Learning dispatching policies for the Single Intersection Scheduling problem

Jeroen van Riel

February 2024

1 Introduction

Recall our definition of the Single Interesection Scheduling Problem. Suppose we have a single intersection with K incoming lanes. Let $i \in \{0, \dots, K-1\}$ be an arbitrary lane and let n_i denote the (possibly infinite) number of total arriving platoons to this lane. We will refer to the j-th platoon of vehicles on lane i as *platoon* (i, j), and we let $Z_{ij} = (R_{ij}, P_{ij})$ be its full specification, where R_{ij} and P_{ij} denote the release date and platoon length, respectively. Note that we do not need to treat vehicles individually due to the Platoon Preservation Theorem. The *arrival process* of lane i is defined as the collection of random variables

$$Z_i = (Z_{ij})_{j=1}^{n_i}. \tag{1}$$

We saw that the Single Intersection Scheduling Problem corresponding to a particular sample

$$z = (z_i)_{i=0}^{K-1} \sim Z = (Z_i)_{i=0}^{K-1}, \tag{2}$$

can be concisely formulated as a Mixed-Integer Linear Program. Therefore, one direct solution approach is to apply standard software for solving this well-defined class of problems.

Recall that the objective of the Single Intersection Scheduling Problem was to minimize the total completion time

$$\sum_{i=0}^{K-1} \sum_{j=1}^{n_i} C_{ij} \tag{3}$$

over all vehicles and lanes. To make comparing objective values a bit easier, we will focus on the total delay, defined as

$$D_z(y) = \sum_{i=0}^{K-1} \sum_{j=1}^{n_i} y_{ij} - r_{ij}. \tag{4}$$

for some problem instance z and some feasible schedule y.

2 Markov Decision Process

An alternative approach is to cast the problem in terms of iteratively deciding which platoon crosses the intersection next, to which we will refer as *dispatching*. The resulting sequential decision problem can be formulated as a Markov Decision Process, which allows us to apply reinforcement learning techniques.

States. For each time step $t = 0, 1, 2, \dots$, let $x^{(t)}$ denote the lane that was last served and let $C^{(t)}$ denote the completion time of the last platoon that was dispatched. Furthermore, let $m^{(t)}$ be a vector where entry $m_i^{(t)}$ counts the number of dispatched platoons from lane i. The state at step t can then be written as

$$s^{(t)} = (x^{(t)}, C^{(t)}, m^{(t)}, z). \tag{5}$$

In particular, the initial state is given by

$$s^{(0)} = (0, 0, 0, z). \tag{6}$$

Actions. At every state transition, some platoon is dispatched. The action a is to decide whether to keep serving the current lane ($a = 0$), or to serve the next lane ($a = 1$).

Transitions. The value of $x^{(t)}$ is simply updated as

$$x^{(t+1)} = \begin{cases} x^{(t)} & \text{if } a^{(t)} = 0, \\ x^{(t)} + 1 \mod K & \text{if } a^{(t)} = 1. \end{cases} \tag{7}$$

The platoon that is dispatched in the transition from t to $t + 1$ is identified by the indices $(i^{(t)}, j^{(t)})$, defined as

$$i^{(t)} = x^{(t+1)}, \tag{8a}$$

$$j^{(t)} = m_{i^{(t)}}^{(t)} + 1. \tag{8b}$$

Therefore, the completion time $C^{(t+1)}$ of the dispatched platoon is given by

$$C^{(t+1)} = \max\left(C^{(t)} + sa^{(t)},\ r_{i^{(t)}j^{(t)}}\right) + p_{i^{(t)}j^{(t)}}, \tag{9}$$

where s is the switch-over time. The platoon counters are simply updated as

$$m_i^{(t+1)} = m_i^{(t)} + \mathbb{1}\{i = i^{(t)}\}. \tag{10}$$

The remaining part of the state does not change.

Rewards. Recall that we assumed for the Single Intersection Scheduling Problem that all processing times are identical $p_j = p = 1$. In the current MDP, we allow platoons of arbitrary *processing time*, so in order to compare results, we assume that p_{ij} is always a positive integer. Observe that the delay for each vehicle in the dispatched platoon is equal and given by

$$C^{(t)} - p_{i^{(t)}j^{(t)}} - r_{i^{(t)}j^{(t)}}, \tag{11}$$

so the total contribution of $(i^{(t)}, j^{(t)})$ to the total delay is given by

$$d^{(t)} = p_{i^{(t)}j^{(t)}}(C^{(t)} - p_{i^{(t)}j^{(t)}} - r_{i^{(t)}j^{(t)}}), \tag{12}$$

which is the negative reward (cost) of step t. Let $N = n_0 + \cdots + n_{K-1}$ be the total number of platoons. Assuming $N < \infty$, the total negative reward is precisely the total delay

$$D_z(y) = \sum_{t=1}^{N} d^{(t)} \tag{13}$$

for the resulting schedule y.

