2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)
October 8-12, 2022, Macau, China

A Modern Perspective on Safe Automated Driving for Different Traffic Dynamics using Constrained Reinforcement Learning

Danial Kamran*[1], Thiago D. Simão*[2], Qisong Yang[3],
Canmanie T. Ponnambalam[3], Johannes Fischer[1], Matthijs T. J. Spaan[3] and Martin Lauer[1]

2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC) | 978-1-6654-6880-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/ITSC55140.2022.9921907

***Abstract*— The use of reinforcement learning (RL) in real-world domains often requires extensive effort to ensure safe behavior. While this compromises the autonomy of the system, it might still be too risky to allow a learning agent to freely explore its environment. These strict impositions come at the cost of flexibility and applying them often relies on complex parameters and hard-coded knowledge modelled by the reward function. Autonomous driving is one such domain that could greatly benefit from more efficient and verifiable methods for safe automation. We propose to approach the automated driving problem using constrained RL, a method that automates the trade off between risk and utility, thereby significantly reducing the burden on the designer. We first show that an engineered reward function for ensuring safety and utility in one specific environment might not result in the optimal behavior when traffic dynamics changes in the exact environment. Next we show how algorithms based on constrained RL which are more robust to the environmental disturbances can address this challenge. These algorithms use a simple and easy to interpret reward and cost function, and are able to maintain both, efficiency and safety without requiring reward parameter tuning. We demonstrate our approach in the automated merging scenario with different traffic configurations such as low or high chance of cooperative drivers and different cooperative driving strategies.**

I. Introduction

Reinforcement learning (RL) promises to produce agents that learn to optimize decision-making problems with limited to no knowledge of the environment. This makes it an attractive approach to automating complex and high-dimensional tasks. The drawback is that RL agents must interact with the environment, exhaustively taking both good and bad actions, in order to learn the best long-term decisions. In real-world and safety-critical domains such as driving, where the consequences of taking bad actions are severe, which diminishes the appeal of classical RL. An ideal response to this problem is safe reinforcement learning, a class of RL methods that guarantees safety during learning or upon execution. Safe RL methods have recently been applied to various automated driving problems with some success [1]–[6]. Existing safe RL approaches to autonomous driving require extensive designer knowledge coded into the solution. In many cases, these methods impose a heavy burden on the designer to identify unsafe states or actions, tune hyperparameters, or define complex reward functions [3], [5], [7], as Figure 1 illustrates. In addition to the issue of extensive prior knowledge needed, these methods can be overly conservative as they often impose hard restrictions on the search space. Instead, we propose *constrained reinforcement learning* as an elegant approach to safety in the automated vehicle domain. Constrained RL models the problem as a constrained Markov decision process (CMDP), introducing a cost function to encode safety-relevant information (such as whether a crash has occurred). This clearly separates the specification of reward (to be maximized) and cost (to be minimized). The constraint is then defined as a threshold regarding the acceptable expected cost, resulting in a simple and highly interpretable parameter. The agent learns to optimize reward while respecting this safety constraint, automatically tuning the trade-off between the two conflicting goals.

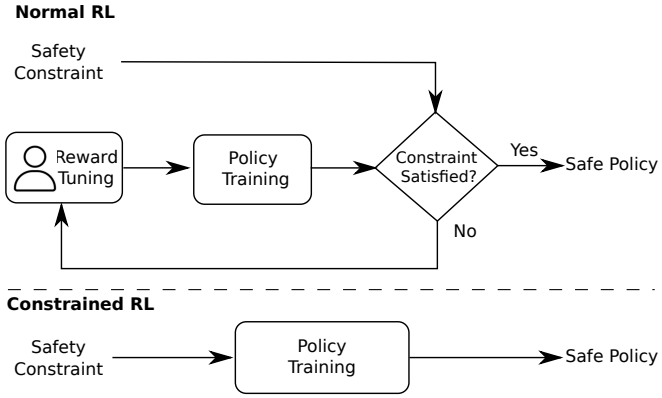

Fig. 1. Two structures for learning safe policies. In normal RL, the user searches for the best reward function that produces a policy that satisfies the required constraint. In constrained RL, the algorithm design is simplified as the safety constraint will automatically be satisfied during training.

