DOI 10.3233/AIC-201580
IOS Press

Reinforcement learning vs. rule-based adaptive traffic signal control: A Fourier basis linear function approximation for traffic signal control

Theresa Ziemke [a,*], Lucas N. Alegre [b] and Ana L.C. Bazzan [b]

[a] *Transport Systems Planning and Transport Telematics, Technische Universität Berlin, Germany*
E-mail: tziemke@vsp.tu-berlin.de
[b] *Instituto da Informática, Universidade Federal do Rio Grande do Sul (UFRGS), Brazil*
E-mails: lnalegre@inf.ufrgs.br, bazzan@inf.ufrgs.br

Abstract. Reinforcement learning is an efficient, widely used machine learning technique that performs well when the state and action spaces have a reasonable size. This is rarely the case regarding control-related problems, as for instance controlling traffic signals. Here, the state space can be very large. In order to deal with the curse of dimensionality, a rough discretization of such space can be employed. However, this is effective just up to a certain point. A way to mitigate this is to use techniques that generalize the state space such as function approximation. In this paper, a linear function approximation is used. Specifically, SARSA(λ) with Fourier basis features is implemented to control traffic signals in the agent-based transport simulation MATSim. The results are compared not only to trivial controllers such as fixed-time, but also to state-of-the-art rule-based adaptive methods. It is concluded that SARSA(λ) with Fourier basis features is able to outperform such methods, especially in scenarios with varying traffic demands or unexpected events.

Keywords: Reinforcement learning, traffic signal control, linear function approximation, transport simulation

1. Introduction

Traffic signal control is a challenging real-world problem. Current solutions to this problem, such as adaptive systems like SCOOT [18], are often centralized or at least partially centralized if each controller is in charge of a portion of the urban network. Alternatives are manual interventions from traffic operators or the use of fixed-time signal plans. However, in the era of big data and advanced computing power, other paradigms are becoming more and more prominent. Among these, we find those derived from machine learning in general and reinforcement learning (RL) in particular. The reader is referred to some surveys in the area (see Section 2). In RL, traffic signal controllers located at intersections can be seen as autonomous agents that learn while interacting with the environment.

The use of RL is associated with challenging issues: the environment is dynamic (and thus agents must be highly adaptive), agents must react to changes in the environment at an individual level while also causing an unpredictable collective pattern, as they act in a coupled environment. Therefore, traffic signal control poses many challenges for standard techniques of multiagent learning.

To understand these challenges, let us first discuss the single agent case, where one agent performs an action once in a given state, and learns by getting a signal (reward) from the environment. To put it simply, RL techniques are based on estimates of values for state-action pairs (the so-called Q-values). These values may be represented as a table with one entry for each state-action pair. This works well in single agent problems and/or when the number of states and actions is small. However, in [28] Sutton and Barto discuss two drawbacks of this approach: First, a lot of memory is necessary to keep large tables when the number of

[*] Corresponding author. E-mail: tziemke@vsp.tu-berlin.de.

state-action pairs is huge, which tends to be the case in real-world applications. Second, a long exploration time is required to fill such tables accurately. Those authors then suggest that generalization techniques may help in addressing this so-called *curse of dimensionality*.

An efficient representation of the states is a key factor that may limit the use of the standard RL algorithms in problems that involve several agents. Moreover, in scenarios in which the states are represented as continuous values, estimation of the state value by means of tabular Q-values may not be feasible. To deal with this problem, in this paper a true online SARSA(λ) algorithm with Fourier Basis linear function approximation is used. As discussed ahead, this option is based on the fact that non-linear function approximation has several drawbacks.

The RL-based adaptive signal control algorithm was implemented in the open-source agent-based transport simulation MATSim [17]. In MATSim, it is possible to investigate the impact of the RL-based adaptive signal control algorithm and compare it to other fixed-time or adaptive signal control methods. For comparison, we run our approach against a rule-based adaptive signal control algorithm based on Lämmer and Helbing [22], which was implemented in MATSim in a previous study [20,30]. The results show that the RL-based approach is able to outperform these approaches in a single intersection scenario. This is especially notable, as these approaches were designed specifically for dealing with the control of signals, whereas the RL-based approach needs no domain knowledge. To the authors' best knowledge, virtually no other work in the literature (especially those stemming from the RL area) includes such kind of comparison. More often than not, comparison of RL approaches is made only to a fixed-time scheme.

The remaining of this paper is organized as follows. The next section discusses background and related work; this includes the rule-based adaptive signal control algorithm that is used as comparison in this study. The RL-based approach developed in this study is described in Section 3. Experiments and results are presented in Section 4, whereas Section 5 contains a discussion of the results and future work.

2. Background and related work

This section introduces some concepts on traffic signal control (Section 2.1) and gives more details about one method in particular, which is used as comparison (Section 2.2); then we discuss related work that is based on RL; the last subsection presents the simulation environment MATSim.

2.1. Traffic signal control

In contrast to fixed-time signals that cyclically repeat a given signal plan, traffic-responsive signals react to current traffic by adjusting signal states based on sensor-data (e.g., from upstream inducting loops or cameras). They can, therefore, react to changes in demand and reduce emissions and waiting times more efficiently.

A variety of traffic-responsive signal control algorithms have been developed. An overview is given, e.g., by Friedrich [8]. Different levels of adjustment are distinguished: *actuated* signals use a fixed-time base plan and adjust parameters like green split, cycle time or offset. *(Fully) adaptive* signals decide about the signal states on the fly. They can modify phase orders or even combine signals into different phases over time. With this, the flexibility of the signal optimization is augmented, which increases the possible improvement, but makes the optimization problem more complex. In order to reduce complexity and communication effort between sensors and a central computation unit (which controls signal states system-wide), *decentralized* (also called *self-controlled*) methods decide locally about signal states without complete knowledge of the system. Usually, every signalized intersection has its own processing unit that accounts for upstream (and sometimes downstream) sensor data of all approaches. A challenge of decentralized systems is to still ensure system-wide stability, especially when dealing with oversaturated conditions. A number of methods were developed that tackle these challenges.

Examples of traffic-responsive approaches from various generations and technological basis are: SCOOT [18] SCATS [23]; Prodyn [16]; OPAC [9]; UTOPIA [6]; TUC (*Traffic-responsive Urban Traffic Control*) [7]; and TUC combined with predictive control [5]. Some can be considered as rule-based as for example Lämmer and Helbing [22]), while others use techniques from RL and model signals as learning agents (see Section 2.3).

2.2. *Lämmer's rule-based adaptive traffic signal control algorithm*

The idea of the self-controlled signals proposed by Lämmer and Helbing [22] is to minimize waiting times and queue lengths at decentralized intersections while also granting stability through minimal service intervals. The algorithm combines two strategies. The **optimizing strategy** selects the signal phase i to be served next as the one with the highest priority index π_i (see Eq. (1)), which takes into account outflow rates and queue lengths of waiting and approaching vehicles that are registered by sensors. Given a prediction of the expected queue length $\hat{n}_i(t, \tau)$ at time $\tau > t$ and the maximum outflow rate $q_i^{\max}$ for phase i, one can derive the expected required green time $\hat{g}_i(t, \tau)$ for clearing the queue at time t using $\hat{g}_i(t, \tau) = \frac{\hat{n}_i(t,\tau)}{q_i^{\max}}$. With this, the priority index is calculated as follows:

$$\pi_i(t) = \begin{cases} \max_{\tau_i(t) \leqslant \tau \leqslant \tau_i^0} \frac{\hat{n}_i(t,\tau)}{\tau + \hat{g}_i(t,\tau)}, & \text{if } i = \sigma(t) \\ \frac{\hat{n}_i(t,\tau_i^0)}{\tau_{\sigma(t)}^{\text{pen}}(t) + \tau_i^0 + \hat{g}_i(t,\tau_i^0)}, & \text{if } i \neq \sigma(t). \end{cases} \quad (1)$$

Two cases are distinguished depending on whether the phase i is already active or not. In either case, the equation basically divides the number of vehicles by the time needed to clear the queue including the (remaining) intergreen time. The priority index can, therefore, be interpreted as a clearance efficiency rate. τ includes either the effect of remaining intergreen time for the selected phase (when it has not yet switched to green), or a lookahead beyond the end of the current queue. It is bounded from below by the remaining intergreen time $\tau_i(t)$, since that time, if larger than zero, will be incurred before traffic can flow, and from above by the full intergreen time τ_i^0, since beyond that it is possible to just switch back from some other state. For a non-active phase (i.e., $i \neq \sigma(t)$), the priority index is reduced by a canceling penalty $\tau_{\sigma(t)}^{\text{pen}}(t)$. This prevents the optimizing regime from frequently switching signal phases. The penalty can be interpreted as the average additional waiting time for vehicles at the previously served links that would occur upon cancellation. The priority index as it is defined in Eq. (1) assumes that each signal phase only serves one link – which is why phases and links are both denoted by i here. The algorithm was further extended to be able to deal with realistic traffic situations like lanes, phase combination with opposing traffic, minimum green times, and overload. Since these extensions make the equation less readable while the main method stays the same, the authors refer to Thunig et. al [30] for more details.

