2018 26th Mediterranean Conference on Control and Automation (MED)
Zadar, Croatia, June 19-22, 2018

A Comparison of Different State Representations for Reinforcement Learning Based Variable Speed Limit Control

Krešimir Kušić, Edouard Ivanjko, and Martin Gregurić

Abstract— **Variable Speed Limit Control (VSLC) is one control method for alleviating congestions on urban motorways. Machine learning techniques, like Reinforcement Learning (RL), are a promising alternative for setting up VSLC because an optimal control policy can be achieved with a smaller computational burden in comparison with optimal control approaches. A drawback is a large number of learning iterations and the problem of the exponential expansion of the state space dimension. This can be solved with function approximation techniques. Three different approaches for feature-based state representation in RL based VSLC are compared in this paper regarding the convergence of Total Time Spent. The microscopic traffic simulator VISSIM with a representative traffic model is used to evaluate the compared approaches. Results show that function approximation methods outperform RL based VSLC formulated with a lookup table by an average improvement of 10 %, where feature extraction methods (Coarse and Tile) coding showed slightly faster learning rate.**

I. INTRODUCTION

Urban motorways are roads designed to provide greater traffic capacity and high Level of Service (LoS). They are constructed with a larger number of nearby on- and off-ramps to ensure a good connection with the local urban traffic network. The problem is that LoS on urban motorways decreases due to periodic congestions in areas near on-ramps during peak hours. To avoid such a scenario, it is possible to increase the capacity of the urban motorway by building additional traffic lanes. But, a more acceptable solution is the application of services from the domain of Intelligent Transportation Systems (ITS) for traffic control [1].

One of such services is Variable Speed Limit Control (VSLC). It controls the traffic flow on the motorway by changing the speed limit according to the current traffic (the focus of this paper) or weather situation. The aim is to raise the LoS of the urban motorway, increase traffic safety, reduce vehicle emissions, and homogenize the traffic flow. Standard VSLC approaches are based on online feedback control and the fundamental flow-density relationship mapped to speed values [1], [2].

Nowadays, new approaches for VSLC apply machine learning. Reinforcement Learning (RL) is an often used approach. One class of RL techniques is Q-Learning (QL), also known as model-free learning. QL-based VSLC (QVSLC) can determine optimal behaviour within a specific traffic condition to increase the traffic flow performance [3], [4]. Additionally, the quality can be improved during operation. The controller is an agent that learns how to execute a limited number of actions in its environment. The agent learns directly from the interactions between states and actions through trial and error converging to an optimal VSLC policy (sequence of speed limits) by accumulating the awards in an action-value lookup table (so-called Q-matrix).

All Authors are with the Faculty of Transport and Traffic Sciences, University of Zagreb, Vukelićeva 4, HR-10000 Zagreb, Republic of Croatia, emails: {kresimir.kusic, edouard.ivanjko, martin.greguric}@fpz.hr

When RL is applied to real-world control problems, it often happens that the state-action space is large and continuous. As a result, the size of the lookup table grows exponentially. Since the convergence of the QL algorithm to the optimal policy requires that all state-action pairs are visited sufficiently many times, the learning process becomes unfeasible [5]. Instead of computing the exact Q-value, the aim is to calculate its approximation covering the whole state-action space with fine enough partitioning. QL with function approximation can successfully solve tasks within higher-dimensional continuous space representation [6]. Function approximation methods are also suitable to create an estimation of the Q-values in regions of the state-action space that were not visited in the learning process. The idea is to construct appropriate basis functions (features) whose linear combination can achieve a reasonable approximation of the Q-value. Features capture important properties of the agent's environment described as a continuous space. Linear function approximation is attractive because it results with simple update rules (gradient descent) and has a convex error surface with a single global optimum [7].

Function approximation in RL based VSLC has been successfully applied by several researchers [3], [8], [9]. Beside linear approximation, other approaches like artificial neural networks in [3] and k-nearest neighbours clustering technique [9] have been studied also. The analysis of learning convergence of RL based VSLC approaches using microscopic simulations for learning and evaluation is still an open area since mostly macroscopic analysis is applied [9]. Well-known feature-based state representation methods are based on the polynomial basis, Fourier basis, coarse coding, tile coding, and Radial Basis Function (RBF). In this paper, the convergence regarding Total Time Spent (TTS) of tile and coarse coding, and RBF based QVSLC with function approximation (QVSLC-FA) is compared to QVSLC with full state representation (QVSLC-FS). A comparison to the no-control case is made also.