Merging into a highway with dense traffic is a challenging task for automated vehicles. The dense traffic means that the window in which a successful merge can occur is small. The position and interaction between other vehicles in the environment is a crucial aspect of the state description that determines when a merge can be successfully executed. This complex state space makes it particularly challenging to define a set of safe states or actions by hand, and manipulating the reward function to include safety considerations is difficult and requires considerable tuning. Further, applying a safe RL method that is overly conservative in this scenario

*Authors have equal contribution.
[1]Karlsruhe Institute of Technology, Germany {danial.kamran, johannes.fischer, martin.lauer}@kit.edu
[2]Radboud University, Nijmegen, The Netherlands thiago.simao@ru.nl
[3]Delft University of Technology, The Netherlands {q.yang, c.t.ponnambalam, m.t.j.spaan}@tudelft.nl

can result in the *freezing robot problem*, whereby the vehicle is unable to merge at all. This makes the highway merge problem a prime candidate for a constrained RL approach.

In this paper, we first describe existing safe RL approaches to the automated driving problem, and highlight recent work with similar goals to our approach as well as their limitations. We then formulate the dense highway merge scenario as a constrained MDP and apply two constrained RL methods to a traditional safe approach to this problem. The experiments demonstrate how constrained RL successfully mitigates the trade-off between merging as quickly as possible and avoiding crashes without additional hyper-parameters or extensive tuning.

II. Related Work

The field of safe reinforcement learning (RL) encompasses several different types of approaches with varying levels of safety guarantees, of which formulating the problem as a constrained Markov decision process (as we propose) is only a sub-set. For a comprehensive overview of safe RL in general, we refer the reader to [8].

In this work, we focus on methods situated in the automated driving domain that aim to adhere to safe behavior either during learning or on execution of the trained agent. The most relevant methods can be divided into two categories: those that encode safety in the reward function, and those that shield unsafe actions from the agent.

A popular approach to safe RL is to include penalties in the reward function that discourages unsafe behavior, an indirect way to incorporate safety [1], [5], [6], [9]. These methods lay the burden of specifying unsafe behavior on the designer, resulting in reward functions that can be hard to specify and even more difficult to verify. A more explicit way to produce safe behavior is to restrict the action space to safe actions, often referred to as *shielding* [10]–[12]. Determining unsafe actions can be done using, for example, a model checker [3] or predictive model [4], [7]. Efforts have been made to combine the two types of approaches, with one using a parameterized reward penalty to restrict actions determined to be unsafe [13]. In general, restricting the search space can produce conservative behavior, as they enforce hard limits on the space of acceptable policies. Further, such methods are very sensitive to incorrect specifications or predictions of unsafe actions.

The use of constrained RL in the autonomous vehicle domain has been so far limited. In one paper, they used LTL specifications to define unsafe states, referring to the result as a constrained optimization problem [14]. However, this decoupled approach does not attempt to balance reward and safety, instead enforcing hard constraints on the search space. A budgeted MDP, which is similar to a constrained MDP, but offers additional control over the budget, has also been used to model the problem of automated driving [15]. Most recently, constrained RL has been evaluated on lane keeping and intersection navigation tasks, where a parallel learning approach was proposed that employs multiple agents to speed up convergence [16]. Our paper highlights the limitations of reinforcement learning that are addressed by taking a constrained optimization approach. We focus on the improvements provided in terms of the ease of specification, robustness to scalarization issues, and elegant trade-off of reward and risk, evaluated on a dense highway merge scenario.

III. Background

In this section, we formalize the definition of constrained RL, and present the algorithms that are used to solve it.

A. Constrained Markov Decision Process

A CMDP [17] is a model that separates reward and safety signals. Similar to a Markov decision process (MDP) [18], a CMDP is a tuple, $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \iota, r, c, d, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the set of actions, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a transition kernel indicating the probability to state s' after taking action a in state s, ι is the initial state distribution, $r : \mathcal{S} \times \mathcal{A} \to [r_{min}, r_{max}]$ is the reward function, $c : \mathcal{S} \times \mathcal{A} \to [c_{min}, c_{max}]$ is the cost function, d is the safety threshold, and $\gamma \in [0, 1]$ is the discount factor.

As in an MDP the goal in a CMDP is to compute a policy that maximizes the accumulated discounted reward

$$\max_{\pi} J^R(\pi) \doteq \mathop{\mathbb{E}}_{(s_t, a_t) \sim \mathcal{T}_\pi} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]. \tag{1}$$

where $\mathcal{T}_\pi = (s_0, a_0, s_1, \ldots)$ is the trajectory distribution induced by $s_0 \sim \iota$, $a_t \sim \pi(\cdot|s_t)$, and $s_{t+1} \sim \mathcal{P}(\,\cdot\,|s_t, a_t)$. Additionally, the optimal policy has to keep the expected accumulated discounted cost bounded

$$J^C(\pi) \doteq \mathop{\mathbb{E}}_{(s_t, a_t) \sim \mathcal{T}_\pi} \left[\sum_{t=0}^{\infty} \gamma^t c(s_t, a_t) \right] \leq d \tag{2}$$

according to the predefined safety threshold d. Depending on the task, it might resemble a bound on the *probability of failure*, for instance if $c(s, a) = \mathbb{1}_{\text{failure}}(s)$, although this requires $\gamma = 1$. An MDP can be seen as an unbounded CMDP, setting $d = \infty$, which essentially allows to ignore the cost function, obtaining the following MDP $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \iota, r, \gamma)$.