An enclosing **stabilizing strategy** ensures that each link is at least served once during a specified minimal service interval to prevent spillbacks. Links that have to be stabilized are added to a stabilization queue. If the queue is non-empty, the phase corresponding to the first element of the queue is switched to green for a guaranteed green time g_i^s depending on the average capacity utilization. If the stabilization queue is empty, the optimizing strategy takes over. Lämmer's control claims to provide intrinsic green waves and locally optimal service, which also results in system-wide optimal service.

An **assumption** of Lämmer's algorithm is the queue-representation of traffic flow: If a link i is served, vehicles can leave the link with a constant outflow rate $q_i^{\max}$, which is assumed to be known. Additionally, queues are assumed to be non-spatially, i.e., the algorithm does not account for vehicles spilling back to upstream lanes or links. Demand is supposed to be manageable on average with the desired cycle time T to ensure stability.

Two **sensors** are used to predict the number of waiting vehicles per link and time. One is positioned at the end of the link to detect waiting and outflowing vehicles; the second one is located further upstream to detect approaching vehicles. Assuming free flow conditions at link i, one can estimate the length of the queue $n_i(t)$ at time t and predict the expected queue length $\hat{n}_i(t, \tau)$ at a time $\tau > t$. While the estimation of queue lengths allows uncertainty, the mere presence of a queue is definite.

2.3. *Reinforcement learning*

In RL, an agent's goal is to learn an optimal control policy $\pi*$, which maps a given state to the best appropriate action by means of a value function. We can model RL as a Markov decision process (MDP) composed by a tuple (S, A, T, R), where S is a set of states; A is a set of actions; T is the transition function that models the probability of the system moving from a state $s \in S$ to a state $s' \in S$, upon performing action $a \in A$; and R is the reward function that yields a real number associated with performing an action $a \in A$ when one is in state $s \in S$. An experience tuple $\langle s, a, s', r \rangle$ denotes the fact that the agent was in state s, performed action a and ended up in s' with reward r. Let t denote the t^{th} step in the policy π. In an infinite horizon MDP, the cumulative reward in the future under policy π is defined by the Q-function, Eq. (2), where $\gamma \in [0, 1]$ is the discount factor for future rewards.

$$Q^{\pi}(s, a) = \mathbb{E}\left[\sum_{\tau=0}^{\infty} \gamma^{\tau} r_{t+\tau} | s_t = s, a_t = a, \pi\right] \tag{2}$$

Since the agent's objective is to maximize the cumulative reward, if it learned the optimal Q-values $Q^*(s, a)$ for all state-actions pairs, then the optimal control policy π^* is as follows:[1]

$$\pi^*(s) = \text{argmax}_a \, Q^*(s, a) \quad \forall s \in S, a \in A. \tag{3}$$

RL methods can be divided into two categories: *Model-based* methods assume that the transition function T and the reward function R are available, or instead try to learn them. *Model-free* methods, on the other hand, do not require that the agents have access to information about how the environment works.

There are many studies that use RL to improve traffic signal performance. For details, we refer the reader to some survey papers, which cover different aspects and perspectives: [4,24,37,38].

Using RL for traffic signal control is especially promising, as one does not need a lot of domain knowledge (as opposed to, e.g., rule-based approaches); rather, the controller learns a policy by itself. However, issues may arise with the aforementioned curse of dimensionality. In fact, depending on the specific formulation (e.g., how states and action spaces are defined), the search space can be very high. For instance, consider an intersection with four incoming approaches with three lanes per approach. If we define the state as the queue length on each lane discretized in 10 levels, it results in $10^{(4\times3)}$ distinct possible states. The reader is referred to [38] for several variants of such formulations.

In [24,26,27], RL is used by traffic signals in order to learn a policy that maps states (normally queues at junctions) to actions (normally keeping/changing the current split of green times among the lights of each phase). In [27] the approach is centralized (a single entity holds the MDP for all traffic signals); a central authority receives information about the length of the queues and elapsed time from various lanes to make a decision about timings at each signal. On the other hand, the approaches in [24] and [26] are decentralized. Each junction learns independently (normally using QL).

Since most of these works use QL, and thus approximate the Q-function as a table, they may fall prey to the curse of dimensionality. This arises when one deals with realistic scenarios, as, e.g., those beyond 2-phase intersections that are common in the literature.

In order to address this, a few works used function approximation. For instance, [1] uses tile coding in function approximation. However, the definition of states only considers queue length.

Recently, many studies have achieved impressive results using deep neural networks to approximate the Q-function (e.g., DQN [25,32,39]). However, linear function approximation has guaranteed convergence and error bounds, whereas non-linear function approximation is known to diverge in multiple cases [3,29]. Moreover, linear function approximation is less computation-intensive, as it relies on a significantly fewer number of parameters. Thus, if the Q-function can be linearly approximated with sufficient precision, linear function approximation methods are preferable.

2.4. *Transport simulation*

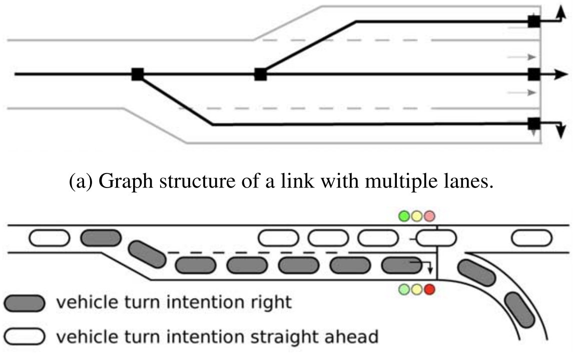
As deployment, operations, and maintenance costs of traffic-responsive signals in general are high, transport simulation tools provide a perfect environment to systematically test and evaluate new signal control methods before applying them in the field.

The agent-based transport simulation MATSim [17], which is used in this study, is especially suitable in this regard, as it is able to run large-scale real-world simulations in reasonable time as. Simulations can be build based on open data (see, e.g., the open Berlin scenario [40]) such that the impact of new signal control approaches can be easily analyzed for arbitrary scenarios[2] and compared to other control methods. Because of its agent-based structure, agent-specific waiting times and varying queue lengths over time at traffic lights can be directly analyzed and compared.

In MATSim traffic is modeled by agents (i.e., persons) that follow a daily plan of activities and trips. Traffic flow is modeled mesoscopically by spatial first-in-first-out (FIFO) queues. Vehicles at the head of a queue can leave a link when the following criteria are fulfilled: (1) The link's free-flow travel time has

[1] For converge guarantees, in the case of QL, please see [35].

[2] An example on how to start a MATSim simulation using the RL signal control presented in this paper can be found at http://matsim.org/javadoc → signals → RunSarsaLambdaSignalsExample.


vehicle turn intention right
vehicle turn intention straight ahead

(a) Graph structure of a link with multiple lanes.

(b) Multiple queues; spillback is captured correctly.