II. VARIABLE SPEED LIMIT CONTROL

As mentioned, one measure for prevention of congestions on urban motorways from the domain of ITS is VSLC. It consists of appropriate Variable Message Signs (VMS) used for displaying variable speed limit values in response to the prevailing traffic conditions. The VMS has to be placed before the section of the urban motorway where congestion occurs. Difference between two consecutive speed limits is usually limited to 20 $[\frac{km}{h}]$.

The main impact of VSLC on the traffic flow is twofold. According to [10], the first impact emphasizes the homogenization effect, whereas the second is more focused on preventing traffic breakdown by reducing the flow using speed limits. The basic idea of homogenization is that speed limits can reduce fluctuations in traffic parameters, i.e. speed differences of vehicles between lanes and within the same lane are reduced. Consequently, a much more stable and safer traffic flow appears and a capacity drop is avoided. Using VSLC, mean speed of vehicles is reduced which directly decreases the mainstream flow arriving at the bottleneck area under the values that can cause the occurrence of critical traffic density and consequently traffic congestion [11]. Thus, the congestion pre-phase can be prolonged and the congestion phase shortened or even entirely avoided. Additional motivation to use VSLC is enhanced traffic safety, where positive impact of VSLC on traffic safety is induced by speed reduction and speed homogenization, which are correlated with a decrease in accident probability [11].

III. Q-LEARNING AND STATE REPRESENTATION

Optimization of VSLC requires the determination of an optimal policy for posting speed limits as actions. For this RL can be applied. The QL algorithm is one of the most commonly used RL model-free algorithm known as off-policy temporal difference control [12], [5]. At every time step, the agent perceives the state of the environment and takes action to transfer the current state to a new state. Then the agent receives a reward to evaluate the quality of the transition. The mapping from the environmental state to the selection of action, $\pi\colon S \mapsto A$, is known as a policy function that specifies what action to take in every state. Through training, the agent learns the policy directly and may cover the whole state-action space and learn the state transition function. By evaluating the rewards of multiple activities, the agent learns how to find a sequence of optimal actions that yield the maximum discounted cumulative rewards over time, $\sum_{t=0}^{\infty} \gamma^t r_t$. The parameter γ is the discount factor that defines the relative importance of the current reward and those earned earlier, $0 \leqslant \gamma \leqslant 1$.

A. Variable Speed Limit Control as an MDP

In RL an agent is used to interact in discrete time steps with its environment that has to be formulated as a Markov Decision Process (MDP). For this, the VSLC problem has to be defined as an MDP with the assumption that an agent makes control decisions. The agent has to activate different speed limits at the end of every decision interval. For every possible state of the environment, the agent can select a particular action and obtains an award depending on how well the selected action had performed. The agent cares for accumulated reward gained from a sequence of executed actions. The transition time between states after activating VSLC control equals the control time step. The state changes every time when the agent takes an action that affects the current traffic state. Thus, the VSLC decision process can be formulated as an MDP problem and solved applying RL [4].

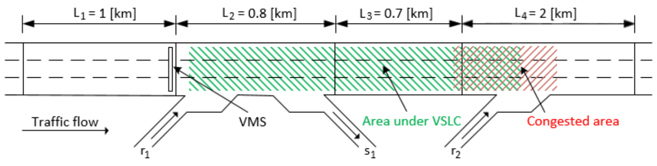

Fig. 1. Controlled motorway stretch divided into four sections

For QVSLC, the controlled stretch of the urban motorway divided into smaller sections (see Fig. 1) represents the model of the agent's environment. Actions are speed limits that the agent sends to the VMS. For each state $s_t \in S$ of the environment, the agent can select an action a_t from a finite set of actions A. For simplicity, the executed action a_t at time t is a single speed limit posted on VMS at the beginning of the controlled sections (L_2 and L_3 in Fig. 1) simultaneously for all analyzed approaches. Traffic density and speed measured in several consecutive sections (L_2, L_3, and L_4 in Fig. 1) are used for state representation.