B. Constrained Reinforcement Learning

Constrained RL addresses the problem of solving an unknown CMDP [19]. Although off-policy methods for constrained RL have been proposed [20]–[22], this paper focuses on on-policy variants. Specifically, we apply PPO-Lagrangian [23] and Constrained Policy Optimization (CPO) [24] to the automated driving domain. These methods represent two main directions in on-policy constrained RL. The first direction is to adapt RL algorithms to their Lagrangian variants, as seen in TRPO-Lagrangian and PPO-Lagrangian [23]. The second direction uses constrained policy optimization methods [25], [26] built on the work of [24].

a) PPO-Lagrangian: Proximal policy optimization (PPO) [27], designed for regular RL problems, not only retains the benefits of trust region policy optimization (TRPO) [28], but also has better sample complexity and convenience to implement. Constrained optimization problems can be solved by a Lagrangian variant of PPO [23], [29]. Instead of fixing the value of the Lagrangian multiplier, we adapt it based on the constraint-satisfying performance. When the policy is unsafe, we increase the Lagrangian multiplier to enhance safety, but decrease it when attaining safe performance. This allows us to leverage an adaptive safety weight λ in the constrained optimization problem:

$$\max_{\pi} \min_{\lambda \geq 0} \mathcal{G}(\pi, \lambda) \doteq f(\pi) - \lambda g(\pi), \tag{3}$$

where $f(\pi) = J^R(\pi)$ and $g(\pi) = J^C(\pi) - d$ in the case of Equations (1) and (2). So, we update the safety weight using

$$\lambda_{k+1} = \max(0, \lambda_k + \alpha_\lambda (J^C(\pi) - d)), \tag{4}$$

where α_λ is the penalty learning rate. In our experiments, we use the undiscounted cumulative cost to measure the real constraint satisfaction.

b) Constrained Policy Optimization (CPO): CPO is a trust-region method for constrained RL with guarantees for near-constraint satisfaction at each iteration [24]. At each gradient step, CPO constrains the policy changes to the cost constraint and divergence neighborhood while guaranteeing reward improvement. Similar to TRPO, Equations (1) and (2) are further constrained to an additional Kullback-Leibler (KL) divergence constraint:

$$\begin{aligned} \pi_{k+1} &= \arg\max_{\pi} \mathop{\mathbb{E}}_{\substack{s\sim\mathcal{T}_{\pi_k} \\ a\sim\pi}} [A_R^{\pi_k}(s,a)] \\ \text{s.t.} \quad & J^C(\pi_k) + \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{\substack{s\sim\mathcal{T}_{\pi_k} \\ a\sim\pi}} [A_C^{\pi_k}(s,a)] \leq d \\ & \mathop{\mathbb{E}}_{s\sim\mathcal{T}_{\pi_k}} [D_{KL}(\pi || \pi_k)[s]] \leq \delta \end{aligned} \tag{5}$$

where δ is the maximum step size, D_{KL} is the KL divergence to indicate the trust region. The advantage functions A_R and A_C, respectively, express the performance change $\mathbb{E}_{s\sim\mathcal{T}_{\pi_k}, a\sim\pi}[A_R^{\pi_k}(s,a)]$ (in reward) and $\mathbb{E}_{s\sim\mathcal{T}_{\pi_k}, a\sim\pi}[A_C^{\pi_k}(s,a)]$ (in cost) of policy π over the current policy π_k. After the transition from Equations (1) and (2) to Equation (5), CPO further approximates the reward function and constraints using linear approximation (first and second order expansions) for small step sizes δ, to ensure the problem is solvable. We refer the reader to [24] for more details on the CPO algorithm.