Fig. 1. Links with multiple lanes in MATSim. Each lane is represented by its own FIFO queue. Traffic signal control for different turning moves is captured. Vehicles on different lanes can pass each other, unless the queue spills over. Source: [12].

passed, (2) the flow capacity of the link is not exceeded in the given time step, and (3) there is enough space on the next link. Despite this simplistic modeling approach, congestion, as well as spillback, can be modeled.

The traffic signal control module was developed by Grether as an extension to MATSim [13]. If a signal exists on a link, leaving the link is not possible while it shows red. First studies focused on fixed-time signals, but also approaches for traffic-responsive signal control have been implemented [11,20,30]. Kühnel et al. [20] and Thunig et al. [30] present the implementation and application of the rule-based signals from Section 2.2 in MATSim. This implementation is also used in the present study as comparison for the RL signal control.

Separated waiting queues for different turning directions at intersections can be modeled in MATSim by lanes, which are a substructure of links (see Fig. 1). They are especially useful to model protected left turns at signalized intersections. Also, the spatial interaction of different waiting queues on a link can be captured correctly by lanes, as Fig. 1(b) depicts. Each lane can be signalized separately. Signals and lanes in MATSim are more extensively described by Grether and Thunig [12].

Events of vehicles entering or leaving links and lanes are thrown on a second-by-second time resolution in the simulation. Sensors on links or lanes that detect single vehicles can be easily modeled by listening to these events. As in reality, the maximum forecast period of such sensors is limited – vehicles can only be detected when they have entered the link. If a link is short, forecasts might not be accurate. In the simulation, responsive signals use these sensor data to react dynamically to approaching vehicles. For every signalized intersection, the control unit is called every second to decide about current signal states. With that, also RL-based signal control approaches can be easily installed into the simulation framework.

In general, MATSim can model user reaction as route, mode or departure time changes. But for this paper, only the traffic flow simulation of MATSim is used. Readers interested in the evolutionary part of MATSim – i.e., how agents adapt their plans and how long-term effects can be analyzed – are referred to [17].

3. Methods

In this section, we first discuss the method used for function approximation, then give details about the formulation of state and action space, as well as rewards, for the specific domain of signal control.

3.1. *Fourier basis linear function approximation with the true online* SARSA(λ)

The proposed RL traffic signal controller implements the true online SARSA(λ) algorithm [34], a modification of the traditional SARSA(λ) that was demonstrated to have better theoretical properties and outperform the original method [33]. The algorithm is called *true online* because it matches its update target (an estimate of the expected cumulative sum of rewards) exactly, in contrast to classical online SARSA(λ), which only approximates it. As detailed later, we use two kinds of features, thus impacting the space state. In order to deal with high dimensional state spaces, the Q-function was linearly approximated using the Fourier basis scheme [19].

When linear approximation is used, the Q-values $Q(s, a)$ for each discrete action a are approximated as a weighted sum of a set of m basis functions $\phi_1, \ldots, \phi_m$, as in Eq. (4), where $\boldsymbol{\theta}$ is the learned vector of weights. We denote $\boldsymbol{\phi}(s, a)$ (and the shorthand $\boldsymbol{\phi}$) as the vector containing the values of all basis functions ϕ_i.

$$Q(s, a) = \boldsymbol{\theta}^{\mathsf{T}} \boldsymbol{\phi}(s, a) = \sum_{i=1}^{m} \theta_i \phi_i(s, a) \qquad (4)$$

The Fourier series is one of the most commonly used continuous function approximation methods, presenting solid theoretical foundations. In [19], it was empirically shown that Fourier basis outperforms other commonly used approximations methods such as polynomial and radial basis functions in continuous RL domains.

When applying Fourier series to the RL setting, it is possible to drop the sin terms of the series.[3] Then, for a n-th order Fourier approximation, each basis function ϕ_i is defined as in Eq. (5), where $\mathbf{c}^i = [c_1, \dots, c_k]$ is a vector that attaches an integer coefficient $c_{1 \leqslant j \leqslant k} \in [0, \dots, n]$ to each feature in $\mathbf{s}$, and k is the dimension of the state space. The coefficient c_j in each coefficient vector $\mathbf{c}^i$ determines the basis function's frequency along the j-th dimension of the state space. Note that the basis functions of all actions except the current action a_t are zeroed, as only the weights corresponding to the selected action must be updated.

$$\phi_i(s, a) = \begin{cases} \cos(\pi \mathbf{c}^i \cdot \mathbf{s}), & \text{if } a = a_t \\ 0, & \text{if } a \neq a_t \end{cases} \tag{5}$$

The set of basis functions $\phi_1, \dots, \phi_m$ can be obtained by systematically enumerating all possible coefficient vectors. Furthermore, as we increase the order n of the approximation, more frequencies are used. However, as the number of Fourier basis functions grows exponentially on the state space dimension, the user can impose constraints on the coefficient vectors to reduce the number of basis in scenarios with large state spaces. A simple approach is to limit each coefficient vector to have a maximum number of non-zero coefficients. For instance, restricting the coefficient vectors to at most two non-zero coefficients allows us to capture the relation between pairs of features in the state space.

After the execution of action a_t, the weights θ are updated via gradient descent, following the true online SARSA(λ) with linear function approximation update rule, as in Eq. (6), where $\delta = r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$ is the temporal difference error and Q_{old} is a scalar temporary variable initialized with zero and set to $Q_{\text{old}} \leftarrow Q(s_{t+1}, a_{t+1})$ after every step.

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha(\delta + Q - Q_{\text{old}})\mathbf{e} - \alpha(Q - Q_{\text{old}})\phi \tag{6}$$

This update rule objective is to minimize the temporal difference error δ, which denotes the error in the current estimates of the Q-values. We refer the reader to [33] for details on its derivation.

The eligibility traces vector $\mathbf{e}$ in Eq. (6) – which is used to address the *credit assignment* problem – is updated as in Eq. (7). Each weight update also takes into account previously visited states, which are credited accordingly to the values accumulated on the vector $\mathbf{e}$. The parameter $\lambda \in [0, 1]$ controls the decay of the eligibility traces at each time step. The higher the value of λ, the higher is the influence of past updates in the update of the current step.

$$\mathbf{e} \leftarrow \gamma\lambda\mathbf{e} + \boldsymbol{\phi} - \alpha\gamma\lambda(\mathbf{e}^{\mathsf{T}}\phi)\phi \tag{7}$$

Given the base learning rate α, each weight θ_i is updated with the scaled learning rate $\alpha_i = \alpha / \|\mathbf{c}^i\|_2$, as proposed in [19]. Both the weights and eligibility traces vectors are initialized with zeros.

In order to address the exploration–exploitation dilemma, the ε-*greedy* exploration strategy is used to choose actions: the action with the highest Q-value is selected with a probability of $1 - \varepsilon$ and a random action is selected with probability ε.

Next, we give the formulations that are specific to the domain of signal control.

3.2. State space

In RL problems, the definition of state space strongly influences the agents' behavior and performance. In traffic signal control, for instance, information related to the level of congestion in the approaching lanes is fundamental in order to appropriately choose the next active signal phase.

In the present setting, the agent observes a vector $\mathbf{s_t} \in \mathbb{R}^k$ at each time step t. This vector partially represents the true state of the controlled intersection and is defined as in Eq. (8), where E is the set of all links of the intersection and L is the set of all approaching lanes, $\rho_i \in \{0, 1\}$ is a binary feature active when i is the current selected signal phase, $\tau \in [0, 1]$ is the elapsed time of the current signal phase divided by the maximal green time $g_{\max}$, the density $\Delta_e \in [0, 1]$ is defined as the number of vehicles on link $e \in E$ divided by it's storage capacity and $q_l \in [0, 1]$ is defined as the number of queued vehicles on lane $l \in L$ divided by the storage capacity of the lane.

$$\mathbf{s}_t = [\rho_1, \dots, \rho_{|\sigma|}, \tau, q_1, \dots, q_{|L|}, \Delta_1, \dots, \Delta_{|E|}] \tag{8}$$

This state definition is inspired by [10], where authors achieved similar performance levels, even when using

[3] For detailed explanation, please see [19].

more complex state definitions (e.g., including positions of each vehicle in the approaching lanes).

