Multiagent Deep Reinforcement Learning for Traffic Light Control

Azlaan Mustafa Samad
Applied Mathematics Student (COSSE program)
A.M.Samad@student.tudelft.nl

Supervisor
Frans A. Oliehoek
Associate Professor in the Department of Intelligent Systems
Delft University of Technology
F.A.Oliehoek@tudelft.nl

Background

Existing traffic light causes numerous issue such as delays, accidents, noise and air pollution. The current traffic light hardly takes into account the real time scenario but changes the light based on a predefined cycle. This leads to unoptimised and inefficient traffic flow.

Introduction

In some recent research work[1], reinforcement learning (RL) is being used in order to control the traffic light by treating the problem as a sequential decision making problem, wherein the agent learns from trial and error by interacting with the environment and the goal of the agent is to maximise the reward in long term. In case of application of RL in traffic flow problems, Markov Decision Process is used to model the problem, where the states represent the different configurations of the traffic light at the intersection, actions are the changing of the traffic lights, rewards are formulated using factors like waiting time, delays, teleports, emergency stops etc. A Deep neural network or Deep Q-Learning is used to find an optimal policy which chooses action in order to maximise the total cumulative reward of the state. After learning to optimise the traffic flow for a single intersection, the same idea can be extrapolated to multi agent system, where the number of intersection is more than one. This can be done using the idea of transfer planning[2] for training the agent for smaller subproblems i.e global coordination is decomposed into local coordination using the theory of coordination graphs to find a joint global optimal action using max-plus algorithm[3] [4].

Approach/Methods

Reinforcement learning is an area of machine learning where the agent interacts with the environment and ultimately tries to maximise some notion of cumulative rewards by finding an optimal policy.
Here the environment is formulated using Markov Decision Process and it utilises dynamic programming to find the solution.

[1] "(PDF) Multi-Agent Reinforcement Learning for Traffic Light Control.." https://www.researchgate.net/publication/221346141_Multi-Agent_Reinforcement_Learning_for_Traffic_Light_Control. Accessed 7 Nov. 2018.
[2] "Approximate solutions for factored Dec-POMDPs with many agents." 6 May. 2013, http://dl.acm.org/citation.cfm?id=2485010. Accessed 7 Nov. 2018.
[3] "Using the max-plus algorithm for multiagent decision making in ...." https://dl.acm.org/citation.cfm?id=2124162. Accessed 7 Nov. 2018.
[4] "Multiagent reinforcement learning for urban traffic control using ...." 15 Sep. 2008, https://dl.acm.org/citation.cfm?id=3120893. Accessed 7 Nov. 2018.

Markov Decision process(MDP)
Formally, an MDP is a four-tuple $< S; A; R; T >$ where,
S is the space of possible states;
A is the space of possible actions;
$R^{a}_{ss'}$ is a reward function specifying the reward *r* for taking action a in state s and ending up in state *s'*.
$T^{a}_{ss'}$ *ss'* is a transition function specifying the probability of taking action a in state s and ending up in state *s'*.
The agent's goal is to maximize its reward over time, giving slightly more preference to short-term than to long-term reward. This goal is captured in the return, the discounted cumulative reward over time:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

where γ is a discount factor such that $0 < \gamma \leq 1$, meaning that future rewards are discounted exponentially.
To maximize the return, the agent finds a policy π, a strategy for choosing an action *a* given a state *s*.
To choose the optimal action in a state, an agent needs a function defined over states and actions. This is the Q-value function $Q : S \times A \to \Re$, which estimates the expected value of taking action a in state s and following π afterwards:

$$Q^{\pi}(s,a) = \sum_{s' \in S} T^{a}_{ss'} [R^{a}_{ss'} + \gamma V^{\pi}(s')]$$

If T and R are known, the optimal policy can be found by planning, using dynamic programming methods that exploit the recursive definition of the value function.

Q-learning with Function Approximation
When the number of states and actions are very large, the Q value is approximated using some function approximation by applying supervised learning techniques.

Here Q is a function parametrized by learned weights . These weights can be updated using *gradient descent method*, 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, $Q^{\pi}(s,a)$.

The gradient descent update can be derived by taking the derivative of the mean squared error (MSE):

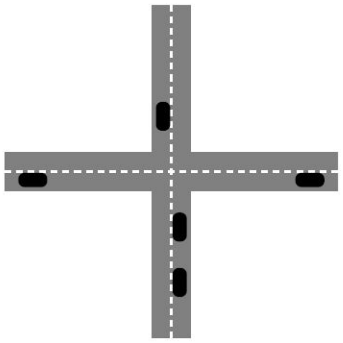
$$MSE(\theta) = \sum_{s \in S} P(s) \, [\, Q^{\pi}(s,a;\theta^{*}) - Q_t(s,a;\theta_t)]^2$$

where $P(s)$ is the sampling distribution, or the probability of visiting state s under policy π.

Neural Networks and Deep Reinforcement Learning
Neural networks are the agent that learns to map state-action pairs to rewards. Like all neural networks, they use coefficients to approximate the function relating inputs to outputs, and their learning consists to finding the right coefficients, or weights, by iteratively adjusting those weights along gradients that promise less error.