IV. Autonomous Driving as an MDP

We formulate the automated driving problem as a Markov decision process (MDP), where at every decision step t, the decision making policy π chooses the best action. The overall goal is to learn the actions that maximize the expected future reward (return) at every time step. In this paper, we focus on merging in a highway environment where the ego vehicle is a reinforcement learning agent that observes the positions and velocities of the surrounding vehicles and controls its acceleration. The aim is to avoid collisions during merging without acting too conservative. To this end, we model the merging scenario depicted in Figure 2 and define the input state as

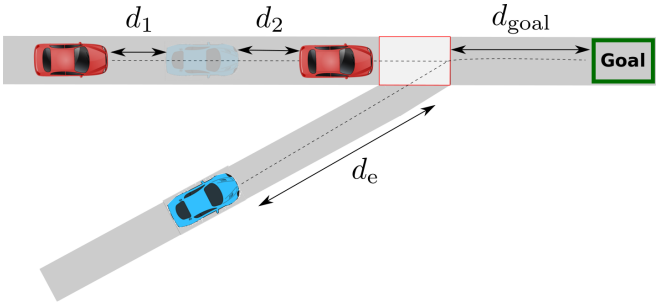

Fig. 2. Example of a merging scenario and the features that make up the observation of the reinforcement learning agent. Here the ego vehicle (blue) has to prevent collisions with vehicles on the main lane and also drive as fast as possible to reach the goal.

$$s_t = \begin{bmatrix} d_{\text{e}} & d_{\text{goal}} & d_1 & \dots & d_n \\ v_{\text{e}} & a_{\text{e}} & v_1 & \dots & v_n \end{bmatrix}, \tag{6}$$

where d_{e} is the ego vehicle's distance to the conflict merging area, d_{goal} is the distance from the conflict area to the goal, and v_{e} and a_{e} are the velocity and acceleration of the ego vehicle, respectively. We also include relative distances and velocities between vehicles on the main lane and the projection of the ego vehicle to the main lane as d_{i} and v_{i} for a maximum of $N = 15$ surrounding vehicles in the state, as shown in Figure 2.

The policy maps a state to an action a_t from the discrete action space $\mathcal{A} = \{\text{Decelerate}, \text{Idle}, \text{Accelerate}\}$ that controls the ego vehicle behavior during merging by sending high-level commands to a low-level speed controller.

Some of the key performance metrics we consider in this domain include:

$$\text{risk}(s_t, a_t) = \begin{cases} c_{\text{collision}}, & \text{if collision,} \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

$$\text{utility}(s_t, a_t) = \begin{cases} c_{\text{success}}, & \text{if success,} \\ -c_{\text{time}}, & \text{otherwise.} \end{cases} \tag{8}$$

In some works, the time penalty c_{time} is not used. Instead, discounting future rewards with $\gamma < 1$ also encourages faster driving.

In order to learn the desired behavior, both safety and utility must be considered, as the two most important aspects for automated driving. It is preferred to learn policies that are safe, thus preventing collisions with other vehicles, while also acting efficiently, thereby exhibiting behavior that is not too conservative.

A. Penalty-based safety

Traditionally, such desired behavior is encoded in the reward function using a combination of these components and adjusting their parameters to increase speed or enforcing

time penalties that encourage faster driving, while at the same time employing high collision penalties to encourage safe driving [1]–[6]. The resulting reward function is given as

$$r(s_t, a_t) = \text{ utility}(s_t, a_t) - \lambda \text{ risk}(s_t, a_t), \tag{9}$$

where λ is the safety weight, which is responsible to balance between utility and safety.

In this case, assuming the values $c_{\text{collision}}$, c_{time} and c_{success} are already defined, the user must choose an appropriate safety weight λ. Notice however, that the appropriate value for λ depends on the structure of the reward function, in other words, for different values of $c_{\text{collision}}$, c_{time} and c_{success}, the appropriate safety weight λ could change significantly.

V. Autonomous Driving as a CMDP

In order to overcome the hyper-parameter sensitivity of such a complex reward function, which is especially important in safety-related applications like automated driving, we propose to instead use constrained RL and formulate safety explicitly in a cost function. In this way, the RL agent automatically satisfies safety constraints identified as cost limits of the policy without requiring any parameter tuning in the reward function. We define the following reward and cost function for our highway merging scenario

$$r(s_t, a_t) = \text{utility}(s_t, a_t), \tag{10}$$

$$c(s_t, a_t) = \text{risk}(s_t, a_t). \tag{11}$$

Now, when the user of this system trains a policy to drive safely, she only needs to define the cost limit d, removing the burden of choosing an appropriate balance between utility and risk, represented by the value λ.

a) The trade-off between safety and utility: In a conventional reward scheme, safety and utility are considered simultaneously in the return, implying that at some points the RL agent may sacrifice safety to reach higher reward or alternatively become too conservative due to large safety punishments. This trade-off is often tuned based on the $\frac{c_{\text{collision}}}{c_{\text{time}}}$ or $\frac{c_{\text{collision}}}{c_{\text{success}}}$ ratio in the reward function. However, this leads to two main issues: hyper-parameter sensitivity and environment over-fitting. After small changes in the environment configuration (like more dense traffic or higher average speed of vehicles) the reward function may not lead to the desired behavior anymore and a new reward parameter tuning needs be applied and the agent needs to be retrained with a new rewarding scheme.