As common in the literature, the proposed RL signal control is only called every three seconds. This means, that one time step for the traffic signal agent corresponds to three seconds of simulation. This reduces the complexity and the size of the state space, without significantly reducing the performance.

3.3. Action space

At each time step t (every three seconds), the traffic signal controller chooses a discrete action $a_t \in A$. In our setting, the number of actions is equal to the number of possible signal phases, therefore, $|A| = |\sigma|$. There are two restrictions in the action selection: the agent can change the current active signal phase only if the elapsed time is greater or equal than the minimal green time $g_{\min}$ and keep it only if the elapsed time is less than the maximal green time $g_{\max}$. These restrictions ensure the feasibility of the signal controller for real-world applications.

3.4. Reward

After taking action a_t, the traffic signal controller receives a scalar reward $r_t \in \mathbb{R}$. As in [10] the reward is defined as the change in cumulative delay, as given in Eq. (9), where D_{a_t} and $D_{a_{t+1}}$ represent the cumulative delay at the intersection before and after executing the action a_t.

$$r_t = D_{a_t} - D_{a_{t+1}} \tag{9}$$

In its turn, the cumulative vehicle delay D, for any time t, is computed as in Eq. (10), where V_t is the set of vehicles on incoming approaches and d_t^v is the delay of vehicle v at time t.

$$D_t = \sum_{v \in V_t} d_t^v \tag{10}$$

4. Experiments and results

4.1. Scenario

This study focuses on a single intersection scenario with four different set-ups. The set-ups vary in demand and/or number of lanes that are usable. The RL control is compared to a fixed-time signal control and rule-based traffic-responsive signal control based on [22] (as introduced in Section 2.2).

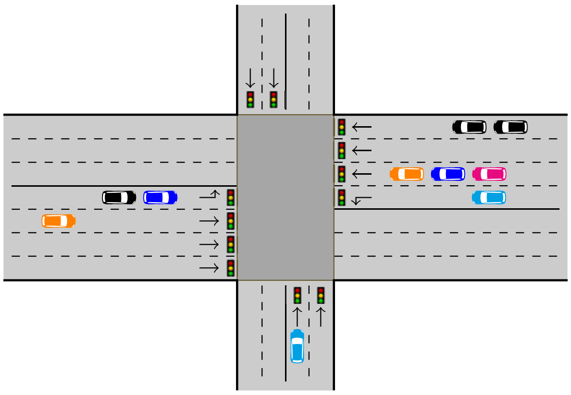


Fig. 2. Single intersection scenario.

Nevertheless, the proposed RL method for traffic signal control is also applicable to real-world scenarios. To do so, every signalized intersection can be modeled as an individual learning agent, only working with local sensor information. This way, green waves are not specifically tackled or pre-defined. We note however that, they may be considered if a different reward function is defined, which is designed to reward offsets that are inline with the emergence of a green wave.

Further, to address more complex scenarios, a setting that considers a network of signals is being investigated, were we show that the RL signal control proposed here is able to keep up with – and in some situations is even able to outperform – Lämmer's algorithm in a real-world scenario with multiple signalized intersections (see [2]).

4.1.1. Traffic signals

The single intersection featured here (see Fig. 2) has four incoming approaches. In the horizontal direction, there is a dedicate left turning lane in each traffic approach, as well as three lanes for straight traffic. In the vertical direction, there are two lanes for straight traffic.

Traffic signals are grouped into three non-conflicting signal phases: Straight traffic in horizontal direction; left turning traffic in horizontal direction; vertical direction. While switching between two signal phases, there is an all red period of one second. The minimum green time for a signal phase is five seconds.

The fixed-time control that is used for comparison purposes is optimized by Webster's method [36]. It has a cycle time of 40 seconds and distributes green times according to average flow rates. The traffic-responsive signal approaches do not have a fixed cycle time: For

Lämmer's control algorithm, a desired and a maximal cycle time can be defined (for this scenario 40 and 60 seconds are used, respectively). For the RL control a maximal green time of 30 seconds per signal phase is used. As mentioned in Section 3.1, the RL control is only called every three seconds to decide about new signal states.

All these parameter settings (such as all red time, minimum green time, update time etc.) can of course be adjusted when applying the RL signal control to other scenarios.

4.1.2. Demand

Four different demand set-ups are modeled. In all set-ups, arrival rates are stochastic: vehicles are inserted as platoons, with a platoon size that is exponentially distributed around an expected value of five. Also the time gap between vehicle platoons is exponentially distributed: its expected value is the platoon size divided by the average flow value. To average out the fluctuations in the results depending on the specific platoon structure of approaching vehicles, each set-up was simulated with 20 different random seeds.

Constant demand. In a first set-up, there is traffic going straight in the horizontal direction, with 1800 vehicles approaching on average per hour, in each of the two approaches. In the vertical direction, there are 600 vehicles on average per hour from each side – all going straight. Additionally, there are 180 vehicles on average per hour from both sides in horizontal direction that want to turn left at the intersection. A period of 86,400 seconds (i.e., one day) is simulated.

Peaks with doubled demand. In a second set-up, the demand is doubled during five time periods over the day of 2,000 seconds length each, in order to analyze the effect of fluctuating demand on the performance of the RL controller. To be more precise, in the time intervals [0, 2,000), [20,000, 22,000), [40,000, 42,000), [60,000, 62,000), and [80,000, 82,000) the average flow rates in horizontal direction are 3600 vehicles per hour going straight and 360 vehicles per hour going left per approach, whereas in vertical direction the average flow rate per approach is 1200 vehicles per hour. During the rest of the simulation, the average flow rates are the same as for the first scenario set-up.

With this demand set-up, it can be analyzed how the control algorithms behave with short periods of overload. Because these periods periodically repeat, the RL control is able to learn from peak to peak while the other controllers behave similarly in all peaks.

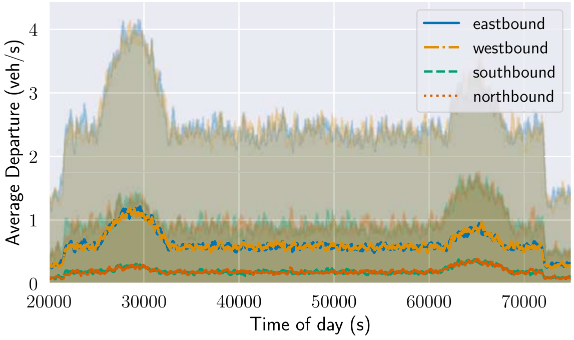


Fig. 3. Average number of departures per second per direction for the third demand set-up with asymmetric periodic demand.

Asymmetric periodic demand. In this third set-up, an artificial morning and evening peak are simulated around a daily demand level that corresponds to the first set-up. To model the peaks, a sinus curve modifies the average flow values. In the morning peak, this sinus curve has its maximum at 8 am with twice the daily demand level for the horizontal direction and 1.5 times the daily demand for the vertical direction. In the evening, this factors are swapped (1.5 for horizontal direction; 2 for vertical direction) with a maximum at 6 pm. During the day (between 10 am and 4 pm), a constant demand level similar to the first demand set-up is used; during the night one half of this constant demand is used. Figure 3 shows the number of departures per second per direction resulting from this set-up. The x-axis is trimmed to the interesting part of the day. The shadowed area depicts the standard deviation. The lines are smoothed with a moving average window of 300 seconds (i.e., 5 minutes) for better clarity.

Having this kind of asymmetric periodic demand makes the situation more difficult for the RL control because a wider spectrum of the state space (i.e., of different vehicle pattern) has to be observed and explored. On the other hand, for Lämmer's algorithm, the evening peak in this set-up is especially challenging, as the main traffic approaches from the secondary road. This is due to the way the algorithm prioritizes between approaches with different flow capacities.

Constant demand with lane closure. For the fourth set-up, a constant demand is used, with 1100 vehicles on average per hour in each of the two approaches of the horizontal direction and, in each case, additionally 110 vehicles on average per hour that want to turn left. Vertical traffic corresponds to the first demand set-up (600 vehicles per hour).