Deep Reinforcement Learning for Traffic Light Control
In traffic light control, an agent is the intersection of the traffic light, whose goal is to minimise the delay and allow for smooth movement of traffic. SUMO an open source traffic simulator is used to gather the information about the states of the intersection and take actions accordingly.

States
The stated in Deep Q-learning is defined as an image representation. In case of traffic flow, the position matrix is used to represent the position of cars in the network as shown, where one represents the presence of vehicle and zero represents the absence of it.



$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Action Space
The action space are defined by the possible legal traffic light configuration, wherein it is not allowed for vehicles in the intersecting direction to move at the same time.
Rewards
The rewards can be expressed as a weighted sum of delay, emergency stops, teleports and switches.

Coordinated Multi Agent Reinforcement Learning

The extension of the Deep Reinforcement Learning from single agents to multiple agents can be done using transfer planning[2] and max plus algorithm[3 4].
The idea of transfer planning is to try and find a solution for the subproblem involving few agents and use this solution to select actions in a coordinated fashion in the larger target problem using max plus algorithm.
To coordinate between multiple agents, we factorize the global Q-function as a linear combination of local suproblems:

$$\widehat{Q}(s, a) = \sum_e Q_e(s_e, a_e)$$

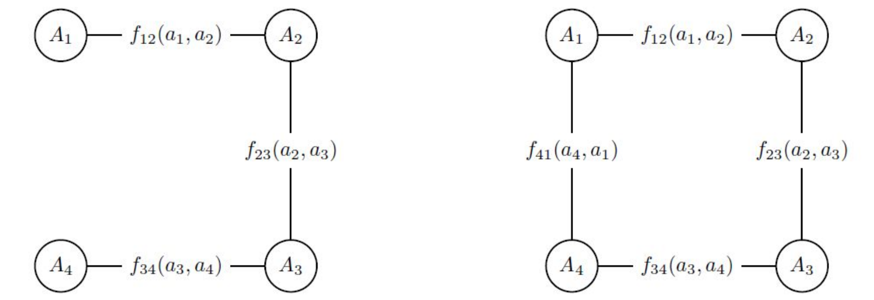
where e corresponds to a subset of the neighbouring agents.
We then use the max-plus coordination algorithm to optimize the joint global action over the entire coordination graph. In transfer planning, instead of training for each sub-problem separately, the source problem's solution can be used for other subproblems as long as their configuration are similar.

Coordination graphs
It is a method of factoring global payoff function g into smaller local factors f_{ij}, where each factor is a smaller function, depending on a subset of the agents in the graph. The CG in the figure below can be decomposed by:

$$CG(a_1, a_2, a_3, a_4) = f_{12}(a_1, a_2) + f_{23}(a_2, a_3) + f_{34}(a_3, a_4)$$

The goal is to find the joint optimal action a^* that maximises the global payoff function g:

$$a^* = argmax_a\ g(a)$$



(a) Example of acyclical coordination graph (b) Example of cyclical coordination graph

Examples of multi-agent systems represented as coordination graphs

Sequential Decision Making

In sequential decision making, the factors in the coordination graphs are represented as a joint Q-functions.

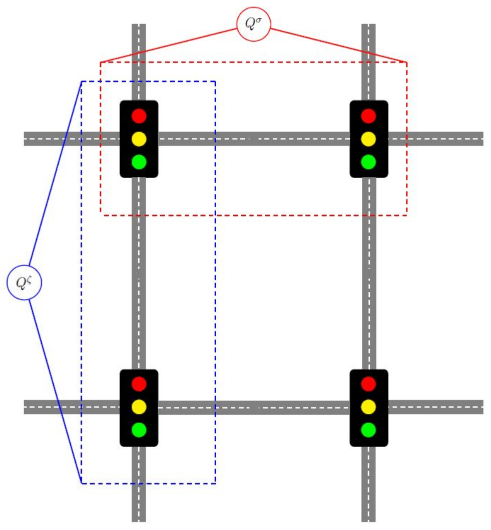
$$f_{ij}(a_i, a_j) = Q_{ij}(s, a_i, a_j; \theta_{ij})$$

Where i and j are neighbouring agents, a_i and a_j are their actions and s_i and s_j their states, s is the joint state over both agents and θ_{ij} is the weight matrix parameterising Q_{ij}.

The joint local function can be learnt by simply adding the individual Q-function, but this approach is not effective since optimal actions by the i is affected by actions taken by j. Better approach is using transfer planning, where joint local function is learnt separately by taking two agent into account and then using it as a source problem for larger subproblem. This idea can be depicted using the following figure:

Goal/Conclusion

The aim of the thesis is to scale the multi agent problem to large number of intersection, for example this can be scaled firstly to a small area of the city and ultimately extend it to an entire city. This would take rigorous study and training of different type of intersection in the city, and then ultimately find the Q-function that will optimise the traffic light for the entire city.



Transfer Planning for Traffic Light Control