Interpretable Multi Time-scale Constraints in Model-free Deep Reinforcement Learning for Autonomous Driving

Gabriel Kalweit*,1, Maria Huegle*,1, Moritz Werling2 and Joschka Boedecker1,3

Abstract—In many real world applications, reinforcement learning agents have to optimize multiple objectives while following certain rules or satisfying a list of constraints. Classical methods based on reward shaping, i.e. a weighted combination of different objectives in the reward signal, or Lagrangian methods, including constraints in the loss function, have no guarantees that the agent satisfies the constraints at all points in time and lack in interpretability. When a discrete policy is extracted from an action-value function, safe actions can be ensured by restricting the action space at maximization, but can lead to sub-optimal solutions among feasible alternatives. In this work, we propose Multi Time-scale Constrained DQN, a novel algorithm restricting the action space directly in the Q-update to learn the optimal Q-function for the constrained MDP and the corresponding safe policy. In addition to single-step constraints referring only to the next action, we introduce a formulation for approximate multi-step constraints under the current target policy based on truncated value-functions to enhance interpretability. We compare our algorithm to reward shaping and Lagrangian methods in the application of high-level decision making in autonomous driving, considering constraints for safety, keeping right and comfort. We train our agent in the open-source simulator SUMO and on the real HighD data set.

I. INTRODUCTION

Deep reinforcement learning algorithms have achieved state-of-the-art performance in many domains in recent years [1]–[4]. The goal for a reinforcement learning (RL) agent is to maximize the expected accumulated reward which it collects while interacting with its environment. However, in contrast to commonly used simulated benchmarks like computer games [5] or MuJoCo environments [6], in real-world applications such as autonomous driving the reward signal is not pre-defined and has to be hand-engineered. Formulating an immediate reward function such that the outcome of the training process is consistent with the goals of the task designer can be very hard though, especially in cases where different objectives have to be combined. Nonetheless, it is crucial for many safety-critical tasks such as autonomous driving amongst others. One way to approach this problem is to use a weighted sum in the immediate reward function, commonly known as reward shaping, and apply classical RL algorithms such as DQN [1] directly without further modifications. In practice, finding the suitable coefficients for the different objectives requires prior knowledge about the task domain or hyperparameter optimization which can be very time consuming. Other, more sophisticated multi-objective approaches [7]–[9] use multiple reward signals and value-functions and try to find Pareto-optimal solutions, i.e. solutions that cannot be improved in at least one objective. Picking one of the Pareto-optimal solutions for execution is, however, non-trivial. Another common approach to ensure consistency with constraints in Q-learning [10], referred to as Safe Policy Extraction (SPE) in the following, is to restrict the action space during policy extraction [11], [12], masking out all actions leading to constraint violations. As we show in this work, however, this approach can lead to non-optimal policies under the given set of constraints.

Notably, in many applications there is one primary objective (e.g. driving as close as possible to a desired velocity) to be optimized, while additional auxiliary costs are used to guide the agent and ensure various side-constraints (e.g. avoid crashes or guarantee comfort). An exemplary setup with multiple objectives can be seen in Figure 1. A common formulation for reinforcement learning with constraints is the constrained Markov Decision Process (CMDP) framework [13], where instead of a weighted combination of the different objectives, agents are optimizing one objective while satisfying constraints on expectations of auxiliary costs. We propose a novel Q-learning algorithm that satisfies a list of single-step and multi-step constraints, where we model multi-step constraints as expectations of auxiliary costs as in the CMDP framework. These multi-step constraints, however, are estimated via truncated value-functions [14], to approximate constraint costs over the next $H$ steps following the current target-policy. The benefit of this formulation is that constraints are independent from the scaling of the immediate reward function and can act on different time scales which allows for an easier and more

*Authors contributed equally.
1Dept. of Computer Science, University of Freiburg, Germany. {kalweitg,hueglem,jboedeck}@cs.uni-freiburg.de
2BMWGroup, Unterschleissheim, Germany. Moritz.Werling@bmw.de
3Cluster of Excellence BrainLinks-BrainTools, Freiburg, Germany.
formulate desired behavior as constraints for a predictable short-term horizon

Our contributions are threefold. First, we introduce a new class of multi-step constraints which refer to the current target policy, so as to increase interpretability. Second, we define an extension of the update in Q-learning which modifies the action selection of the maximization step to ensure an optimal policy with constraint satisfaction in the long-term. We further show that the constrained update leads to the optimal deterministic policy for the case of Constrained Policy Iteration. Third, we employ Multi Time-scale Constrained Q-learning within DQN and evaluate its performance in high-level decision making for autonomous driving. We show that Multi Time-scale Constrained DQN (MTS-CDQN) is able to outperform reward shaping. Safe Policy Extraction and Lagrangian optimization techniques in the context of this application. We further use the open HighD data set [15], containing 147 hours of top-down recordings of German highways, to learn a smooth and anticipatory driving policy satisfying traffic rules.