Between 6 am and 6 pm a lane closure (e.g., due to a road work) is simulated eastbound in horizontal direction which results in a reduction of flow capacity to one third (two lanes are closed). This interesting to look at because Lämmer's adaptive algorithm is not capable of dealing with such a spontaneous capacity change and still assumes the old flow capacity values, while RL is able to learn from the new situation without any domain knowledge.

4.2. *Results*

The proposed method of the true online SARSA(λ) with Fourier basis linear function approximation for signal control is applied to the single intersection scenario presented in the previous section and compared to RL signal control methods with other configurations (in Section 4.2.1, where our method is compared to a tabular variant), as well as to a fixed-time and a rule-based adaptive signal control approach (in Section 4.2.2).

Due to the stochastic arrival rates, results presented here are averaged over 20 runs with different random seeds, whereby the random seed influences the platoon structure of approaching vehicles (the average flow rate stays the same).

The shadowed area in the plots depicts the standard deviation regarding average delay or total queue length, accordingly. The lines are smoothed with a moving average window of 300 seconds (i.e., 5 minutes) for better clarity.

4.2.1. *Comparison with other RL-based signal control methods*

Here we compare the proposed method with the traditional tabular SARSA(λ) [28], using the first set-up of the scenario presented in Section 4.1. We also discuss optimal settings regarding the order of the Fourier basis approximation, state and reward.

Tabular vs. linear SARSA(λ). In order to transform the continuous state space defined in Section 3.1 to a discrete state space for the tabular SARSA(λ), the queue q and density Δ attributes were discretized in equally distributed bins/intervals. The binary features ρ_i for each phase are already discrete and the feature τ has a finite number of possible values as the elapsed time increases in steps of five seconds; therefore, they did not need to be discretized.

In order to allow a fair comparison, the same discount factor, value of λ and exploration rate were used for both methods. The discount factor was set to $\gamma = 0.95$, $\lambda = 0.1$ and the exploration rate was set to $\varepsilon = 0.01$ (this latter means that the agent is mostly taking the action with the highest Q-value, but still exploring with a fixed low chance). For the tabular SARSA(λ), a learning rate of $\alpha = 0.1$ was used, while for true online SARSA(λ) with linear function approximation, $\alpha = 10^{-6}$ was used. These values are common in the literature and produced the best results for each method after extensive experimentation with different values.

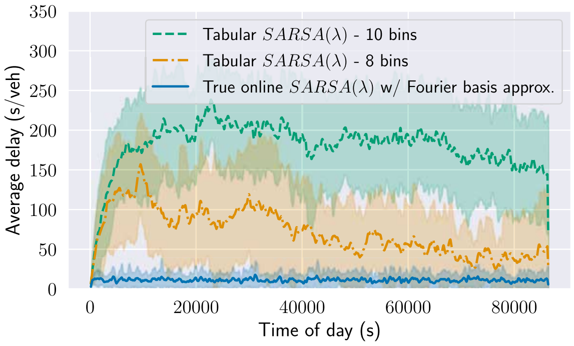


Fig. 4. Average delay for tabular and linear function approximation RL implementations.

As the state space in this case is large, and the number of Fourier basis functions grows exponentially on the number of dimensions of the state space, we placed constraints on the coefficient vectors $\mathbf{c}^i$. In this setting, adding coefficients with more than two non-zero elements did not improve the results. Thus, we further limited each coefficient vector $\mathbf{c}^i$ to have at most two non-zero elements.

In Fig. 4 the average delay per vehicle at each second of the simulation is depicted for true online SARSA(λ) with Fourier basis linear function approximation and for tabular SARSA(λ) with 8 vs. 10 discretization bins of the q and Δ features.

With 10 bins, the learning is very slow, as the number of discretization bins exponentially increases the size of the state space. Reducing the number of bins to 8 significantly speeds up learning and reduces the delay. However, by reducing the number of bins, different states (in which different actions are optimal) are perceived as the same, thus leading to a sub-optimal performance in the long run.

The usage of function approximation not only avoids the curse of dimensionality, but introduces generalization, i.e., when updating the Q-function after taking an action in a given state, similar states are also affected and have their Q-values changed. With that, the true online SARSA(λ) with Fourier basis linear function approximation results in a much faster learning curve and overall lower delay values.

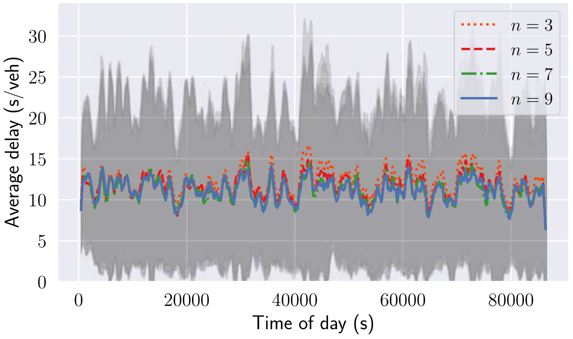


Fig. 5. Impact of different values for the order n of the Fourier basis approximation.

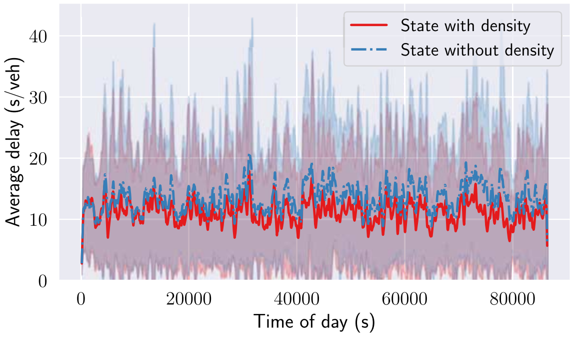


Fig. 6. Impact of state definition.

Order of the Fourier basis approximation. Figure 5 shows the impact that the value of the Fourier approximation order n has on the agent's performance. As expected, the higher the value of n, the more accurate is the approximation of the Q-function. Changing the order from $n = 3$ to $n = 9$ results in a notable reduction on average delay; however, when n is sufficiently high ($n = 7$ and $n = 9$), there is no further improvement. For this reason, the Fourier approximation order is fixed to $n = 7$ for all following experiments.

State definition. Although the q (flow) features provide the traffic signal control agent with queue information on each lane, the Δ (density) features are also important, as they inform how many vehicles (that may be queued in the following seconds) there are on each link. Figure 6 shows that, by removing the Δ features from the state definition, the average delay increases. This difference might be even higher in scenarios with very high demand, where a high number of vehicles are moving and approaching the queues.

Reward definition. The definition of the reward function has a high impact on the performance of the RL-based controller [15]: In Fig. 7 the reward function defined in Section 3.1 is compared to another reward function found in the literature [24], defined as the change in total queue length between successive actions. The traffic signal controller using change in cumulative delay as reward not only converges to better performance, but produces a learning curve that decreases orders of magnitude faster. This result shows that the choice of which reward function to use is one of the most critical implementation decisions when designing a reinforcement learning controller.

4.2.2. *Comparison with fixed-time and rule-based signals*

In this section, the true online SARSA(λ) with Fourier basis linear function approximation is compared to fixed-time and rule-based adaptive signals in all four set-ups of the single intersection scenario.