The reward r_t received by the agent at time step t after arriving in the new state s_{t+1} has the following form for all implemented QVSLC approaches:

$$r_t = \begin{cases} -\delta, \text{ if } (a(t-2) = a(t-1))^\wedge(a(t-1) \neq a(t)) \\ -\delta, \text{ if } |a(t-1) - a(t)| > 20 \\ 0, \text{ if } 105 < min\{v_i(t+1) \mid i = 2,3,4\} < 110 \\ \delta, \text{ if } min\{v_i(t+1) \mid i = 2,3,4\} \geqslant 110 \\ -TTS(t), \text{ if otherwise} \end{cases}, \quad (1)$$

with slight modification compared to the reward function used in [3] by including the positive reward δ in the case when the agent recognized free-flow conditions. The constant δ should be at least equal to the maximal expected TTS value, measured between the previous and the current control time step. The first condition of r_t prevents oscillations of the speed limit, and second presents punishment if the difference between two consecutive speed limits is too large. If the average speed of the observed motorway sections is between $105 < v_i < 110$, it can be assumed that there is currently no congestion. Therefore, the agent does not receive punishment in such states. In the case where measured average speeds are above 110 $[\frac{km}{h}]$ free flow traffic is assumed, and the agent receives a reward. This two conditions can be true if the agent executes actions related to speed limits higher then the free flow speed, and there is a free flow condition across affected sections. In all other cases, the agent receives a punishment that is proportional to TTS spent between the previous and current control time step measured across the affected area (L_2, L_3 and beginning of L_4 in Fig. 1) during every control time interval. As such TTS can be used to define the objective function of the traffic flow optimization [3], [13].

In the beginning, the agent does not know anything about the environment and which action to take. An exploration-exploitation strategy has to be applied for testing all possible actions. In this paper, the ϵ-*greedy* mechanism is applied in all compared approaches. First, the agent tries random actions, and with time the agent starts to use the learned knowledge. The exploration parameter ϵ bounded with $0 < \epsilon < 1$ is used to manage the share of exploration and exploitation during learning. When $\epsilon \to 1$ the agent acts only randomly and can even worsen the current LoS on the urban motorway. When $\epsilon \to 0$ the agent uses the learned knowledge stored in a Q-matrix (the memory of what the agent has learned through experience). During multiple simulations, ϵ is gradually decreased according to the function $\epsilon{=}\exp\frac{(1-n)}{600}$ (n denotes the number of the current simulation). When ϵ reaches 0.03 it remains constant.

B. Q-Learning with Full State Representation

For an implementation of QVSLC-FS, discrete states and actions need to be defined. Actions belong to one of the following two sets:

$$A_1 = \{130, 110, 100, 90, 80, 70, 60\}, \tag{2}$$

$$A_2 = \{130, 110, 100, 80, 60\}, \tag{3}$$

where the set A_1 has been used for the analysis of QVSLC-FS and QVSLC-FA, and the smaller set A_2 only in QVSLC-FS with constant learning rate $\alpha = 0.5$. This has been done to make a Q-matrix with a smaller number of elements (3125).

A Q-value is assigned to each state-action pair as a measure of the quality of each combination. The learned Q-value function $Q\colon\ S \times A \mapsto \mathbb{R}$ represents a mapping from state-action pairs to expected long-term return obtained by executing a specific action in a given state [5], [14]. For a non-deterministic environment, the basic idea of QL is to update the Q-value iteratively by using the new received training sample $(s_t, a_t, s_{t+1}, r_{t+1})$ according to:

$$\begin{aligned} Q(s_t, a_t) &= Q(s_t, a_t) \\ &\quad + \alpha_n (r_{t+1} + \gamma \max_{a' \in A} Q(s_{t+1}, a') - Q(s_t, a_t)), \end{aligned} \tag{4}$$

where $Q(s_t, a_t)$ is the Q-value for the respective state-action pair (s_t, a_t) at time step $(t+1)$, r_{t+1} is the reward received after performing the action a_t in state s_t and inducing a change to the new state s_{t+1}, and parameter α_n is the learning rate which controls how fast the Q-values are altered (for QVSLC-FS $\alpha_n{=}\frac{1}{n}$). The rate of α_n should be decreased over time to ensure convergence to an optimal policy as explained in [5]. In this analysis, γ is set to 0.8, and it remains the same for all tested approaches. Q-values for implemented QVSLC-FS are stored in a five-dimensional Q-matrix with 6125 elements according to $|S \times A_1|$, where S is a finite set of states defined as:

$$S = \{\rho_2(t), \rho_3(t), \rho_4(t), a(t-1)\}, \tag{5}$$

where $\rho_i(t)$ represents the density of the traffic flow at the ith section for time step t. Different density values are coded in five grades based on the values $(10, 15, 22, 30)$. The critical density for the applied model is $\rho_c = 29\ [veh/km/lane]$. The last term in (5) $a(t-1)$ is the speed limit from the previous control time interval.