**II. RELATED WORK**

A plethora of work exists to find solutions for CMDPs, most of them belonging to (1) Trust region methods [16] or (2) Lagrange multiplier methods [17]–[19], where the CMDP is converted into an equivalent unconstrained problem by making infeasible solutions sub-optimal. However, these methods only guarantee near-constraint satisfaction at each iteration. In Reward Constraint Policy Optimization (RCPO), constraints are represented by reward penalties which are added to the immediate reward function via optimized Lagrange multipliers [20]. Since the approach optimizes both long-term reward and long-term penalty simultaneously, no clear distinction between return and constraint violation can be formalized. This stands in contrast to our work, where return and constraints can act on different time-scales to increase interpretability. Put differently, our approach provides the possibility to formulate constraints on the short-time scale, but optimizes satisfaction of these constraints on the long-term horizon, as shown in Figure 2. Further, RCPO is an on-policy method, whereas our approach belongs to the family of off-policy Q-learning algorithms. In [21], batch-constrained reinforcement learning was proposed to improve off-policy learning from a fixed batch of transitions, which restricts the action space in DQN and DDPG in order to force the RL agent to act close to on-policy with respect to the current batch of transitions. The approach was aiming at the minimization of the extrapolation error in off-policy learning which is introduced by a mismatch between the data set and the true state-action visitation of the current policy. In the context of autonomous lane changes, DQN was also used in [22]–[24]. In [11], [12], DQN is combined with SPE to filter out unsafe actions, where the set of transitions for off-policy RL is collected by an agent with enabled safety module. Q-learning has also been applied to a highway simulation parameterized on the basis of the HighD data set [15] in [25]. Robust control methods are able to model constraints on the short-term horizon and ensure their long-term satisfaction through constraints on terminal costs [26]–[28]. However, they rely on accurate models of the environment, which, in applications of automated driving, contains multiple traffic participants with unknown policies. There exists prior work to predict the behaviour of other vehicles [29], but accurate modeling is still a very challenging problem. Our approach combines the intuitive formulation of constraints on the short-term horizon as in model-based approaches with the robustness of a model-free RL method for the long-term optimization.

**III. PRELIMINARIES**

In this section, we define the theoretical background.

A. Markov Decision Processes (MDP)

In a reinforcement learning setting, an agent interacts with an environment, which is typically modeled as an MDP \( \langle S, A, P, r, \gamma \rangle \). The agent is following policy \( \pi : S \to A \) in some state \( s_t \), applying a discrete action \( a_t \sim \pi \) to reach a successor state \( s_{t+1} \sim P \) according to a transition model \( P \). In every discrete time step \( t \), the agent receives reward \( r_t \) for selecting action \( a_t \) in state \( s_t \). The goal of the agent is to maximize the expectation of the discounted long-term return \( \max_{a_t} \mathbb{E}_{a_t} \sum_{i \geq t} \gamma^{i-t} r_i \), where \( \gamma \in [0, 1) \) is the discount factor. The action-value function \( Q^\pi(s_t, a_t) = \mathbb{E}_{a_{t+1} \sim \pi_{t+1} : s_{t+1} \sim P} \mathbb{E}_{r_t} \sum_{i \geq t} \gamma^{i-t} r_i \) represents the value of following a policy \( \pi \) after applying action \( a_t \). The optimal policy can be inferred from the optimal action-value function \( Q^\ast(s_t, a_t) = \max_{a_t} Q^\ast(s_t, a) \) by maximization.