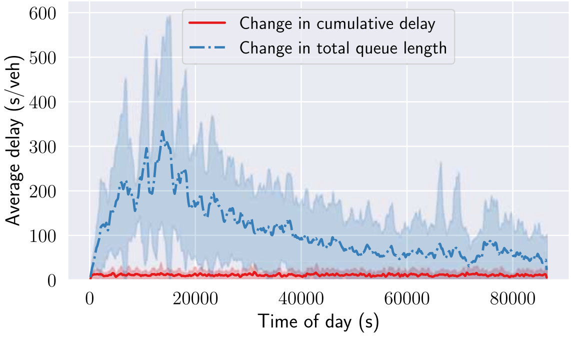
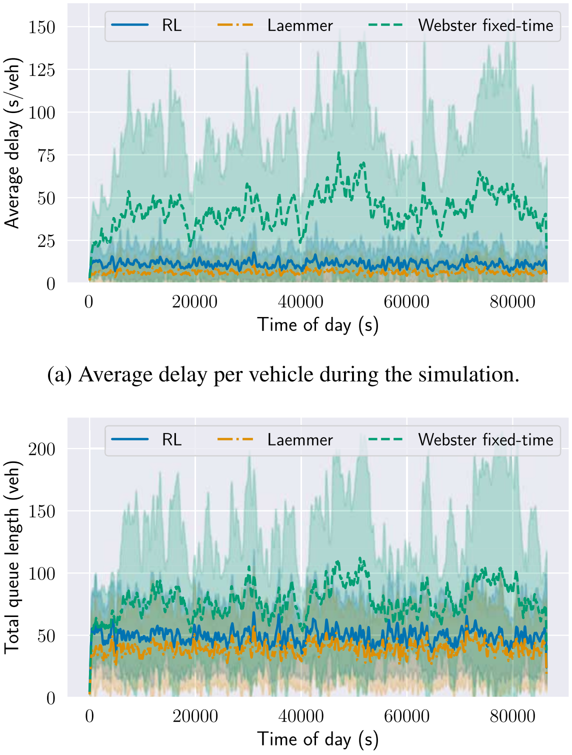


Fig. 7. Impact of reward definition.

First set-up – constant demand. Figure 8 shows the performance regarding average delay and total queue length for the first set-up (constant average flow rates). It can be seen that for this, somewhat homogeneous setup, both the RL-based and Lämmer approaches perform much better than the Webster fixed-time control in terms of average delay and queue length. Also, they produce less variation in these measures, demonstrating robustness against traffic fluctuations. Note that for constant average flow rates, the fixed-time control used here (optimized by Webster's method) is already quite good. RL is able to outperform the fixed scheme because it seems to be more stable regarding platoon variations. This can be seen in both plots in Fig. 8, with the standard deviation (shown as the shadowed area in the plots) being lower for the RL-based control.

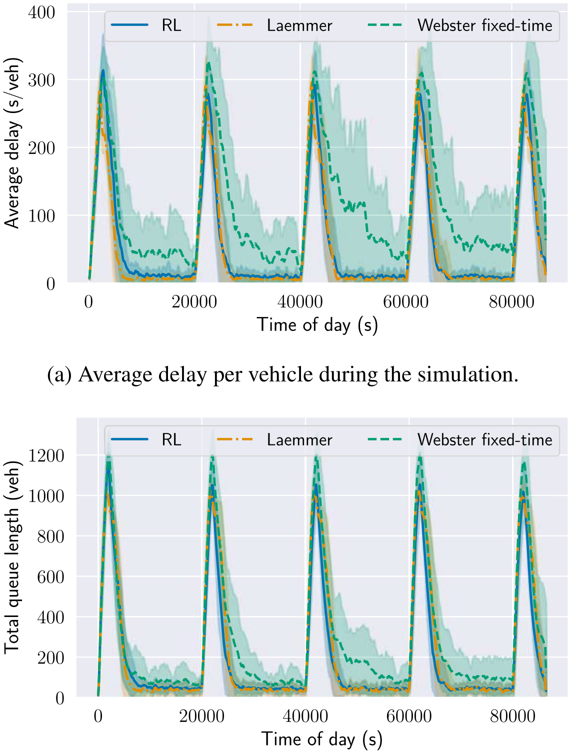
Second set-up – peaks with doubled demand. Figure 9 depicts how the different signal controllers are able to handle short phases of overload. Flow rates are doubled during five time periods over the day (see description in Section 4.1.2). For this, less homogeneous



(a) Average delay per vehicle during the simulation.

(b) Total queue length during the simulation.

Fig. 8. Single intersection scenario with constant average flow rates (first set up of Section 4.1.2).



(a) Average delay per vehicle during the simulation.

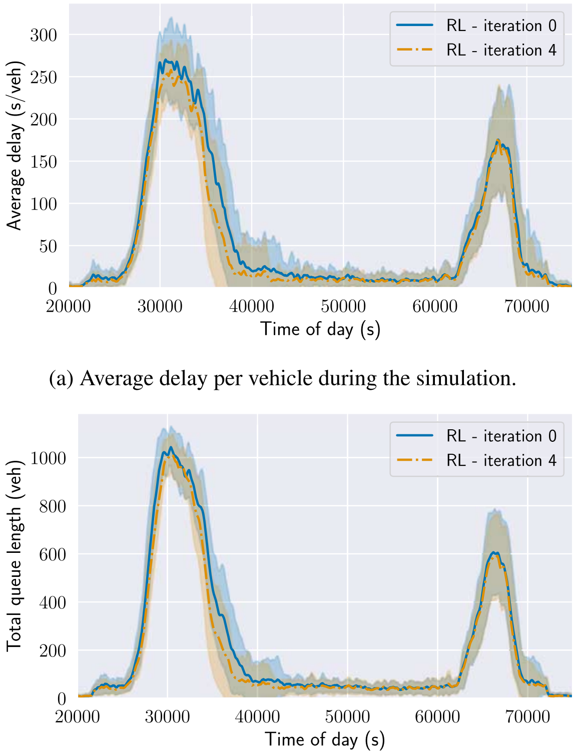
(b) Total queue length during the simulation.

Fig. 9. Single intersection scenario with periodically repeating time periods where the average flow rates are doubled (second set up of Section 4.1.2).

demand, both adaptive approaches clearly outperform the fixed-time control which is even not able to resolve the queues of one peak before the next peak begins. The RL controller improves its performance from the second peak onwards, as in the first peak it was experiencing an overload situation for the first time. In this more difficult set-up, the difference in performance between RL and Lämmer becomes less visible, with both presenting the same length of queues when there is low demand. Interestingly, RL decreases the queue lengths faster than Lämmer after the peaks, which indicates that RL better adapts to changes in flow pattern.

Third set-up – asymmetric periodic demand. This set-up presents the effects of more heterogeneous demand, where in the morning more traffic is approaching on horizontal direction while in the evening more traffic is approaching on the minor vertical road. With this, a wider spectrum of the state space needs to be explored by the RL-based controller because a lot of different vehicle patterns occur. This is why RL is able to improve further when it is run for multiple iterations (i.e. days) in the simulation – in contrast to the first demand set-up. A comparison of average delays and total queue length between the first and the fifths iteration is given in Fig. 10. The x-axis is trimmed to only show the relevant part of the day. Especially in the crowded morning peak, learning over the days helps to narrow and flatten the curves.

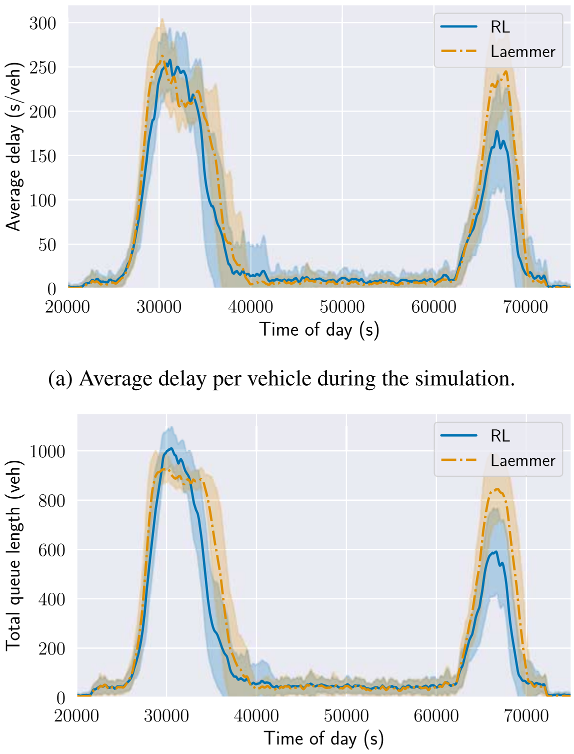
Compared to Lämmer's rule-based control, it can be seen that RL behaves very well in the evening peak, when the main traffic is approaching on the minor road, see Fig. 11. This is probably due to the priority calculation of Lämmer's algorithm, where approaches with lower flow capacity values result in lower priorities for the same demand pressure. Because also a lot of traffic is approaching on the major road with its high flow capacity, the minor road does not get the main priority. During the morning peak Lämmer and RL behave similarly, with Lämmer resulting in lower maximal queue length, but RL resolving the peak faster. As in the first demand set-up, Lämmer is slightly better with low constant demand values during the day.



