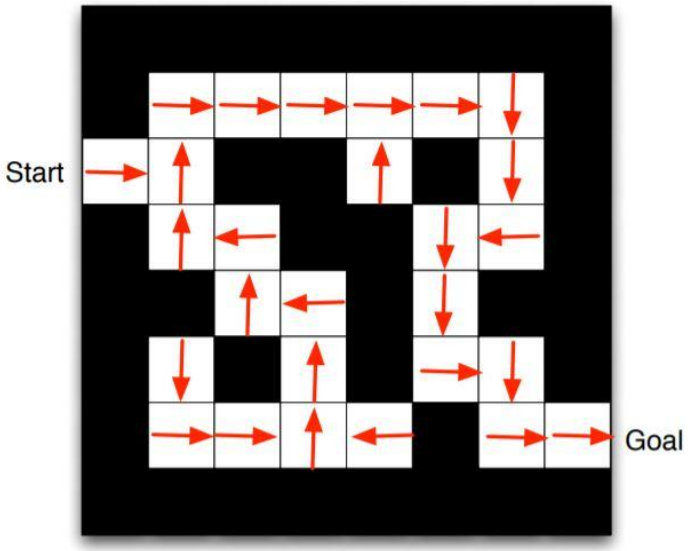
Value Function

- Value function is a prediction of future reward.
- Used to evaluate the goodness/badness of states and therefore to select between actions.

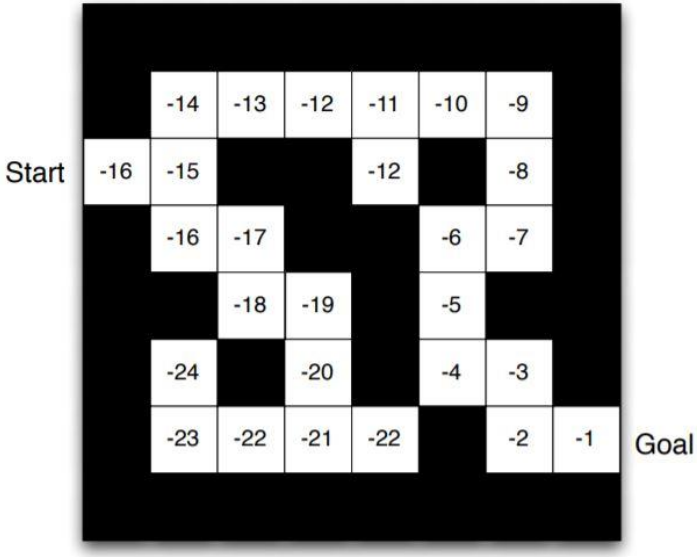
$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s\right]$$

for all $s \in \mathcal{S}$

Maze Example: Policy and Value function



Policy[3]



Value Function[3]

Model

A model predicts what the environment will do next.

Eg.

$$p(s', r \,|\, s, a) \doteq Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$$

Markov Decision Process

- Markov decision processes formally describe an environment for reinforcement learning

- A state S_t is Markov if and only if:

$$P(S_{t+1} \mid S_t, \ldots, S_2, S_1) = P(S_{t+1} \mid S_t)$$

Definition of MDP(tuple)

A Markov Decision Process is a tuple $\langle S, A, P \rangle$

S is the space of possible states.

A is the space of possible actions.

P is a state transition probability matrix.

$$p(s', r \,|\, s, a) \doteq Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$$

Q-Value Function

The action-value function $q_\pi(s, a)$ is the expected return starting from state s, taking action a, and then following policy π.

$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t | S_t = s, A_t = a] = \mathbb{E}_\pi[\sum \gamma^k R_{t+k+1} | S_t = s, A_t = a]$$

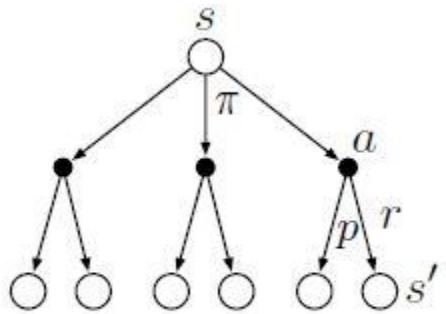
Bellman Equation

It expresses a relationship between the value of a state and the values of its successor states.