B. Constrained Markov Decision Processes (CMDP)

We consider a CMDP \( \langle S, A, P, r, \gamma, C \rangle \), with constraint set \( C = C^a \cup C^p \), where the set of signals \( C^a = \{ c_i : S \times A \to \mathbb{R} \} \) for single-step constraints only depend on the current state and action. We define the set of safe actions for a single-step constraint \( c_i \in C^a \) as \( S_{c_i}(s_t) = \{ a \in A \mid c_i(s_t, a) \leq \beta_{c_i} \} \). The set \( C^p = \{ \gamma \beta_{c_i} : S \times A \to \mathbb{R} \mid 1 \leq \beta_{c_i} \} \)
Fig. 3. In this MDP, state $s_0$ is marked as unsafe. Assume a safety check with a horizon of two time steps and that the initial state is $s_0$. Unconstrained Q-learning (red) chooses the unsafe path leading to $s_9$ with a return of +2 and Safe Policy Extraction (blue) after Q-learning leads to a safe path to state $s_{10}$ with a return of +0.5. Multi Time-scale Constrained Q-learning (green) chooses the safe path to $s_{11}$ with a return of +1.

\[ i \leq M \] consists of multi-step constraint signals $J_{i+1}$ with horizon $H_i$, which are dependent on policy $\pi$. The set of expected safe actions of a multi-step constraint $J_{i+1}$ defined in [14] to leads to more interpretable constraints. We apply the for-constraints are represented by the expected sum of the effect of the current policy of the agent for a current decision step, it can be crucial to represent the safe policy for the induced constrained MDP $C$ is defined as $E_s \{ H(s_t, a) \leq \beta_{i+1} \}_{1 \leq t \leq H}$. We define $S_i(s_t)$ as the intersection of all safe sets.

\[ C. Safe Policy Extraction \]

Given an action-value function $Q$ and a set of constraints $C$, we can extract the optimal safe policy $\pi$ w.r.t. $Q$ by $\pi(s_t) = \arg\max_{a \in S_i(s_t)} Q(s_t, a)$. We call this method Safe Policy Extraction, abbreviated by SPE.

**Proposition 1:** Given an MDP $M$ and set of constraints $C$, SPE after Q-learning is not guaranteed to give the optimal safe policy for the induced constrained MDP $M_C$.

**Proof:** Follows from Figure [3].

\[ \text{IV. MULTI-STEP CONSTRAINTS} \]

While common constraints are only dependent on the current decision step, it can be crucial to represent the effect of the current policy of the agent for a longer time scale. Typically, long-term dependencies on constraints are represented by the expected sum of discounted or average constraint signals $J_i$, i.e., $J_i(s_t, a_t) = \mathbb{E}_{a_{t+1} \sim \pi, s_{t+1} \sim P} [J(s_t, a_t)] = \mathbb{E}_{a_{t+1} \sim \pi, s_{t+1} \sim P} [\sum_{t \geq t+1} \gamma^t J_i]$. Instead, we only consider the next $H$ steps: $J_{i+1}(s_t, a_t) = \mathbb{E}_{a_{t+1} \sim \pi, s_{t+1} \sim P} [J_{i+1}(s_t)] = \mathbb{E}_{a_{t+1} \sim \pi, s_{t+1} \sim P} [\sum_{t \geq t+1} \gamma^t j_i]$. Due to the fixed horizon $H$, discounting is not needed, which leads to more interpretable constraints. We apply the formulation of truncated value-functions defined in [14] to predict the truncated constraint-values. We first estimate the immediate constraint-value and then follow a consecutive bootstrapping scheme to get to the estimation of the full horizon $H$. The update rules for constraint-values $J_i$ are:

\[ J_i(s_t, a_t) \leftarrow (1 - \alpha_{q}) J_i(s_t, a_t) + \alpha_{q} j_t \text{ and } J_i(s_t, a_t) \leftarrow (1 - \alpha_{j}) J_i(s_t, a_t) + \alpha_{j} (j_t + J_{i+1}(s_{t+1}, \arg\max_{a \in S_i(s_{t+1})} Q(s_{t+1}, a))) \]

with constraint-specific learning rate $\alpha_{q}$. To cope with infinite state-spaces, we jointly estimate $J_i(s_t, a_t) \leftarrow \{1 \leq h \leq H \}$ with function approximator $J_i(\cdot, \cdot | \theta^Q)$, parameterized by $\theta^Q$. The targets are given by $y_{i+1} = j_t$ and $y_{t+h+1} = j_t + J_{t+1}(s_{t+1}, \arg\max_{a \in S_i(s_{t+1})} Q(s_{t+1}, a) | \theta^Q) J_{t+h+1}$, where $J^*$ represent target networks. We then minimize the mean squared error between the given targets and the current prediction. The set of expected safe actions for the constraint is then defined as $S_i(s_t) = \{a \in A | J_i(s_t, a) \leq \beta_{i+1} \}$. V. MULTI TIME-SCALE CONSTRAINED Q-LEARNING