We propose to consider safety as the cost of policy and decouple it from other factors in the desired behavior of the RL agent by leveraging constrained RL. We can then enforce safety by setting a suitable cost limit d which is a meaningful parameter, in our case specifying the average number of safety violations of the policy, without the requirement of again tuning the parameters of the reward function.

We may notice that other objectives, such as comfort, compliance with traffic rules or fuel consumption, could also be defined as separate constraints. This makes the goals easier to interpret and avoids a highly complex scalarized reward function.

Reward engineering can also become easier with this approach. Consider for instance the task of choosing the values for c_{time} and c_{success}. On the one hand, if $c_{\text{time}} > c_{\text{success}}$, a regular RL agent might choose to crash in order to avoid getting time penalties, ignoring the reward for completing a task. On the other hand, a constrained RL would still have a reasonable behavior due to the safety constraints.

VI. Experimental Analysis

In this section we empirically evaluate the two ways we may tackle the automated driving task with normal reinforcement learning and constrained reinforcement learning. The goal is to validate the hypothesis that training a safe and high performing policy using constrained RL is easier from the user perspective than using regular RL.

A. Experimental Set-Up

For our evaluations we use the highway-env framework, which provides environments for tactical decision-making in different automated driving tasks [30]. In this framework, the RL agent controls the ego vehicle, while the other vehicles follow an Intelligent Driver Model (IDM) [31] and only react to the ego vehicle once it enters their lane. At the beginning of each episode, some vehicles with probability of p_{coop} will be cooperative which consider the projected position of the ego-vehicle on their lane as the front vehicle position and open a merging gap for the ego vehicle with different comfortable deceleration limit in their IDM controller ($a_{\text{comf-max}}$). In order to simulate different traffic dynamics, we implement three different environments for the automated merging scenario:

- Low Cooperative: Low chance of having cooperative drivers with p_{coop}=0.3 and early cooperative brake with $a_{\text{comf-max}}$=1.0 m/s^2
- High Cooperative: High chance of having cooperative drivers with p_{coop}=0.6 and early cooperative brake with $a_{\text{comf-max}}$=1.0 m/s^2
- Low Cooperative with Late Brake: Low chance of having cooperative drivers with p_{coop}=0.3 and late cooperative brake with $a_{\text{comf-max}}$=5.0 m/s^2

1) Baselines: We consider three different RL agents in order to solve each merging scenario: Normal PPO with multiple collision penalties, PPO-Lagrangian with multiple cost limits (we set $\alpha_\lambda = 0.05$ and update the penalty 40 times per epoch; the remaining hyperparameters are the same as for PPO), and CPO also using multiple cost limits.

2) Metrics: We evaluate the average episode cost (AverageEpCost), which is the expected accumulated cost of a trajectory and the episode length (EpLen), which indicates how fast the policy can finish one episode. Hence, in all the plots lower values are better.

B. Results

1) Ease of use: As we discussed in Section IV and Section V, finding the appropriate value to balance between utility and safety can be a challenge. Figure 3 shows the performance of the different algorithms on the *Low Cooperative*