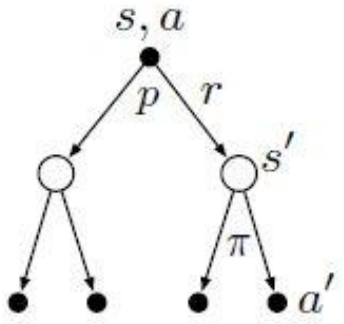
$$v_\pi(s) = \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma v_\pi(s') \right]$$

for all $s \in \mathcal{S}$.

Backup Diagrams

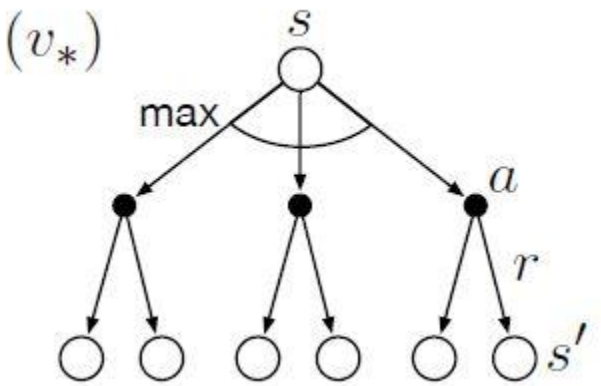


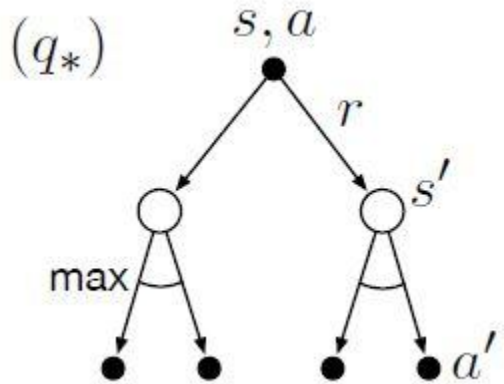
Backup diagram for v_π



q_π backup diagram

Bellman Optimality Equation





$$v_*(s) \doteq \max_\pi v_\pi(s)$$

$$q_*(s,a) \doteq \max_\pi q_\pi(s,a)$$

$$q_*(s,a) = \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a]$$

Reinforcement Learning Algorithms

- Based on Policy:
  - On Policy: Eg. SARSA
  - Off Policy: Eg. Q-Learning
- Based on Model:
  - Model free: Eg. Q-Learning
  - Model Based: Eg. Value or Policy Iteration

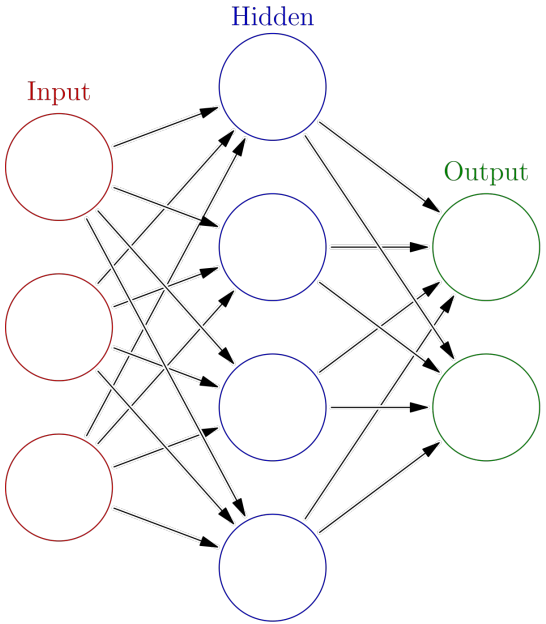
SARSA and Q Learning

The update rule is as given below:

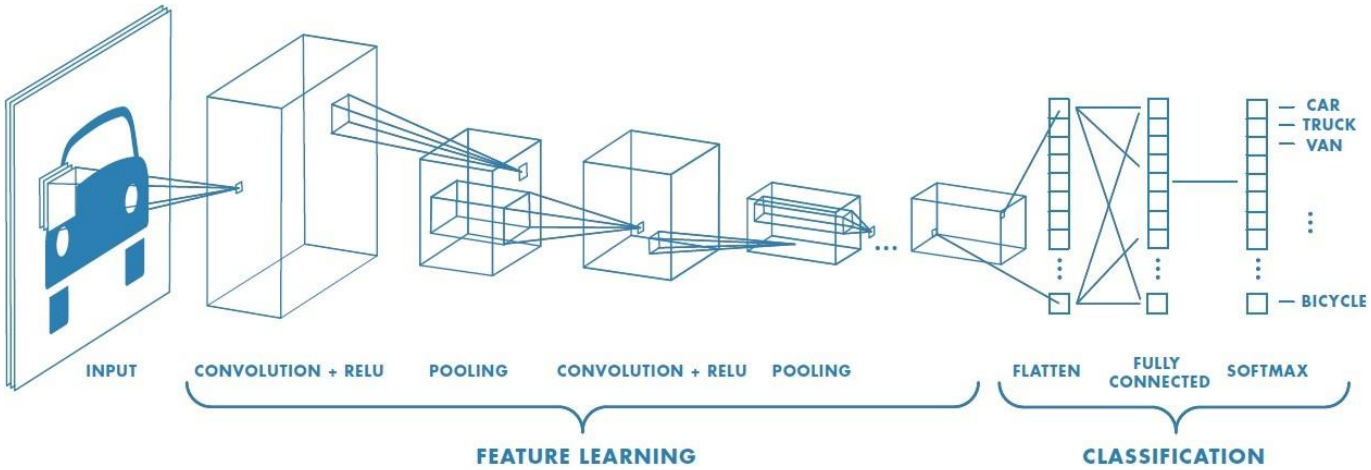
SARSA: $$\mathcal{Q}(s_t, a_t) \leftarrow \mathcal{Q}(s_t, a_t) + \alpha[r_{t+1} + \gamma \mathcal{Q}(s_{t+1}, a_{t+1}) - \mathcal{Q}(s_t, a_t)]$$

Q-Learning: $$\mathcal{Q}(s_t, a_t) \leftarrow \mathcal{Q}(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a \mathcal{Q}(s_{t+1}, a) - \mathcal{Q}(s_t, a_t)]$$

Deep Q Learning (Q learning with Function Approximation)



- Approximating the optimal Q-values $Q_*(s, a)$ by a parameterized deep neural network (DNN) such that the output of the neural network $Q(s, a; \theta) \approx Q_*(s, a)$, where θ are features/parameters.
- These weights can be updated using gradient descent methods, minimizing the mean squared error between the current estimate of Q(s; a) and the target, which is defined as the true Q-value of the s; a-pair under policy π, $Q_\pi(s; a)$.



A typical CNN(Convolutional Neural Network)

Problems with Deep Q-Learning

- Correlation in the sequence of observation:
  - Solution: Experience Replay
- Change in target Q value due to updating:
  - Solution: Periodical updation of the Target Q-value.

Experience Replay

- Storing of agent's experience e_t (over many episodes):
$$e_t = (s_t, a_t, r_t, r_{t+1})$$
- This is saved into a Dataset such as:
$$D_t = \{e_1, e_2, \ldots, e_t\}$$
- Q-value updates are applied to these randomly selected sample of experience from pool of stored samples.

Target Network

- Updating the Target Q-value after a certain number of steps. Eg. Soft Update

$$\theta' = \beta\theta + (1-\beta)\theta'$$

where, β is the update rate and β <<1.