We extend the Q-learning update to use a set of constraints $C = C^* \cup C^\pi$ with corresponding safe action set $S_i(s_t)$:

\[ Q(s_t, a_t) \leftarrow (1 - \alpha) Q(s_t, a_t) + \alpha (r(s, a) + \gamma \max_{a \in S_i(s_{t+1})} Q(s_{t+1}, a)) \]

The optimal deterministic constrained policy $\pi^*$ in $\Pi_{S_i}$ is:

\[ \pi^*(s_t) = \arg\max_{a \in S_i(s_t)} Q(s_t, a) \]

**Theorem 1:** Given an MDP $M$ and set of constraints $C$, Multi Time-scale Constrained Policy Iteration (MTS-CPI) converges to the optimal deterministic policy $\pi^*$ for the induced constrained MDP $M_C$.

**Proof:** Given a set of constraints $C$ and the maximum horizon $H$ of all constraints, we can define the truncated constraint violation function $J_{i+1}$ of horizon $H$ by $J_{i+1}(s) = \mathbb{E}_{a_{t+1} \sim \pi, s_{t+1} \sim P} [\sum_{t \geq t+1} \gamma^t j_i]$. $J_{i+1}(s)$ represents the amount of constraint violations within horizon $H$ when following the current policy $\pi_k$. We can then define the complete safe set $S_i(s)$ for state $s$ under policy $\pi_k$ at iteration $k$ by $S_i(s) = \{ a | J_{i+1}(s, a) = 0 \}$. At policy improvement, the policy is updated by:

\[ \pi_{k+1} (s) \leftarrow \arg\max_{a \in S_i(s)} \sum_{s'} p(s'|s, a) (r(s, a) + \gamma V_{\pi_k}^c (s')) \]

Therefore, by definition $\pi_{k+1} (s) \in S_i(s)$. The monotonic improvement of Policy Iteration (PI) holds for MTS-CPI w.r.t. the constrained value-function $V_{\pi_k}^c (s)$:

\[ V_{\pi_k}^c (s) \leq \max_{a \in S_i(s)} Q_{\pi_k}^c (s, a) \]

\[ = \max_{a \in S_i(s)} r(s, a) + \sum_{s'} p(s'|s, a) V_{\pi_k}^c (s') \]

\[ = r(s, \pi_{k+1} (s)) + \sum_{s'} p(s'|s, \pi_{k+1} (s)) V_{\pi_k}^c (s') \]

\[ \leq r(s, \pi_{k+1} (s)) + \gamma \sum_{s'} p(s'|s, \pi_{k+1} (s)) \max_{a \in S_i(s)} Q_{\pi_k}^c (s', a) \]

\[ = V_{\pi_{k+1}}^c (s) \]

Optimality then follows from the optimality of PI [30].

In the following, we analyze the empirical performance of sampling-based Multi Time-scale Constrained Q-learning with function approximation and leave the theoretical analysis as future work. The effect of the constrained Q-update can be seen in Figure [3]. In the given MDP, state $s_6$ is marked as unsafe and has to be avoided. Vanilla Q-learning without knowledge about this constraint leads
Algorithm 1: Fixed Batch MTS-CDQN

1. initialize $Q$ and $Q'$ and set replay buffer $R$
2. initialize all multi-step constraints $\mathcal{J}$ and $\mathcal{J}'$
3. for optimization step $n=1,2,\ldots$ do
   4. sample minibatch $(s_i,a_i,s_{i+1},r_i)_{1 \leq i \leq 5}$ from $R$
   5. set target $y_i^Q = r_i + \gamma \max_{a \in S_C(s_{i+1})} Q'(s_{i+1},a|\theta'^Q)$
   6. minimize MSE of $y_i^Q$ and $Q(s_i,a|\theta)$
   7. update target network $Q'$
   8. foreach multi-step constraint $\mathcal{J}$ do
      9. set multi-step constraint targets $y_{i+h}^Q = j_i + $ \[ J_{i+1}(s_{i+1}, \arg \max_{a \in S_C(s_{i+1})} Q(s_{i+1},a|\theta'^Q))|\theta'^Q_{i-1} = \] \[ J_{i+h}(s_i, a_i|\theta'^Q) \]
     10. minimize MSE of $y_{i+h}^Q$ and $J_{i+h}(s_i, a_i|\theta'^Q)$
     11. update target networks $\mathcal{J}'$
   12. for execution step $e=1,2,\ldots$ do
     13. get current state $s_i$ from environment
     14. apply $\pi(s_i) = \arg \max_{a \in S_C(s_i)} Q(s_i,a)$