(a) Average delay per vehicle during the simulation.

(b) Total queue length during the simulation.

Fig. 10. First vs. fourth iteration (i.e., days) of RL in the single intersection scenario with asymmetric morning and evening peak (third set up of Section 4.1.2).



(a) Average delay per vehicle during the simulation.

(b) Total queue length during the simulation.

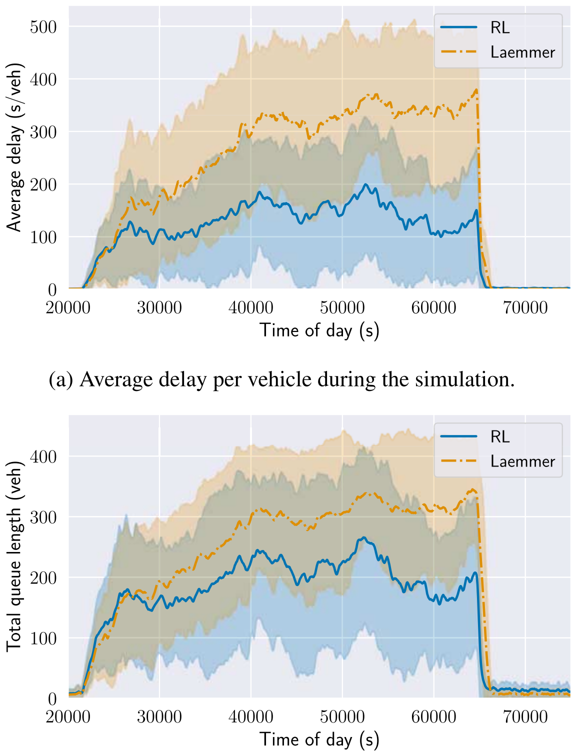
Fig. 11. Single intersection scenario with asymmetric morning and evening peak (third set up of Section 4.1.2).

Fourth set-up – constant demand with lane closure. Recall that, contrarily to rule-based approaches, the RL-based control does not require domain knowledge. With this, it has advantages when unexpected events occur that change the underlying situation (e.g. a change in flow capacities, storage capacities, free speed etc.). To verify this, the fourth set-up simulates a lane closure event, where two of the three lanes in horizontal direction eastbound are closed for some time (see description in Section 4.1.2). This results in a flow capacity drop by two thirds. As Lämmer's rule-based control still calculates priorities based on the original flow capacity values, it results in quite high delays and queue length, as seen in Fig. 12. To better see the difference between RL and Lämmer, total delay and queue length for the fixed-time control are not shown in that figure, as they are even higher. The x-axis is again trimmed to the relevant part of the day.

Interestingly, RL is worse than Lämmer at the beginning of the lane closure, where it is still learning to handle this new situation. However, it quickly overtakes Lämmer and stays more or less stable, while Lämmer's delay and queue length values increase further. When the lane closure ends, queue lengths and delays drop faster with RL, whereas Lämmer's control quickly takes over (similar to the first demand set-up) as it's domain knowledge is again beneficial.

5. Conclusion and future work

As discussed in the first two sections of the present paper, traffic signal control is a challenging domain for RL techniques, being the subject of an increasing body of research. In the present paper, it was shown that specific techniques from RL can help to improve the performance of traffic signal control, and even outperform state-of-the-art rule-based adaptive signal control methods, especially in scenarios with varying traffic demands or unexpected events. It was argued that tabular RL methods may not be feasible due to the curse of dimensionality. When it is possible to employ them, it is often the case that they need long learning times



(a) Average delay per vehicle during the simulation.

(b) Total queue length during the simulation.

Fig. 12. Single intersection scenario with constant demand where two lanes are closed eastbound on horizontal direction between 6 am and 6 pm (fourth set up of Section 4.1.2).

before convergence in the case of realistic intersections with more than two signal phases and when a more complex definition of state is used. Recall that the results presented here show that including more features (i.e., not only queue but also density) played a significant role in the performance.

To address the curse of dimensionality, we used Fourier basis linear function approximation alongside the true online SARSA(λ) algorithm, which to the authors' best knowledge was not used for traffic signal control before. This method was implemented in MATSim and compared to optimal fixed-time and rule-based adaptive signal control in a single intersection scenario, in which the demand was varied. It could be seen that our approach outperforms the fixed-time controller and is competitive with the rule-based adaptive controller in terms of average delay and queue length. Despite its slightly lower performance in the homogeneous demand set-up, the RL control is able to handle overload situations and quickly reduces the queues afterwards. Contrary to rule-based approaches, the RL control does not require domain knowledge which is why it clearly outperforms the rule-based approach when unexpected events happen that change underlying network properties (e.g. lane closures). Note, that Lämmer's rule-based approach is, similarly to our approach, a local signal control that still ensures system-wide stability and performs well in real-world applications [21]. Finally, we remark that a comparison with approaches other than fixed-time is rarely seen in the RL literature and is, therefore, a key feature of this work.

As a next step, the signal control based on true online SARSA(λ) with Fourier basis linear function approximation has already been applied to real-world scenarios using MATSim, and compared to the signal control approaches employed here [2]. The experiments substantiate the performance of the RL control, which emphasize its advantage in scenarios that are more challenging. In all real-world experiments, the RL control was able to keep up with the rule-based control and even outperformed it in some situations. Because we model every intersection as an independent learning agent, state and action spaces are still computationally manageable; the computation can even be parallelized.

As future avenues for research, we envision the following. First, it remains to be investigated whether the RL signal control can be further improved by designing the learning task using other space and action formulations. Additionally, since the issue of which reward scheme to use seems to be a key issue, a possible extension of this work could consider using the methods proposed in [14,15] for designing a reward function that fits this domain best.

A further study will analyze the effect of self-controlled signals by RL on the long-term decisions of travelers, e.g. regarding route or mode choice. With this, the problem becomes bi-level: Signal agents react to sensor data and traveler agents react to experienced travel times that are, in turn, affected by the signal control. As a consequence, delays and queue length might increase again, as intersections that are efficiently controlled attract more traffic. For rule-based adaptive traffic signals this effect was already verified in the simulation [31].

Acknowledgements

The authors thank Prof. Kai Nagel for his support and supervision during the internship of Lucas N. Alegre at TU Berlin. Thanks also to Dr. Ricardo Grunitzki for a previous version of the tabular Q-learning code. The authors are grateful to the Deutsche Forschungsgemeinschaft (DFG) for funding the project *Optimization and network wide analysis of traffic signal control*, as well as to the Brazilian Research Council (grant no. 307215/2017-2 for Ana Bazzan).

References

[1] M. Abdoos, N. Mozayani and A.L.C. Bazzan, Hierarchical control of traffic signals using Q-learning with tile coding, *Appl. Intell.* **40**(2) (2014), 201–213. doi:10.1007/s10489-013-0455-3.

[2] L.N. Alegre, T. Ziemke and A.L.C. Bazzan, Using reinforcement learning to control traffic signals in a real-world scenario, an approach based on linear function approximation (forthcoming).

[3] L. Baird, Residual algorithms: Reinforcement learning with function approximation, in: *Machine Learning Proceedings 1995*, Morgan Kaufmann, 1995, pp. 30–37, http://www.sciencedirect.com/science/article/pii/B978155860377650013X. ISBN 978-1-55860-377-6. doi:10.1016/B978-1-55860-377-6.50013-X.