Algorithm

Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$

For episode $= 1, M$ **do**

  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$

  For $t = 1$,T **do**

    With probability ε select a random action a_t

    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$

    Execute action a_t in emulator and observe reward r_t and image x_{t+1}

    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

    Store transition $\left(\phi_t, a_t, r_t, \phi_{t+1}\right)$ in D

    Sample random minibatch of transitions $\left(\phi_j, a_j, r_j, \phi_{j+1}\right)$ from D

    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^{-}\right) & \text{otherwise} \end{cases}$

    Perform a gradient descent step on $\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$ with respect to the network parameters θ

    Every C steps reset $\hat{Q} = Q$

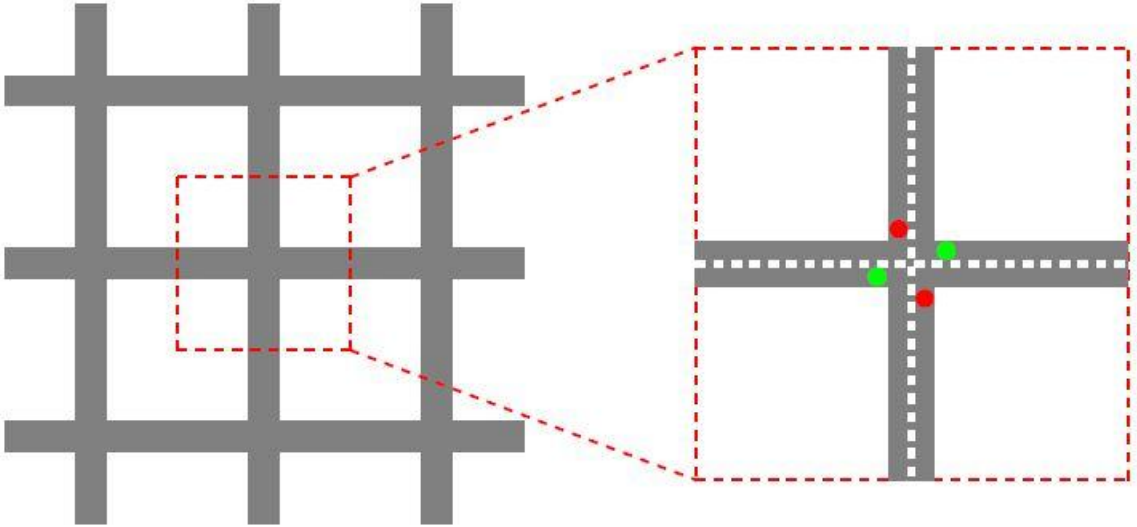
  End For

End For

SUMO(Simulation of Urban MObility)

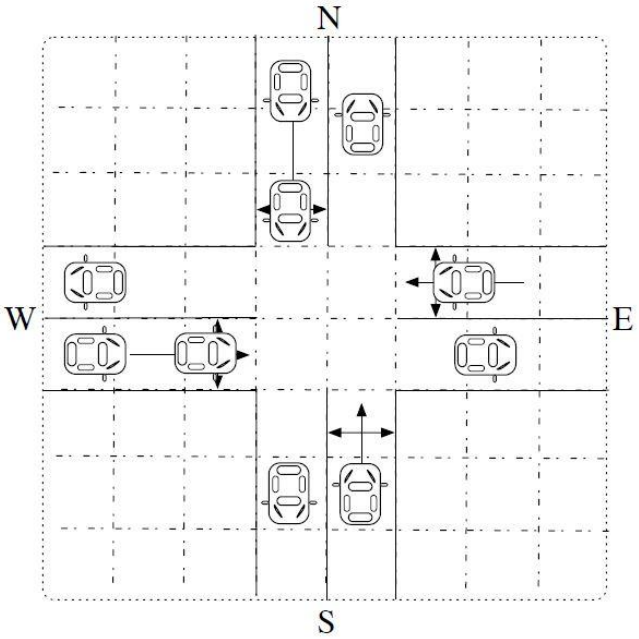
- free, open source software that allows for a realistic simulation of different traffic networks.
- implemented in C++
- allows to import maps from OpenStreetMap etc.
- plethora of tools which can be used for visualisation, emission calculations, network importing and finding the route