...to a policy choosing the upper path to $s_9$ with a reward of +2. A safety check at policy extraction can then be used to avoid this unsafe path, however at the point of decision it can only choose the path leading to $s_{10}$ with a non-optimal return of +0.5. Incorporating the constraint in the Q-update directly propagates the non-optimal value of the upper path back to $s_1$, such that Multi Time-scale Constrained Q-learning converges to the optimal constrained policy leading to $s_{11}$ with a return of +1.

In order to employ Multi Time-scale Constrained Q-learning within DQN, the target has to be modified to $y_i^Q = r_i + \gamma \max_{a \in S_C(s_{i+1})} Q'(s_{i+1},a|\theta'^Q)$, where $Q'$ is the target-network, parameterized by $\theta'^Q$. We refer to this algorithm as Multi Time-scale Constrained DQN (MTS-CDQN). A general description is shown in Algorithm 1.

Please note, however, that we jointly fit Q-function and multi-step constraint-values $\mathcal{J}_h$ in one function approximator, in order to minimize the number of parameters.

VI. EXPERIMENTAL SETUP

We evaluate Multi Time-scale Constrained DQN on the task of autonomous lane changes in the open-source simulator SUMO. We take the settings of SUMO as described in [23], but changed the value of $\text{lcKeepRight}$ to be in $\{5, 8, 10\}$ for the meta-configurations of the driver types, in order to enforce the drivers to keep right. For all experiments, we use the network architecture from DeepSet-Q [23] to deal with a variable number of surrounding vehicles. We estimate multi-step constraint-values and Q-values in one architecture (using multiple output heads in the last layer) to speed up training. The optimized network architecture is shown in Figure 4. All methods were trained with $2.5 \times 10^6$ gradient steps on the same fixed batch of $5 \times 10^3$ transitions, collected by a random controller which applied random lane changes to left or right whenever possible. The random controller was only forced to satisfy the safety constraint. MTS-CDQN is still capable of learning the optimal deterministic policy for the constrained MDP even if the transition set contains constraint violations.

---

![Figure 4. Architecture of Multi Time-scale Constrained DQN analogous to [23] with modified output to jointly estimate Q- and comfort-values.](image)

A. Application to Autonomous Driving

We use the state and action representation of the MDP formulation proposed in [23], where the state is represented by a list of relative distances, relative velocities, relative lane indices and vehicle lengths for all surrounding cars in sensor range. Additionally, it contains the velocity of the agent and whether lanes to the left and right of the agent are available or not. The discrete action space $\mathcal{A}$ includes three actions: keep lane, perform left lane change and perform right lane change. Acceleration and maintaining safe-distance to the preceding vehicle are controlled by a low-level SUMO controller. We use model-based control of acceleration to guarantee comfort and safety in the short-term and RL to optimize the return in long-term w.r.t. the constrained MDP, for which model-based approaches are limited in this domain. The primary objective is to drive as close as possible to a desired velocity. Thus, we define the reward function $r: S \times A \rightarrow \mathbb{R}$ as:

$$r(s,a) = r_{\text{speed}}(s,a) = 1 - \frac{|v_{\text{current}}(s) - v_{\text{desired}}(s)|}{v_{\text{desired}}(s)}$$

where $v_{\text{current}}$ and $v_{\text{desired}}$ are the actual and desired velocity of the agent. In contrast to [23], we explicitly avoid penalizing lane changes in the reward function and use a multi-step constraint to guarantee additional comfort and for increased interpretability (e.g. to perform not more than $x$ lane changes in $T$ seconds). In this work, we focus on three constraints (more constraints can be easily added in the same manner):

a) Safety: To guarantee safe driving, we use the same safety module as proposed in SUMO. We formulate the constraint signal as $c_{\text{safe}}(s,a) = 1$ if $a$ is not safe. Additionally, we restrict lane changes on the outermost lanes (it is not allowed to drive outside the lanes) by using a second constraint signal $c_{\text{lane}}(s,a) = 1_{l_{\text{max}}<0} + 1_{l_{\text{min}}\geq\text{num lanes}}$. The safe set of the safety constraint can then be formulated as $S_{\text{safe}}(s,a) = \{a \in \mathcal{A} | c_{\text{safe}}(s,a) + c_{\text{lane}}(s,a) \leq 0 \}$. The acceleration controlled by the low-level SUMO controller always satisfies this safety constraint. Thus, driving straight is always safe. In case of contradicting constraints we give safety higher priority.