The QL algorithm (4) converges to the optimal Q-values if every state-action pair is visited plenty often and the learning rate is decreased appropriately over time. After the Q-values for sufficiently many state-action couples have been estimated during the learning process, the optimal action for a particular state is determined as the one with the largest Q-value. Then the QL agent can be applied for optimal control using only its knowledge.

C. Q-Learning with Function Approximation

The update rule for QVSLC-FS applies (4) to obtain an optimal speed limit policy. This approach requires a lookup table to store the learned Q-values for all possible (a_t, s_t) pairs. When RL is applied for traffic control on an urban motorway, states can be defined using traffic parameters such as normalized density and average speed, which are suitable to represent all traffic situations. As a result, the sets S, and A become vast or infinite, and the stochastic algorithm (4) loses its efficiency. Dimensions of state representation grow depending on the discretization of traffic flow parameters. The size of the lookup table used to store Q-values grows exponentially revelling the curse of dimensionality. To visit every state-action pair plenty often becomes impossible [5]. Thus, learning the optimal Q-values requires some form of function approximation.

Feature-based state representations have been used as a method for constructing the basis for the function approximation. Coarse and tile coding, and RBF have been chosen for implementation and comparison. Features capture important properties of continuous space of the agent's environment. In case of coarse and tile coding, the state space is mapped into a vector of binary features. RBF uses the Gaussian function to map the state vector $\vec{s_t}$ into features represented with real numbers within the interval $[0, 1]$. Mentioned approaches are used to create a basis function whose combinations approximate the Q-values. In the setting of QVSLC-FA, the idea is to approximate the Q-value by learning the parameter vector $\vec{\theta}$ of an approximate value function Q_θ as:

$$Q_\theta(s, a) = \vec{\theta}^T \vec{\phi}_{s,a} \approx Q(s, a), \tag{6}$$

where $\vec{\phi}_{s,a}$ is an m-dimensional column vector that captures important properties from the state-action pair (s_t, a_t), and $\vec{\theta}$ is a parameter whose dimension is identical to $\vec{\phi}_{s,a}$ [8]. At the beginning all components of $\vec{\theta}$ are set to zero. Now, the task is to learn $\vec{\theta}$ by applying the incremental stochastic gradient descent update rule (7) allowing approximation of the Q-value function (6):

$$\begin{aligned} &\vec{\theta}_{t+1} = \vec{\theta}_t + \\ &\alpha_n (r_{t+1} + \gamma \max_{a' \in A} (\vec{\theta}_t^T \vec{\phi}_{s_{t+1}, a'}) - \vec{\theta}_t^T \vec{\phi}_{s_t, a_t}) \vec{\phi}_{s_t, a_t}, \end{aligned} \tag{7}$$

where $\alpha_n = \frac{1}{10k+n}$ and k is the number of tilings explained in the next section.

D. Feature-Based State Representation

Generalization of states is essential when QL is applied to a continuous state space. Feature-based representation captures important properties of the state. Often to binary numbers $(0/1)$ (coarse and tile coding) or to the interval $[0,1]$ (RBF). The used state vector $\vec{s}_t$ was slightly modified to the one used in [3]. Three additional density states were added for improving the representation of the current traffic situation. In all three methods of feature constructions (coarse coding, tile coding, and RBF) the state set becomes now a state vector $\vec{s}_t \in \mathbb{R}^{m_0}$, where $m_0 = 8$ is the number of state components. Also, the chosen state vector comprises average speeds and past two actions as components of the current state, still satisfying MDP conditions:

$$\vec{s}_t = \Big(\frac{a(t-1)}{v_f}, \frac{a(t-2)}{v_f}, \frac{\rho_2(t)}{\rho_j}, \frac{\rho_3(t)}{\rho_j}, \frac{\rho_4(t)}{\rho_j}, \frac{v_2(t)}{v_f}, \frac{v_3(t)}{v_f}, \frac{v_4(t)}{v_f}\Big). \quad (8)$$

The numerators of the first two components of the state vector (8) are the previously executed speed limit values. Variable $\rho_i(t)$ is the current density, and $v_i(t)$ is the current average speed in section i. Every element in the state vector (8) is normalized into the interval $[0,1]$ by appropriate denominators, where $v_f = 130$ $[\frac{km}{h}]$ is the free flow speed, and $\rho_j = 80$ $[veh/km/lane]$ is the jam density. Using the earlier mentioned feature based methods, the important properties from (8) are captured into the feature vector $\vec{\phi}_{s,a}$.