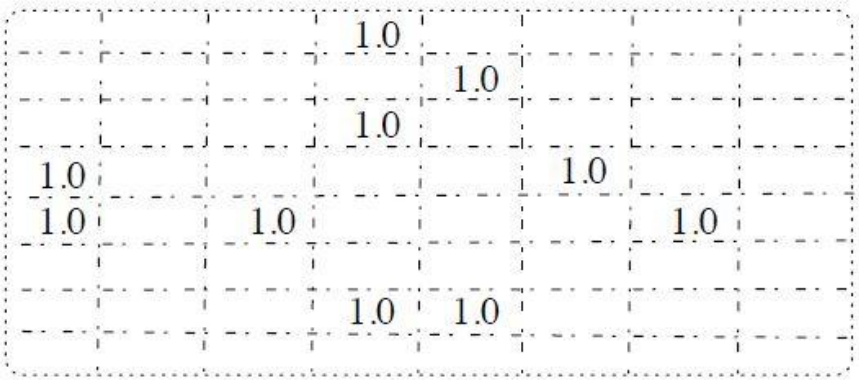
Single Agent Traffic Flow



A traffic light agent within a larger traffic network. The coloured circles represent the traffic light setting for each lane[5].

States





Snapshot of traffic intersection at a particular time[4]

Rewards

- Waiting time
- Queue length
- Average speed of cars in a lane.

- Eg. the absolute negative difference between queue length in north-south/south-north direction and those in east-west/west-east direction, i.e.

$$r_t^{TLS_i} = -\left| maxq_t^{WE} - maxq_t^{NS} \right|$$

[6]

Rewards(cont.)

- Liang et al.[4] defines the rewards as the change of the cumulative waiting time between two neighboring cycles.

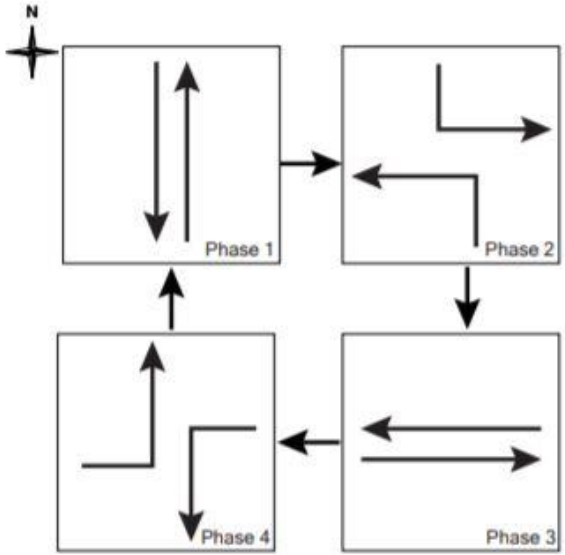
$$r_t = W_t - W_{t+1}$$

$$W_t = \sum_{i_t=1}^{N_t} w_{i_t,t}$$

Here, the waiting time of the vehicle i till the t^{th}cycle is denoted by:

$$w_{i_t,t}, (1 \leq i_t \leq N_t)$$

Actions



The four phases of traffic lights.[6]

Yellow Light

- $T_{yellow} = \frac{v_{max}}{a_{dec}}$
- *Fixed time cycle*
- *As an action itself.*

TU Delft

Multi-Agent Traffic Flow

- A multi-agent system (MAS) is a system of multiple interacting agents within the same environment.

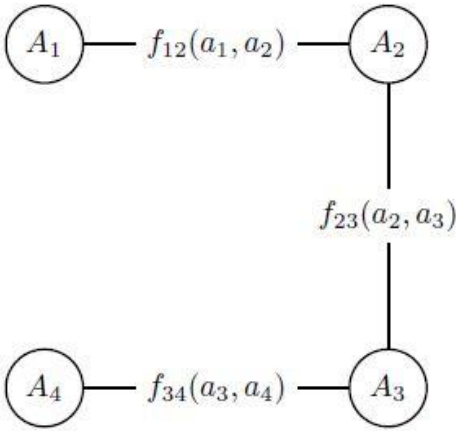
- In a cooperative multi-agent system (CMAS), agents cooperate to reach a common goal, often to maximize a common reward. Thus, in a CMAS, at every time step, the agents need to find an optimal joint action:

$$\overrightarrow{a^*} = (a_1, a_2, \ldots, a_N)$$

Difficulties with MARL

- Exponential growth in *joint* action space due to increase in no. of agents.
- combining all locally optimal actions into a joint action is not guaranteed to reach the global optimum.
- Moving targets due to non-stationarity.