b) Keep-Right: As second single-step constraint, we add a keep-right rule. The agent ought to drive right when there is a gap of at least $t_{\text{gap}}$ (we set $t_{\text{gap}}$ to 10 s in our experiments) on the same lane and on the lane right to the agent assuming driving with the desired velocity before the closest leader is reached. This rule is part of the
traffic regulations in Germany (with a time span of 20 s). We can then formulate the constraint signal as \( c_l(s, a) = 1_{a \neq \text{left}} \cdot \Delta t_{a \rightarrow \text{left}} \geq t_{gap} \) and \( \Delta t_{a \rightarrow \text{right}} \geq t_{gap} \), where \( \Delta t \) is the true gap time span. Additionally, the agent is not allowed to leave its current lane, if there is no leader on the same lane or one lane to the left, i.e. \( c_l(s, a) = 1_{a = \text{left}} \cdot \Delta t_{a \rightarrow \text{left}} \geq t_{gap} \) and \( \Delta t_{a \rightarrow \text{right}} \geq t_{gap} \). The safe set thus becomes \( S_{KR}(s, a) = \{ a \in \mathcal{A} | c_l(s, a) + c_l(s, a) \leq 1 \} \), where KR abbreviates Keep-Right.

c) Comfort: In order to guarantee comfort, we approximate a multi-step prediction of lane changes based on our target-policy. We set the immediate lane change value \( j_i \) to 1, if the agent performs a lane change and 0 otherwise. Within the defined time span, a maximum of \( \beta_{LC_{max}} \) lane changes are allowed. We calculate the amount of lane changes over \( H = 5 \) (10s) by using \( \mathcal{J}^L_c \), i.e. the safe set can be defined by \( S_{LC_{max}}(s, a) = \{ a \in \mathcal{A} | \mathcal{J}^L_c(s, a) \leq \beta_{LC_{max}} \} \). In our experiments, we set \( \beta_{LC_{max}} = 2 \). Lowering the threshold for a fixed horizon results in a more conservative behaviour, since less lane changes are allowed. Increasing the threshold adds flexibility to the behaviour of the agent, however, it will most probably lead to more lane changes. The same holds for a fixed threshold and varying horizon analogously. Furthermore, a longer horizon increases the complexity of constraint-value estimation. A hard constraint on the number of lane-changes could be avoided by an alternative formulation of the comfort multi-step constraint, where optional lane-changes are performed only if the expected velocity increases by a certain amount in a certain time. We define the immediate gain as \( j_i = v_{t+1} - v_t \) and the corresponding safe set as \( S_{VG_{min}}(s, a) = \{ a \in \mathcal{A} | \mathcal{J}^L_c(s, a) \geq \beta_{VG_{min}} \} \). We only allow additional lane-changes, if the velocity gain over \( H = 5 \) exceeds \( \beta_{VG_{min}} = 0.25 \text{ m s}^{-1} \).

B. Baselines

To highlight the advantages of MTS-CDQN, we compare to the following baselines:

a) Safe Policy Extraction (SPE): In this baseline, we check for constraints only at policy extraction.

b) Reward Shaping: We compare to a reward shaping approach, where we add weighted penalties for lane changes and for not driving on the right lane, i.e.: \( r(s, a) = r_{speed}(s, a) - \lambda_{LC_{max}} - \lambda_{KR} \cdot p_{KR} \). We set \( p_{LC} = 1 \) if action \( a \) is a lane change and 0 otherwise. Further, we set \( p_{KR} = l_{current} \) for the current lane index \( l_{current} \), where lane index 0 is the right most lane.

c) Additional Loss Terms: As an alternative, we approximate the solution of our constrained MDP using the reward \( r_{speed}(s, a) \) by including constraint penalties in the loss. We penalize the objective for constraint violations, solving the constrained surrogate:

\[
L(\theta^Q) = \frac{1}{n} \sum_{i=1}^{n} \left( y_i - Q(s_i, a_i; \theta^Q)^2 + \lambda_{safe} 1_{a_i \notin S_{safe}} + \lambda_{KR} 1_{a_i \notin S_{KR}} + \lambda_{comfort} 1_{a_i \notin S_{comfort}} Q(s_i, a_i; \theta^Q) \right)^2
\]

We multiply the constraint masks by the squared Q-values to penalize constrained violations according to their value.