[4] A.L.C. Bazzan, Opportunities for multiagent systems and multiagent reinforcement learning in traffic control, *Autonomous Agents and Multiagent Systems* **18**(3) (2009), 342–375. doi:10.1007/s10458-008-9062-9.

[5] L.B. de Oliveira and E. Camponogara, Multi-agent model predictive control of signaling split in urban traffic networks, *Transportation Research Part C: Emerging Technologies* **18**(1) (2010), 120–139. doi:10.1016/j.trc.2009.04.022.

[6] M. Di Taranto, UTOPIA, in: *Proc. of the IFAC-IFIP-IFORS Conference on Control, Computers, Communication in Transportation*, International Federation of Automatic Control, Paris, 1989, pp. 245–252.

[7] C. Diakaki, M. Papageorgiou and K. Aboudolas, A multivariable regulator approach to traffic-responsive network-wide signal control, *Control Engineering Practice* **10**(2) (2002), 183–195. doi:10.1016/S0967-0661(01)00121-6.

[8] B. Friedrich, Adaptive signal control – an overview, in: *Proc. of the 9th Meeting of the Euro Working Group Transportation*, Bari, Italy, 2002.

[9] N.H. Gartner, OPAC – a demand-responsive strategy for traffic signal control, *Transportation Research Record* **906** (1983), 75–81.

[10] W. Genders and S. Razavi, Evaluating reinforcement learning state representations for adaptive traffic signal control, *Procedia Computer Science* **130** (2018), 26–33. doi:10.1016/j.procs.2018.04.008.

[11] D. Grether, J. Bischoff and K. Nagel, Traffic-actuated signal control: Simulation of the user benefits in a big event real-world scenario, in: *2nd International Conference on Models and Technologies for ITS*, Leuven, Belgium, 2011.

[12] D. Grether and T. Thunig, Traffic signals and lanes, in: *The Multi-Agent Transport Simulation MATSim*, A. Horni, K. Nagel and K.W. Axhausen, eds, Ubiquity, London, 2016, Chapter 12. doi:10.5334/baw.

[13] D.S. Grether, Extension of a multi-agent transport simulation for traffic signal control and air transport systems, PhD thesis, TU Berlin, Berlin, 2014.

[14] R. Grunitzki, B.C. da Silva and A.L.C. Bazzan, A flexible approach for designing optimal reward functions, in: *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017)*, S. Das, E. Durfee, K. Larson and M. Winikoff, eds, IFAAMAS, São Paulo, 2017, pp. 1559–1560, http://ifaamas.org/Proceedings/aamas2017/pdfs/p1559.pdf.

[15] R. Grunitzki, B.C. da Silva and A.L.C. Bazzan, Towards designing optimal reward functions in multi-agent reinforcement learning problems, in: *Proc. of the 2018 International Joint Conference on Neural Networks (IJCNN 2018)*, Rio de Janeiro, 2018.

[16] J. Henry, J.L. Farges and J. Tuffal, The PRODYN real time traffic algorithm, in: *Proceedings of the Int. Fed. of Aut. Control*, I.F.A.C. Conf and R. Isermann, eds, IFAC, Baden-Baden, 1983, pp. 307–312.

[17] A. Horni, K. Nagel and K.W. Axhausen (eds), *The Multi-Agent Transport Simulation MATSim*, Ubiquity, London, 2016. doi:10.5334/baw.

[18] P.B. Hunt, D.I. Robertson, R.D. Bretherton and R.I. Winton, SCOOT – a traffic responsive method of coordinating signals, in: *TRRL Laboratory Report, 1014*, TRRL, Crowthorne, Berkshire, UK, 1981.

[19] G. Konidaris, S. Osentoski and P. Thomas, Value function approximation in reinforcement learning using the Fourier basis, in: *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI'11*, AAAI Press, 2011, pp. 380–385.

[20] N. Kühnel, T. Thunig and K. Nagel, Implementing an adaptive traffic signal control algorithm in an agent-based transport simulation, *Procedia Computer Science* **130** (2018), 894–899. doi:10.1016/j.procs.2018.04.086.

[21] S. Lämmer, Die Selbst-Steuerung im Praxistest, *Straßenverkehrstechnik* **3** (2016), 143–151.

[22] S. Lämmer and D. Helbing, Self-control of traffic lights and vehicle flows in urban road networks, *Journal of Statistical Mechanics: Theory and Experiment* **2008**(04) (2008), 04. doi:10.1088/1742-5468/2008/04/P04019.

[23] P. Lowrie, The Sydney coordinate adaptive traffic system – principles, methodology, algorithms, in: *Proceedings of the International Conference on Road Traffic Signalling*, Sydney, Australia, 1982.

[24] P. Mannion, J. Duggan and E. Howley, An experimental review of reinforcement learning algorithms for adaptive traffic signal control, in: *Autonomic Road Transport Support Systems*, T. Leo McCluskey, A. Kotsialos, P.J. Müller, F. Klügl, O. Rana and R. Schumann, eds, Springer International Publishing, Cham, 2016, pp. 47–66. doi:10.1007/978-3-319-25808-9_4.

[25] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller, Playing atari with deep reinforcement learning, in: *NIPS Deep Learning Workshop*, 2013.

[26] K.J. Prabuchandran, A.N.H. Kumar and S. Bhatnagar, Decentralized learning for traffic signal control, in: *Proceedings of the 7th International Conference on Communication Systems and Networks (COMSNETS)*, 2015, pp. 1–6. ISBN 9781479984398.

[27] L.A. Prashanth and S. Bhatnagar, Reinforcement learning with average cost for adaptive control of traffic lights at intersections, in: *Procedings of 14th International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2011, pp. 1640–1645. doi:10.1109/ITSC.2011.6082823.

[28] R.S. Sutton and A.G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, 1998.

[29] R.S. Sutton and A.G. Barto, *Reinforcement Learning: An Introduction*, 2nd edn, The MIT Press, 2018.

[30] T. Thunig, N. Kühnel and K. Nagel, Adaptive traffic signal control for real-world scenarios in agent-based transport simulations, *Transportation Research Procedia* **37** (2019), 481–488, http://www.sciencedirect.com/science/article/pii/S2352146518306343. doi:10.1016/j.trpro.2018.12.215.

[31] T. Thunig and K. Nagel, Effects of user adaption on traffic-responsive signal control in agent-based transport simulations, in: *6th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, 2019, pp. 1–7. doi:10.1109/MTITS.2019.8883373.

[32] E. van der Pol, Deep reinforcement learning for coordination in traffic light control, PhD thesis, University of Amsterdam, 2016.

[33] H. van Seijen, A.R. Mahmood, P. Pilarski, M. Machado and R. Sutton, True online temporal-difference learning, *Journal of Machine Learning Research* **17** (2016), 1.

[34] H. Van Seijen and R.S. Sutton, True online TD(λ), in: *Proceedings of the 31st International Conference on International Conference on Machine Learning, ICML'14*, Vol. 32, JMLR.org, 2014, pp. I-692–I-700.

[35] C.J.C.H. Watkins and P. Dayan, *Q-learning, Machine Learning* **8**(3) (1992), 279–292.

[36] F.V. Webster, Traffic signal setting, Technical Report, 39, Road Research Laboratory, London, 1958.

[37] H. Wei, G. Zheng, V.V. Gayah and Z. Li, A survey on traffic signal control methods, 2019, CoRR, abs/1904.08117.

[38] K.-L.A. Yau, J. Qadir, H.L. Khoo, M.H. Ling and P. Komisarczuk, A survey on reinforcement learning models and algorithms for traffic signal control, *ACM Comput. Surv.* **50**(3) (2017). doi:10.1145/3068287.

[39] R. Zhang, A. Ishikawa, W. Wang, B. Striner and O.K. Tonguz, Partially observable reinforcement learning for intelligent transportation systems, 2018, CoRR, abs/1807.01628.

[40] D. Ziemke, I. Kaddoura and K. Nagel, The MATSim open Berlin scenario: A multimodal agent-based transport simulation scenario based on synthetic demand modeling and open data, *Procedia Computer Science* **151** (2019), 870–877. doi:10.1016/j.procs.2019.04.120.