This feature vector is defined differently for the three chosen state representations as follows:

1) Coarse Coding: The receptive field corresponds to circles in the state space. According to two-dimensional state space, if the state coordinate is inside the circle, then the corresponding feature has the value 1 and is said to be present. Otherwise, the feature is equal to 0. The receptive field can overlap, enabling generalization between different states as it can be seen in [12]. Extending of coarse coding in an eight-dimensional state space corresponds to $\vec{s}_t$ (8), and the receptive fields are now hypersphere tiles with radius $r = 0.7$. The state space was filled with $k = 900$ points. Each of those represents the centre of a hypersphere. The points are placed randomly within the state space, thus achieving an overlap effect. The dimension of the feature vector (9) is $m = 1 + |A_1|l$ with $|A_1|$ representing the cardinality of the action set, while $l = k$, plus one extra element for bias. The same state vector is used in all three approaches with function approximation. Each possible combination of actions and hypersphere tiles has a unique component in $\vec{\theta}$.

2) Tile Coding: The receptive field of the features is grouped into partitions of the input space. Each partition is called a tiling, and each element of the partition is called a tile. According to a two-dimensional state space example the simplest tiling is a uniform grid, where receptive field (tiles) are squares [12]. One tiling in eight-dimension fills the state space with p^{m_0} equally spaced hypercube tiles. The binary feature vector in this case is $\mathbb{R}^m$, where $m = 1+|A_1|l$. Here $l = kp^{m_0}$, parameter p=3 is the number of tiles (hypercubes) along one dimension, parameter $k = 64$ is the number of tilings, and m_0 is the dimension of the state vector (8). To ensure a higher resolution of the state space partitioning, k tilings are created. Each tiling is shifted by the displacement vector $\vec{d} = (\frac{1}{10}, \frac{3}{10}, \frac{5}{10}, \frac{7}{10}, \frac{9}{10}, \frac{11}{10}, \frac{13}{10}, \frac{15}{10})$, meaning that it is shifted from the previous tiling by $\frac{\omega}{k}$ times $\vec{d}$ [12]. Tile width ω was defined as $\omega = \frac{1.4}{(p-1+\frac{1}{k})}$. With those small shifts, the state space is filled with k overlapping tiles. A single point within $\mathbb{R}^{m_0}$, corresponds to the coordinates of the state vector (8), will fall in precisely one tile in every of the k tilings. These k tiles correspond to k features in (9) that become active when a particular state occurs, and particular action has been executed. As mentioned, the feature vector is in $\mathbb{R}^m$ and has the following form:

$$\vec{\phi}_{s,a} = (1, \phi_1(s,a_1), \ldots, \phi_l(s,a_1), \ldots, \phi_1(s,a_7), \ldots, \phi_l(s,a_7)), \quad (9)$$

where index of action a indicates the speed limit which has been executed in respective state s. The index l stands for possible active tiles at time t. First component stands for the bias term to properly scale the function values. The number of total elements in (9) seems a bit bigger, but computing the dot product of $\vec{\theta}^T\vec{\phi}$ in (7) gains computational advantages because most binary features in (9) are always zero.

3) RBF: This approach is a natural generalization of coarse coding to continuous-valued features. The feature can gain value from the interval $[0,1]$. Typical RBF uses the following Gaussian response:

$$\phi_{i_{s,a}} = \exp\left(-\frac{||\vec{s}_t - \vec{c}_i||^2}{2\sigma_i^2}\right), \quad (10)$$

where $\{\phi_i(||\vec{s}_t - \vec{c}_i||) \,|\, i = 1, 2, \ldots, k\}$ is a set of k arbitrary functions, known as radial basis functions. The ith function is a component of the feature vector $\vec{\phi}_{s,a}$. The response of (10) depends on the distance between the state vector $\vec{s}_t$, and the center of the basis function $\vec{c}_i$, and relatively to the width σ_i of the radial basis function with respect to the center $\vec{c}_i$ [12]. The feature vector is in $\mathbb{R}^m$, where $m = 1 + |A_1|l$. In the implementation of QVSLC using feature construction with RBF, $l = k = 64$, and σ_i has been set to 0.5 according to [15]. This method reduces performance when there is more than two state dimensions like in QVSLC because all components (associated with the currently executed action) in (9) are active at time t. But so far, the number of RBF functions (10) within the feature vector (9) has been reduced compared to the previous two cases.