2.1 Partial observability

Note that the exact arrivals z are a constant part of the state and that all transitions are deterministic. The only randomness in the MDP stems from the distribution of the initial state, which is essentially the distribution of Z. This model fits the assumption that all future arrivals are known to the scheduler in advance. Alternatively, we could assume that the number of known future arrivals is limited. Let us first model this in terms of *observations*, thereby obtaining a Partially Observable Markov Decision Process (POMDP).

Assume that the scheduler does not know about all arrivals upfront, but sees a certain number h of next arrivals at every lane i, then the visible *horizon* of lane i at step t consists of the release dates and processing times

$$h_i^{(t)} = (z_{ij} : m_i^{(t)} \leq j \leq m_i^{(t)} + h). \tag{14}$$

Therefore, the observation at step t is deterministic and given by

$$h^{(t)} = (h_i^{(t)})_{i=0}^{K-1}. \tag{15}$$

Because the arrival process is independent of the actions that are taken by the scheduler, the POMDP can simply be reformulated as a MDP. Instead of sampling the full arrival process at the initial state, we can assume that some next platoon is sampled at every transition. In that case, the states should be written as

$$s^{(t)} = (x^{(t)}, C^{(t)}, m^{(t)}, z^{(t)}). \tag{16}$$

In the initial state, we have

$$z_i^{(0)} = (Z_{ij})_{j=1}^{h}, \tag{17}$$

for each lane i. For the lane $i = i^{(t)}$ that was served in the transition from step t, we have the following simple shift

$$(z_i^{(t+1)})_j = \begin{cases} (z_i^{(t)})_{j+1} & \text{if } 1 \leq j \leq h-1, \\ Z_{i,m_i^{(t)}+h+1} & \text{if } j = h, \end{cases} \tag{18}$$

and we have $z_i^{(t+1)} = z_i^{(t)}$ for all the other lanes. Equation (18) shows that only the last component of the updated horizon representation is a new random sample.

2.2 Discussion

Note that we have exploited the Platoon Preservation Theorem implicitly in the design of the MDP by modeling platoons as single units. Once we start considering a network of intersections, this theorem no longer holds, which means that optimal schedules could require some platoons to be *split*.

The current definition of the action space imposes a linear order of the served lanes, sometimes also called *round robin*. For $K > 2$, we may want to consider arbitrary orders. For example, we could redefine the actions such that $a^{(t)} \in \{0, \dots, K-1\}$ indicates the next lane to be served, so that we simply have $x^{(t+1)} = a^{(t)}$. Furthermore, the completion time update from equation 9 should be changed to

$$C^{(t+1)} = \max\left(C^{(t)} + s \cdot \mathbb{1}\{x^{(t)} \neq a^{(t)}\},\ r_{i^{(t)}j^{(t)}}\right) + p_{i^{(t)}j^{(t)}}. \tag{19}$$

We defined the length h of the horizon in terms of the number of next platoons that are visible. This makes the implementation simpler, because the horizon vector has fixed dimensions. Alternatively, we could want to model a fixed look-ahead time, which would require that the horizon $h_i^{(t)}$ of lane i depends on the current timestep. This would require some additional care in dealing with the resulting feature vectors, which can now differ in length across lanes and steps.

3 Method

3.1 Deep Q-Learning

Let the *total discounted return* from step t be defined as

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \tag{20}$$

where γ is the *discount factor*. The state-action value $q_\pi(s, a)$ is defined as the expected total discounted return by taking action a in state s and following π afterwards, so we have

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a], \tag{21}$$

for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$. These values are also refered to as *q-values*. It turns out that all optimal policies share the same q-values, so the optimal state-action value function can be defined as

$$q_*(s, a) = \max_\pi q_\pi(s, a). \tag{22}$$

It can be shown that

$$q_*(s,a) = \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a')\right], \tag{23}$$

which is known as the *Bellman equation*.

The main goal of Q-learning is to estimate the optimal q-values. This is done using the following *temporal difference update*

$$\hat{q}(s_t, a_t) \leftarrow \hat{q}(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_{a} \hat{q}(s_{t+1}, a) - \hat{q}(s_t, a_t)]. \tag{24}$$

This method is an *off-policy* method, because it can be shown that $\hat{q}$ will converge to q_* regardless of the policy π that is being followed, as long as all states are visited infinitely often.