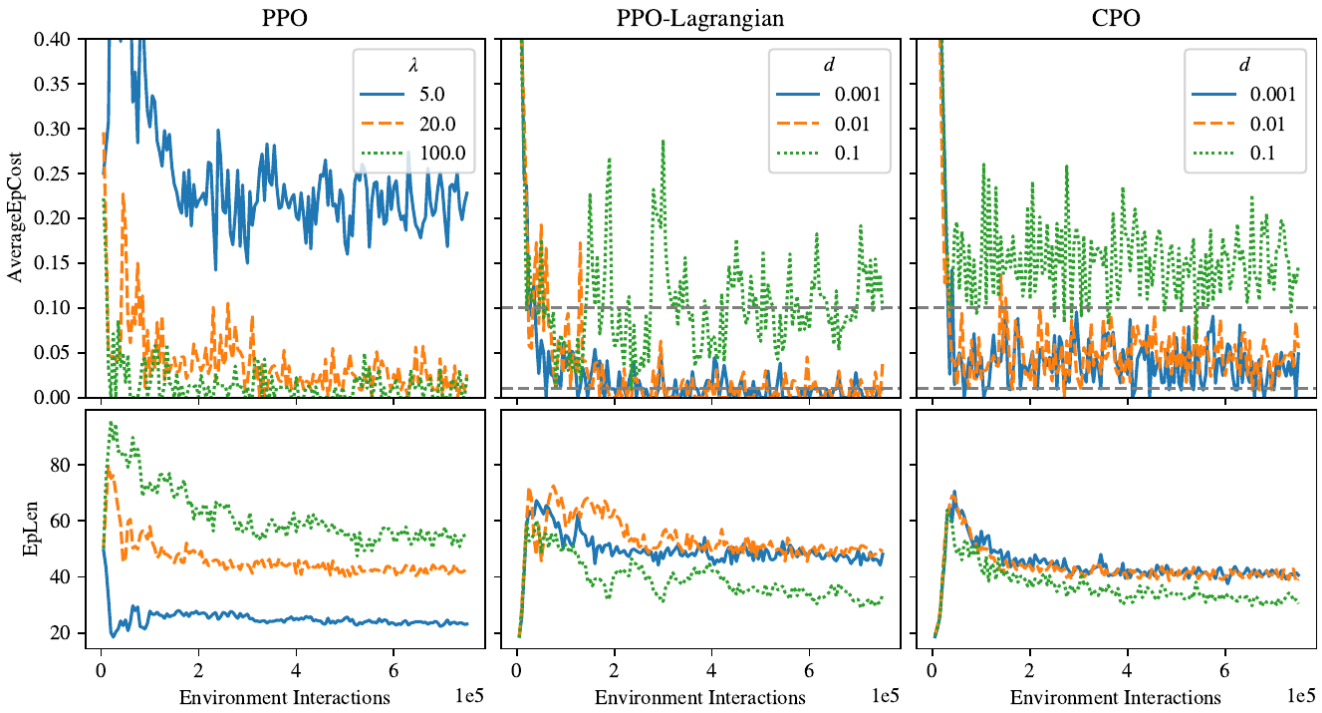

Fig. 3. Training results for PPO with different safety weights (left), PPO-Lagrangian with different cost limits (middle) and CPO algorithm with different cost limits (right). The grey dashed lines indicate the cost limits.

environment using different values for the safety weight λ (in case of PPO) and safety bound d (in case of PPO-Lagrangian and CPO). On the one hand, we notice that equipping PPO with a λ too low, such as 5, can lead to extremely unsafe policies, while setting it to 20 or 100 provide safer policies. On the other hand, the constrained methods can find safe policies. It is easy to see that PPO-Lagrangian is approaching the desired safety bound. This experiment makes clear that from the user's perspective choosing a value for d is much more meaningful than choosing a value for λ, since there is an obvious connection between the safety bound d and the safety level of the policy returned. Consider for instance that the user is willing to allow an expected cost of 0.1, after observing the results from PPO with 3 different values for λ on Figure 3, it is not clear what should be the value of λ in that case.

Considering the results for PPO with $\lambda = 5$, we may conclude that setting $c_{\text{time}} = 0.1$ and $c_{\text{success}} = 1$ encourages the agent to terminate the episode as soon as possible leading to more crashes, making it mandatory to set $\lambda = 5$. On the other hand, we notice that the constrained agents manage to reduce the number of collisions, almost independently of the cost limit.

2) Safety satisfaction: Although both, PPO-Lagrangian and CPO, try to learn safe policies, according to Figure 3, PPO-Lagrangian is more successful to satisfy the specified cost limit d in its configuration. This suggests that based on the desired safety requirement, one can directly specify the required cost limit for a PPO-Lagrangian agent before training without the necessity to tune the reward function for safety satisfaction.

3) Evaluations on Different Traffic Dynamics: In traditional RL, the reward function needs to be specialized for every new environment with different configuration. In order to study if constrained RL can address this challenge, we trained PPO agents with different collision penalties (λ) in their reward function in environments with different traffic dynamics. After training, we evaluated each trained policy for 100 episodes in the configured environment and compared the collision rate and average episode time of each agent in Table I and Table II. The first conclusion from these results is that the PPO agent requires specialized collision penalty in order to learn safe behavior for each environment. For the High Cooperative environment, all PPO agents have collision rates below 5% while for the Low Cooperative agent the PPO with λ=0.1 has 14% collision rate. Moreover, in the Low Cooperative with Late Brake environment (as the most challenging configuration) only PPO agents with $\lambda \geq 5$ have collision rates below 5%. Next we trained the PPO-Lagrangian as a constrained RL agent in the three environments and compared its evaluations with the PPO agent. It is visible that the PPO-Lagrangian agent could learn policies with collision rates below 5% for all of the three environments with fixed parameters in the reward and cost functions (d=0.01 and α_λ=0.1). The important conclusion is that the PPO-Lagrangian algorithm is not sensitive to the

TABLE I

COMPARING BASELINES IN LOW COOPERATIVE AND HIGH COOPERATIVE ENVIRONMENTS.