In all mentioned QVSLC-FA approaches (coarse, tile, and RBF) the components in (9) associated with actions that are not executed at time t will be zero, except the bias term.

IV. SIMULATION RESULTS

To simulate the described QVSLC implementations, a simulation framework consisting of the microscopic simulator VISSIM and MATLAB is used [2], [13]. Every simulation lasted 2.5 $[h]$. All QVSLC approaches are learned during 5000 simulations with different seeds to generate a stochastic traffic flow i.e. a stochastic environment for the agent.

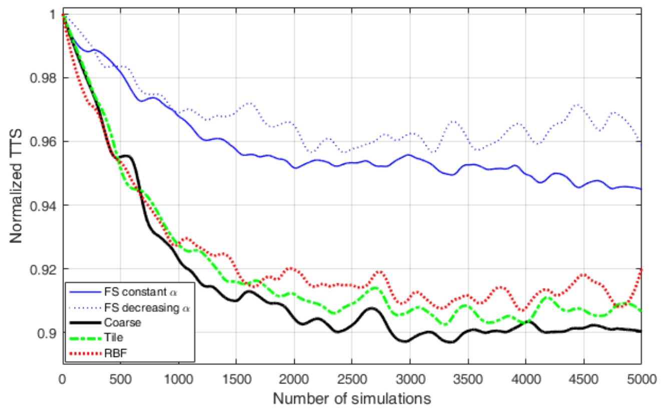

Fig. 2. Convergence of the normalized TTS during the learning process

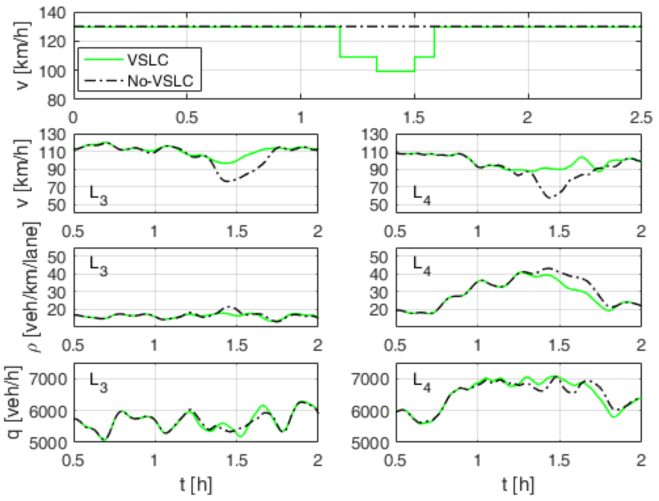

Fig. 3. Traffic parameters for tile coding in sections L_3 and L_4

A. Model of the Urban Motorway

Used motorway model was taken from [16] and modified to be suitable for the applied simulation framework [13]. It is a three-lane motorway as shown in Fig. 1 and divided into four sections. VSLC is active within the green area in sections L_2 and L_3, and at the beginning of L_4. The first section L_1 is without VSLC. One has to note that in a real-world situation, vehicles decelerate before the VMS, but in the applied simulator, vehicles do not react until they pass the VMS. This phenomenon is indicated by a shift of the green region into section L_4. New speed limit value is sent to the VMS (see Fig. 1) during every control time step, T_c (300 $[s]$ in this paper). All traffic data are collected consecutively regarding the sampling time, T (30 $[s]$ in this paper). Section L_2 contains one on- and off-ramp (r_1 and s_1). The second on-ramp r_2 is placed in section L_4, and here congestion is created by changing the input flow at this on-ramp. Congestion gradually propagates upstream into section L_3 and creates a disturbance in it.

The simulated traffic flow consists of 96 % cars, 2 % trucks, and 2 % of buses. The mainstream flow has a constant demand of 4500 $[\frac{veh}{h}]$ during the whole simulation, of which 95 % of traffic remains on the main lanes while the other 5 % exit through the off-ramp s_1. The on-ramp r_1 has a constant traffic demand of 1350 $[\frac{veh}{h}]$ while traffic demand at the on-ramp r_2 changes during simulation. It starts with a constant value of 300 $[\frac{veh}{h}]$, whereupon it increases linearly reaching the maximum of 1250 $[\frac{veh}{h}]$. This value remains for a half an hour after which the traffic demand linearly decreases to its starting value and stays constant to the end of the simulation. One has to notice that the applied traffic simulator generates vehicles for the envisaged traffic flows stochastically using a particular seed for each repeatable simulation.