The deep Q-learning implementation that we are using also employs an experience replay mechanism.

3.2 Observations

We apply deep Q-learning on the observations defined in (14). Note that the size of the vector $h_i^{(t)}$ becomes smaller than h whenever less than h platoons are waiting on lane i. In order to keep the dimensions of the feature vector fixed, we fill the remaining entries with zeros. Furthermore, we do not use $h^{(t)}$ as the state feature at step t. Instead, we apply a simple shift to obtain

$$\hat{h}_i^{(t)} = h^{(t)}_{x^{(t)}+i \bmod K}. \tag{25}$$

This makes sure that the i-th entry of the feature always corresponds to the i-th lane from the current lane.

4 Experiments

We define a couple of different problem instance classes by specifying the distribution of Z. For each problem class, we train a dispatching rule using deep Q-learning by sampling episodes according to the given distribution. Next, we compare the performance of the learned dispatching rules to the solutions found by solving the corresponding MILP formulations. More precisely, we estimate

$$\mathbb{E}_{z\sim Z}\left[D_z(y^*(z))\right], \tag{26}$$

where $y^*(z)$ denotes an exact solution for z. For some trained policy π, let $y_\pi(z)$ denote the schedule that results from dispatching the vehicles from z according to π, then we compare the performance of π, given by

$$\mathbb{E}_{z\sim Z}\left[D_z(y_\pi(z))\right], \tag{27}$$

to the optimal performance in (26).

For all the experiments, the distribution of Z is parameterized as follows. We assume the processing times are integers distributed as $P_{ij} \sim \text{Uni}[1, \theta_i]$. We define interarrival times $X_{ij} \sim \text{Exp}(\lambda_i)$, parameterized such that $\mathbb{E}[X_{ij}] = \lambda_i$. The release dates are given by

$$R_{ij} = \sum_{l=1}^{j-1} P_{il} + \sum_{l=1}^{j} X_{il}. \tag{28}$$

4.1 Effect of arrival rate

Assume we have $K = 2$ lanes, with fixed switch-over time $s = 2$ and a fixed number of arrivals $n_i = n = 30$ for each lane. We set $\theta_i = \theta = 3$ and equal arrival rates $\lambda_i = \lambda$ for each lane, for the following four scenarios $\lambda^{(1)} = 2, \lambda^{(2)} = 3, \lambda^{(3)} = 4, \lambda^{(4)} = 5$.

λ	exact ($g = 0.1$)	dqn ($h = 10$)
2	-705	-1012
3	-445	-375
4	-258	-251
5	-185	-201

Table 1: Episodic reward (rounded to integers) for each of the four scenarios for both methods. The results for the exact approach are averaged over 100 randomly generated instances. The results for the DQN method are obtained from a single run (consisting of multiple episodes).

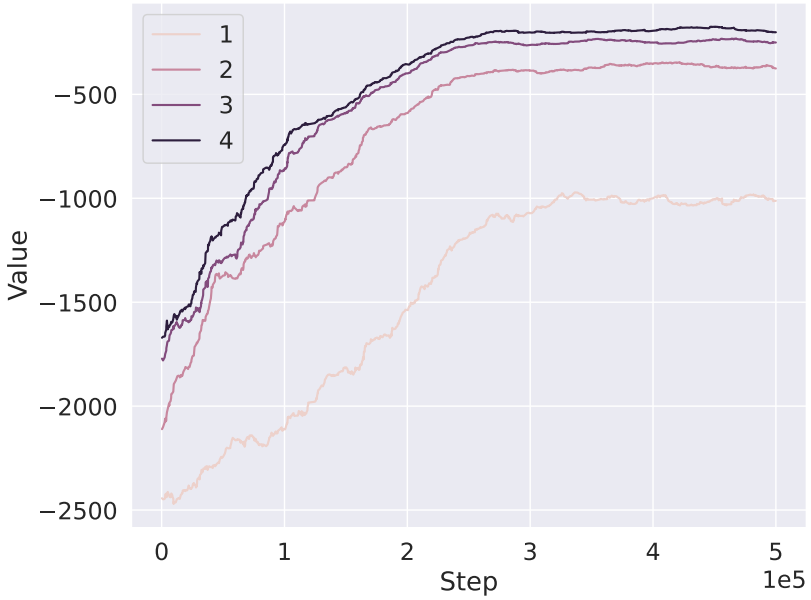

Figure 1: For the DQN, we use a fixed horizon of $h = 10$. The smoothed return obtained by a **single run** of the DQN algorithm is shown for each scenario.