Coordination Graphs

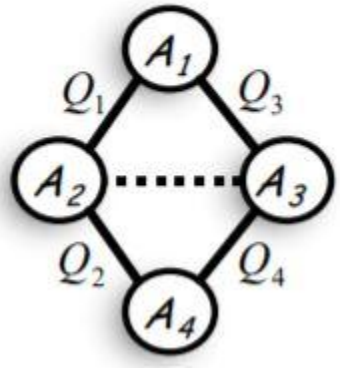


$$CG(a_1, a_2, a_3, a_4) = f_{1,2}(a_1, a_2) + f_{2,3}(a_2, a_3) + f_{3,4}(a_3, a_4)$$

$$\overrightarrow{a^*} = arg \max_{\vec{a}} g(\vec{a})$$

$$\max_{\vec{a}} g(\vec{a}) = \max_{a_1} [\max_{a_2} [\, f_{12}(\cdot) + [\, \ldots \max_{a_N} f_{N-1,N}(\cdot)]]]$$

Variable Elimination



$$\max_{a_1,a_2,a_3,a_4} Q_1(a_1, a_2) + Q_2(a_2, a_4) + Q_3(a_1, a_3) + Q_4(a_3, a_4)$$

$$\max_{a_1,a_2,a_3} Q_1(a_1, a_2) + Q_3(a_1, a_3) + \max_{a_4}[Q_4(a_3, a_4) + Q_2(a_2, a_4)]$$

$$\max_{a_1,a_2,a_3} Q_1(a_1, a_2) + Q_3(a_1, a_3) + f_4(a_2, a_3)$$

$$\max_{a_1,a_2} Q_1(a_1, a_2) + f_3(a_1, a_2)$$

where, $f_3(a_1, a_2) = \max_{a_3}[Q_3(a_1, a_3) + f_4(a_2, a_3)]$

$$f_2(a_1) = \max_{a_2}[Q_1(a_1, a_2) + f_3(a_1, a_2)]$$

$$f_1 = max_{a_1} f_2(a_1)$$

Max-Plus[8]

$G = (V, E)$ Representation of coordination graphs with *V* vertices and *E* the edges.

$$\mathrm{u}(\mathrm{a}) = \sum_{i \in V} f_i(a_i) + \sum_{(i,j) \in E} f_{ij}(a_i, a_j)$$

Global Payoff

Local Payoff

denotes the pair of neighboring agents.

Coordinated Payoff

$$\mu_{ij}(a_j) = \max_{a_i} \{ f_i(a_i) + f_{ij}(a_i, a_j) + \sum_{k \in \Gamma(i) \setminus j} \mu_{ki}(a_i) \} + c_{ij}$$

Maximum payoff *i* can achieve for a given action of *j*.

all neighbours of *i* except *j*.

normalisation vector

Max-Plus[8]

At convergence, we define: $\mathrm{g_i}(\mathrm{a_i}) = f_i(a_i) + \sum_{j \in \Gamma(i)} \mu_{ji}(a_i)$

It can be shown that: $\mathrm{g_i}(\mathrm{a_i}) = \max_{\{\mathrm{a}' | \mathrm{a}'_\mathrm{i} = \mathrm{a_i}\}} \mathrm{u}(\mathrm{a}')$

Each agent *i* now selects its optimal action: $a_i^* = \mathrm{argmax}_{\mathrm{a_i}}\, \mathrm{g_i}(\mathrm{a_i})$