The penalties for the baselines were optimized with random search using a fixed budget of \( 1.25 \cdot 10^6 \) gradient steps due to computational costs. In total, we sampled 50 configurations for each baseline. The results of the random search are shown in Figure 5, which indicates the total amount of comfort and KR constraint violations on the x-axis. MTS-CDQN is compared to DQN with reward shaping, constraint violation loss and SPE.
For both methods, configurations show either high speed in combination with a high number of constraint violations, or they violate a low number of constraints but are quite slow. This underlines the difficulty of finding proper settings in both reward shaping and Lagrangian methods.

C. Real Data

In order to evaluate the real-world applicability of our approach, we generate a transition set from the open HighD data set [15], containing 147 hours of top-down recordings of German highways, as shown in Figure 7. The data set includes features for the different vehicles, such as a distinct ID, velocity, lane and position. We discretize 5 s before and after occurrences of lane changes with a step size of 2 s, leading to a consecutive chain of 5 time steps with one lane change per chain. The acting vehicle is then considered as the current agent. In total, this results in a replay buffer of \( \sim 20000 \) transitions with \( \sim 5000 \) lane changes.

VII. RESULTS

The results for agents considering the three defined constraints can be found in Figure 6. MTS-CDQN is the only agent satisfying all constraints in every time step while taking most advantage of the maximum allowed number of lane changes, showing high speed and the lowest variance. All other agents are not able to drive close to the desired velocity or cause a tremendous amount of constraint violations. The results of reward shaping and Lagrangian optimization suffer from high variance and are not consistent. The worst performance is shown by the SPE agent, staying on the initial lane with no applied lane changes over all training runs. Thus, MTS-CDQN is by far the best performing agent, driving comfortable and fast without any violations. The comfort constraint formulation based on the change in velocity led to an equivalent behavior (data and results not shown). Since both formulations are capable of implementing the desired behavior in an interpretable manner, it is up to the task designer to choose the preferred constraint. In MTS-CDQN, the comfort predictions \( J_H \) over longer time scales can deviate from the true values. In Table I, the percentages of true comfort constraint violations are shown. The agent is violating the true comfort constraint in only 3.1\% of all decision steps on average, which is negligible. The values become smaller in dense traffic due to the increased amount of situations where the safety module does not allow for lane changes. A comparison of DQN with SPE and MTS-CDQN trained in simulation to MTS-CDQN trained on the open HighD data set [15] is shown in Figure 8. While there is a larger difference in performance for scenarios with 50 and more vehicles, the agents trained in simulation and the real data perform equivalently for scenarios with 20 to 40 vehicles. Furthermore, MTS-CDQN trained on real data outperforms DQN with SPE trained directly in simulation, which is not capable to solve the task adequately while satisfying all constraints in all time steps. The learned policy of MTS-CDQN generalizes to new scenarios and settings, even with mismatches between simulation and the real recordings.

VIII. CONCLUSION

We introduced Multi Time-scale Constrained Q-learning, an approach to incorporate hard constraints directly in the Q-update to find the optimal deterministic policy for the induced constrained MDP. For its formulation, we define a new class of multi-step constraints based on truncated value-functions. In high-level decision making for autonomous driving, MTS-CDQN is outperforming reward shaping, Lagrangian optimization and Safe Policy Extraction in terms of final performance and constraint violations, while offering more interpretable constraint formulations. MTS-CDQN can learn a policy satisfying traffic-rules directly from real transitions without the need of simulated environments, which is a major step towards the application to real systems.
REFERENCES

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alexander Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharsan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[2] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vassar Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine de Freitas, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[3] Manuel Watter, Jost Tobias Springenberg, Jostchka Boedecker, and Martin A. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In Corrinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2746–2754, 2015.

[4] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–39, 2016.

[5] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *CoRR*, abs/1207.4078, 2012.

[6] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In 2012 IEEE RSJ International Conference on Intelligent Robots and Systems, pages 5026–5033, Oct 2012.