Env. Config	Low Cooperative							High Cooperative						
Agent	PPO						PPO-Lag	PPO						PPO-Lag
λ	0.1	1	2.5	5	10	100		0.1	1	2.5	5	10	100	
Collision Rate (%)	14	3	4.5	0.6	2.6	**0**	3.3	4.6	4.3	1	4.3	2	**0**	0.33
Avg. Time (s)	**48.2**	51.1	52.7	59.9	56.8	79.3	56.7	47.5	**47.3**	49.4	48.1	49.1	60.3	49.1

TABLE II

COMPARING BASELINES IN LATE COOPERATIVE BRAKE ENVIRONMENT.

Env. Config	Low Cooperative with Late Brake						
Agent	PPO						PPO-Lag
λ	0.1	1	2.5	5	10	100	
Collision Rate (%)	16	14	19.3	3.3	0.1	**0**	1.3
Avg. Time (s)	47.4	**45**	47.5	60.6	68.8	78.2	81.9

environment disturbances and therefore the designer can put less effort for training safe RL policies in automated driving environments which may have different traffic configurations.

4) Effect of penalty learning rate α_λ: We also performed a hyper-parameter analysis for the PPO-Lagrangian's penalty learning rate α_λ. We considered the values $\alpha_\lambda \in [0.005, 0.01, 0.05, 0.1, 0.5]$. Figure 4 shows that overall the PPO-Lagrangian algorithm has a low hyper-parameter sensitivity with respect to α_λ in terms of safety. That is, for all learning rates the algorithm is converging to a constraint satisfying policy. We also notice that, α_λ has a more significant impact in the performance during learning, demonstrated by the average episode length. PPO-Lagrangian finds policies with lower episode length using lower learning rates (0.01 and 0.005), indicating that the results presented on Figure 3 and Table I and Table II could still be improved, while larger learning rates can increase the episode length.

5) Video Demonstration: The supplementary video compares the use of different safety penalties for a penalty-based RL model and constrained RL algorithms. Large penalties generally lead RL agents to drive safely but overly conservative (slower), while smaller penalties lead to faster driving but also reckless behavior. We observe that the Lagrangian-PPO agent learns to balance between safe and fast driving, a behavior less conservative than an RL agent with large penalties and also less reckless than an RL agent with small penalties.

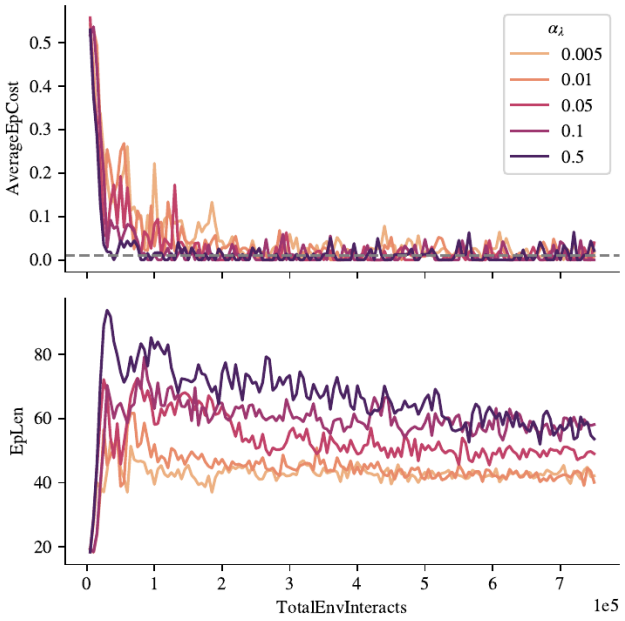

Fig. 4. Training results for the PPO-Lagrangian method using different penalty learning rates on the *Low Cooperative* environment with cost limit $d = 0.01$, as indicated by the grey dashed line.

C. Limitations

Notice that constrained RL defines safety in expectation, which in our application still allows a number of collisions even after the learning is finished, to mitigate this issues we could combine such an approach with methods that enforce hard constraints [10]–[12]. This method also does not guarantee safety while the agent is still learning, finding ways to ensure constrained RL satisfies the safety constraints is an active line of research [32]–[34]. Furthermore, investigating more sophisticated penalty learning schedules can be applied in the constrained RL algorithm in order to achieve faster convergence and even more adaptive policies.