B. Results

Normalized TTS values obtained during the learning process are given in Fig. 2. A polynomial interpolation was used for a comprehensive illustration. All approaches reduce TTS, but linear function approximation approaches have a steeper decrease rate. The two blue curves represent the results of QVSLC-FS algorithms. The first has a constant learning rate α, and it learns faster due to the lower number of elements in Q-matrix as a consequence of the smaller action set A_2. The second has a decreasing learning rate α_n. Slower learning rate can be related to the fast decrease of parameter α_n, where the agent cannot correct wrong actions executed at the beginning of the learning process. In general, given QVSLC-FA methods with state generalization did better than QVSLC-FS. All QVSLC-FA methods have a similar TTS decrease rate with coarse coding having a somewhat steepest one.

Impact of VSLC on the mainstream traffic parameters is most evident in sections L_3 and L_4 as shown in Fig. 3. Two cases are shown. First is the case of no-control (denoted black) and second for QVSLC with tile coding (indicated in green). Tile coding was selected for presentation since the typical gradual decrease of the speed limit, and its increase without large unallowed changes is most apparent. This is a desirable behaviour of VSLC that had to be learned [10]. In the density graph in Fig. 3 for section L_4 the positive effect of VSLC is evident with actions that actively reduce the density compared with the case of no-control. The timely applied sequence of speed limits keeps the traffic flow speed at a higher value compared to the case of no-control. The gradual reduction of speed of the vehicles coming into section L_4 allows the congestion to dissolve more quickly than in the case of no-control. After the congestion has dissolved, a steep increase in speed in sections L_3 and L_4 can be observed in comparison to the case of no-control. Other sections and QVSLC approaches are not shown due to lack of space, but a similar behaviour can be observed for other compared approaches.

Additional Measures of Effectiveness (MoE) (Travel Time (TT) on mainstream, TTS, and queue length at the on-ramp r_2) were gathered during simulations. The performance has improved regarding these additional MoEs also as shown in Table I. QVSLC improves the LoS on the mainstream (lower TT) and at the congested on-ramp (shorter queues). Only QVSLC-FS with decreasing α_n has a minor deterioration

TABLE I

OBTAINED TRAFFIC MOES FOR BEST CASES

	No	QVSLC-FS $\alpha = 0.5$		QVSLC-FS dec. α		QVSLC-FA Coarse		QVSLC-FA Tile		QVSLC-FA RBF	
	VSLC	Obtained	Red. $[\%]$	Obtained	Red. $[\%]$	Obtained	Red. $[\%]$	Obtained	Red. $[\%]$	Obtained	Red. $[\%]$
Max. TT $[s]$	387	322	16.7	285	26.2	304	21.2	319	17.5	313	19.1
Avg. TT $[s]$	184	184	0	185	-0.6	177	3.6	179	2.3	180	1.9
TTS $[veh \cdot h]$	749	729	2.6	736	1.7	708	5.5	714	4.6	723	3.5
Max. Queue $[veh]$	36	31	13.9	11	69.4	16	55.6	21	41.7	36	0
Avg. Queue $[veh]$	4.2	4.0	4.1	0.9	77.0	1.2	71.3	1.2	70.7	2.5	39.2

of TT. The reduction of TTS in Table I is somewhat lower than in Fig. 2 because random actions at the beginning of the learning process worsen the LoS on the controlled motorway compared to the case of no-control. Coarse coding shows the best behaviour regarding obtained convergence rate and value of TTS.

V. CONCLUSIONS

Four different approaches for state representation for QVSLC are applied in this paper to learn the optimal VSLC policy for minimizing TTS on urban motorways. Obtained policies are evaluated using a simulation framework consisting of the microscopic simulator VISSIM and MATLAB. The simulation results show that function approximation can learn the needed control policy from scratch faster compared to standard full state representation. In about 5000 simulations the TTS value was reduced up to 10% compared to the starting value by applying function approximation. Coarse coding obtained the best results with the fastest convergence of TTS. The learned Q-value function can also be used as an input knowledge for QVSLC being applied on another similar motorway segment. The agent only needs to update the lacking knowledge of the specifics of the new environment, thus avoiding learning from scratch again.