[7] Shie Mannor, Nahum Shimkin, and Sridhar Mahadevan. A geometric approach to multi-criterion reinforcement learning. *J Mach. Learning Res.*, 5, 04 2004.

[8] Matteo Pirotta, Simone Parisi, and Marcello Restelli. Multi-objective reinforcement learning with continuous pareto frontier approximation. In *AAAI*, 2014.

[9] T. Brys, A. Harutyunyan, P. Vranck, M. E. Taylor, D. Kudenko, and A. Nowe. Multi-optimization of reinforcement learning problems by reward shaping. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 2315–2322, July 2014.

[10] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, pages 279–292, 1992.

[11] Branka Mirchevska, Christian Pek, Moritz Welring, Matthias Althoff, and Jostchka Boedecker. High-level decision making for safe and reasonable autonomous lane changing using reinforcement learning. *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2118–2125, 2018.

[12] Mustafa Mukadam, Akansel Cosgun, Alireza Nakhaei, and Kikuo Fujimura. Tactical decision making for lane changing with deep reinforcement learning. *NIPS Workshop on Machine Learning for Intelligent Transportation Systems*, 2017.

[13] Eitan Altman. *Constrained Markov decision processes*, volume 7. CRC Press, 1999.

[14] Gabriel Kalweit, Maria Huegle, and Jostchka Boedecker. Off-policy multi-step q-learning. *CoRR*, abs/1909.13518, 2019.

[15] Robert Krajewski, Julian Bock, Laurent Kloeker, and Lutz Eckstein. The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2118–2125, 2018.

[16] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 22–31. JMLR. org, 2017.

[17] D.P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[18] Va Borkar. An actor-critic algorithm for constrained markov decision processes. *Systems & Control Letters*, 54:207–213, 03 2005.

[19] Shalabh Bhatnagar and K. Lakshmanan. An online actor–critic algorithm with function approximation for constrained markov decision processes. *Journal of Optimization Theory and Applications*, 153(3):688–708, Jun 2012.

[20] Chen Tessler, Daniel J. Mankowitz, and Shie Mannor. Reward constrained policy optimization. *CoRR*, abs/1805.11074, 2018.

[21] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 2052–2062. PMLR, 2019.

[22] Branka Mirchevska, Manuel Blum, Lawrence Louis, Jostchka Boedecker, and Moritz Welring. Reinforcement learning for autonomous maneuvering in highway scenarios. In *Workshop for Driving Assistance Systems and Autonomous Driving*, pages 32–41, 2017.

[23] Maria Huegle, Gabriel Kalweit, Branka Mirchevska, Moritz Welring, and Jostchka Boedecker. Dynamic input for deep reinforcement learning in autonomous driving. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2019, Macau, SAR, China, November 3-8, 2019*, pages 7566–7573. IEEE, 2019.

[24] Maria Huegle, Gabriel Kalweit, Moritz Welring, and Jostchka Boedecker. Dynamic interaction-aware scene understanding for reinforcement learning in autonomous driving. To appear in Proc. of the *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[25] Huanjie Wang, Shihua Yuan, Mengyu Guo, Ching-Yao Chan, Xueyuan Li, and Wei Lan. Tactical driving decisions of unmanned ground vehicles in complex highway environments: A deep reinforcement learning approach. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automotive Engineering*, 0(0):0954407019898009, 0.

[26] Andrea Bajcsy, Sylvia L. Herbert, David Fridovich-Keil, Jaime F. Fisac, Sampada Deglurkar, Anca D. Dragan, and Claire J. Tomlin. A scalable framework for real-time multi-robot, multi-human collision avoidance. In *International Conference on Robotics and Automation, ICRA 2019*, Montreal, QC, Canada, May 20-24, 2019, pages 936–943. IEEE, 2019.

[27] David Fridovich-Keil, Jaime F. Fisac, and Claire J. Tomlin. Safely probabilistically complete real-time planning and exploration in unknown environments. In *International Conference on Robotics and Automation, ICRA 2019*, Montreal, QC, Canada, May 20-24, 2019, pages 7470–7476. IEEE, 2019.

[28] Richard Cheng, Gábor Orosz, Richard M. Murray, and Joel W. Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 3387–3395. AAAI Press, 2019.

[29] Mikael Henaff, Alfredo Canziani, and Yann LeCun. Model-predictive reinforcement learning with uncertainty regularization for driving in dense traffic. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[30] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994.