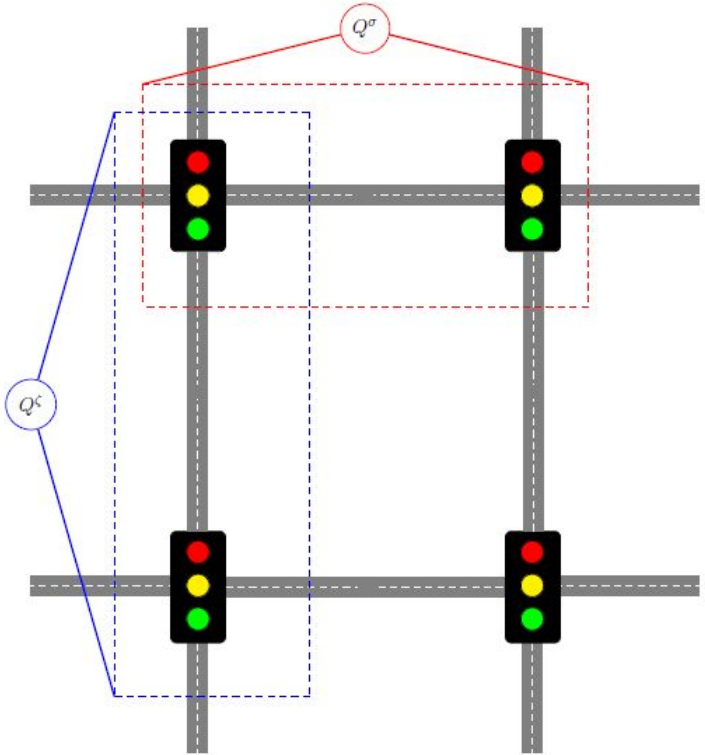
If there is one maximizing action for every agent *i*, the global optimal joint action is unique. $a^* = \mathrm{argmax}_{\mathrm{a}}\, \mathrm{u}(\mathrm{a})$

TU Delft

Transfer Planning[7]

- Once the joint Q-value function for the two-agent scenario is learned, this is used as the source problem in transfer planning.
- this is used as the source problem in transfer planning.
- After finding the local joint Q-value function for the factors in the multi-agent problem, the Q-function is reused for all similar factors in the larger multi-agent problem.

Transfer planning for traffic light control.



Transfer Planning + Max-Plus

Thus, the joint local Q-value Q_f is learned for each factor f in coordination graph CG, separately from the other agents.

Then, during execution, Q_f is used to compute local payoff functions for use in e.g. max-plus.

Advantages

- Less time spent.
- Similar factored graphs can be learnt from a single source.
- No more source problem than the factors are needed.

Research Questions

- How to decide the reward function in order to effectively coordinate traffic and thereby reduce traffic jams?
- How to scale from few agents involving few intersections to a suburb of a city and finally the entire city?
- Training the agent for different types of intersections which we come across in the real life scenario and extending it to bigger maps.
-

References:

[1] Commission of the European communities. White paper-European transport policy for 2010: time to decide. Office for Official Publications of the European Communities, 2001.

[2] GMA News Online. Stress, pollution, fatigue: How traffic jams affect your health.

[3] David Silver: Reinforcement Learning course in Imperial College London.

[4] Xiaoyuan Liang, Xunsheng Du, Guiling Wang, and Zhu Han. Deep reinforcement learning for traffic light control in vehicular networks.arXivpreprint arXiv:1803.11115, 2018.

[5] Elise van der Pol. Deep reinforcement learning for coordination in traffic light control. 2016

[6] Yilun Lin, Xingyuan Dai, Li Li, and Fei-Yue Wang. An efficient deep reinforcement learning model for urban traffic control.arXiv preprintarXiv:1808.01876, 2018.

[7] Frans A Oliehoek, Shimon Whiteson, and Matthijs TJ Spaan. Approximate solutions for factored Dec- POMDPs with many agents. In Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems, pages 563-570. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[8] Carlos Guestrin, Michail Lagoudakis, and Ronald Parr. Coordinated reinforcement learning. In ICML, volume 2, pages 227{234, 2002}.

TU Delft

Thank You!