Feature construction for linear methods is a practically applicable method to generalize a continuous higher dimensional state space. In this case, generalization is needed to capture essential properties of multiple traffic state variables. But, with the increase of state dimensionality, the feature's vector grows exponentially depending on the resolution of the state space partitioning. Therefore, future research will be focused on finding most representative traffic parameters to reduce the dimension of the state vector. This would enable finer partitioning (higher resolution) of the state space thus improving the generalization of unvisited space regions. Also, implemented QVSLC will be augmented to a multi-agent approach to assign speed limits to several consecutive sections on the controlled urban motorway.

ACKNOWLEDGMENT

The authors would like to thank the company PTV Group for providing a research license of the microscopic simulator VISSIM. This work has been partly supported by the Faculty of Transport and Traffic Sciences, University of Zagreb and by the European Regional Development Fund under the grant KK.01.1.1.01.0009 (DATACROSS). This research has also been carried out within the activities of the Centre of Research Excellence for Data Science and Cooperative Systems supported by the Ministry of Science and Education of the Republic of Croatia.

REFERENCES

[1] M. Hadjipollas, M. Lestas, P. Ioannou, G. Hadjipollas, and A. Pitsillides, “Evaluation of new ramp metering and variable speed limit algorithms on the Cyprus Highway Network,” *International Trade and Freight Transportation Conference (ITFTC2008)*, pp. 171–177, 2008.

[2] K. Kušić, N. Korent, M. Gregurić, and E. Ivanjko, “Comparison of two controllers for variable speed limit control,” in *2016 International Symposium ELMAR*, Zadar, Croatia, 12-14 Sept 2016, pp. 101–106.

[3] E. Walraven, “Traffic Flow Optimization using Reinforcement Learning,” Master's thesis, Faculty EEMCS, Delft University of Technology, the Netherlands, 2014.

[4] Z. Li, P. Liu, C. Xu, H. Duan, and W. Wang, “Reinforcement learning-based variable speed limit control strategy to reduce traffic congestion at freeway recurrent bottlenecks,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 11, pp. 3204–3217, Nov 2017.

[5] C. J. C. H. Watkins and P. Dayan, “Q-learning,” in *Machine Learning*, 1992, pp. 279–292.

[6] R. S. Sutton, “Generalization in reinforcement learning: Successful examples using sparse coarse coding,” in *Advances in Neural Information Processing Systems 8*. MIT Press, 1996, pp. 1038–1044.

[7] G. Konidaris, S. Osentoski, and P. Thomas, “Value function approximation in reinforcement learning using the Fourier basis,” in *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence*, August 2011, pp. 380–385.

[8] L. A. Prashanth and S. Bhatnagar, “Reinforcement learning with function approximation for traffic signal control,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 412–421, 2011.

[9] T. Schmidt-Dumont and J. V. Vuuren, “Decentralised reinforcement learning for ramp metering and variable speed limits on highways,” September 2017, submitted for review in IEEE Transactions on Intelligent Transportation Systems.

[10] A. Hegyi, B. D. Schutter, and J. Hellendoorn, “Optimal coordination of variable speed limits to suppress shock waves,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 1, pp. 102–112, 2005.

[11] M. Papageorgiou, E. Kosmatopoulos, and I. Papamichail, “Effects of variable speed limits on motorway traffic flow,” *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2047, pp. 37–48, 2008.

[12] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, ser. A Bradford book. Bradford Book, 1998.

[13] K. Kušić, “Framework for Simulation of Variable Speed Limit Control Systems on Urban Motorways Based on Learning,” Master's thesis, Faculty of Transport and Traffic Science, Croatia, 2017.

[14] A. A. Sherstov and P. Stone, “Function approximation via tile coding: Automating parameter choice,” in *Proceedings of the 6th International Conference on Abstraction, Reformulation and Approximation*, ser. SARA'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 194–205.

[15] R. M. Kretchmar and C. W. Anderson, “Comparison of CMACs and radial basis functions for local function approximators in reinforcement learning,” in *International Conference on Neural Networks*, vol. 2, Jun 1997, pp. 834–837.

[16] I. Papamichail, K. Kampitaki, M. Papageorgiou, and A. Messmer, “Integrated ramp metering and variable speed limit control of motorway traffic flow,” *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 14 084 – 14 089, 2008.