VII. CONCLUSION

In this paper, we addressed the challenge of learning safe and efficient policies in automated driving with RL. In contrast to traditional RL methods that learn safe policies by discouraging unsafe outcomes using penalties in the reward function, we investigate a new perspective on safety of the learned policies using constrained RL. We showed the main drawback of the traditional RL algorithms is the requirement of reward engineering for every specific traffic configurations (e.g. fewer cooperative drivers or different cooperative strategies) in order to learn safe policies. The proposed methodology provides a clear interface for the designer who only needs to set the desired cost limit for the policy being learned instead of manually balancing safety and utility until finding the best policy. In light of our

experiments, this helps to learn safe and efficient policies in environments with different traffic dynamics using a fixed setup for the constrained RL agent.

ACKNOWLEDGMENT

This research is partly accomplished within the project "UNICARagil" (FKZ 6EMO0287). We acknowledge the financial support for the project by the Federal Ministry of Education and Research of Germany (BMBF).

REFERENCES

[1] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura, "Navigating occluded intersections with autonomous vehicles using deep reinforcement learning," in *ICRA*, IEEE, 2018, pp. 2034–2039.

[2] T. Tram, A. Jansson, R. Grönberg, M. Ali, and J. Sjöberg, "Learning negotiating behavior between cars in intersections using deep q-learning," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2018, pp. 3169–3174.

[3] M. Bouton, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer, "Safe reinforcement learning with scene decomposition for navigating complex urban environments," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2019, pp. 1469–1476.

[4] M. Bouton, A. Nakhaei, D. Isele, K. Fujimura, and M. J. Kochenderfer, "Reinforcement learning with iterative reasoning for merging in dense traffic," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020, pp. 1–6.

[5] D. Kamran, C. F. Lopez, M. Lauer, and C. Stiller, "Risk-aware high-level decisions for automated driving at occluded intersections with reinforcement learning," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2020, pp. 1205–1212.

[6] M. Bouton, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer, "Cooperation-aware reinforcement learning for merging in dense traffic," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2019, pp. 3441–3447.

[7] D. Isele, A. Nakhaei, and K. Fujimura, "Safe reinforcement learning on autonomous vehicles," in *IROS*, IEEE, 2018, pp. 1–6.

[8] J. García and F. Fernández, "A comprehensive survey on safe reinforcement learning," *JMLR*, vol. 16, pp. 1437–1480, 2015.

[9] P. Wang, C.-Y. Chan, and A. de La Fortelle, "A reinforcement learning based approach for automated lane change maneuvers," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 1379–1384.

[10] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *AAAI*, AAAI Press, 2018, pp. 2669–2678.

[11] G. Kalweit, M. Huegle, M. Werling, and J. Boedecker, *Deep Constrained Q-learning*, arXiv:2003.09398, 2020.

[12] N. Jansen, B. Könighofer, S. Junges, A. Serban, and R. Bloem, "Safe reinforcement learning using probabilistic shields," in *CONCUR*, ser. LIPIcs, vol. 171, 2020, 3:1–3:16.

[13] S. Mo, X. Pei, and C. Wu, "Safe reinforcement learning for autonomous vehicle using monte carlo tree search," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–8, 2021.

[14] M. Bouton, J. Karlsson, A. Nakhaei, K. Fujimura, M. J. Kochenderfer, and J. Tumova, "Reinforcement learning with probabilistic guarantees for autonomous driving," 2019, arXiv:1904.07189.

[15] N. Carrara, E. Leurent, R. Laroche, T. Urvoy, O.-A. Maillard, and O. Pietquin, "Budgeted reinforcement learning in continuous state space," in *NeurIPS*, Curran Associates, Inc., 2019, pp. 9295–9305.

[16] L. Wen, J. Duan, S. E. Li, S. Xu, and H. Peng, "Safe reinforcement learning for autonomous vehicles through parallel constrained policy optimization," in *23rd IEEE International Conference on Intelligent Transportation Systems*, IEEE, 2020, pp. 1–7.

[17] E. Altman, *Constrained Markov Decision Processes*. CRC Press, 1999, vol. 7.

[18] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st. John Wiley & Sons, Inc., 1994.

[19] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, *et al.*, "Safe Learning in Robotics: From Learning-Based Control to Safe Reinforcement Learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, no. 1, pp. 411–444, 2022.

[20] S. Ha, P. Xu, Z. Tan, S. Levine, and J. Tan, *Learning to Walk in the Real World with Minimal Human Effort*, arXiv:2002.08550, 2020.

[21] Q. Yang, T. D. Simão, S. H. Tindemans, and M. T. J. Spaan, "WCSAC: Worst-Case Soft Actor Critic for Safety-Constrained Reinforcement Learning," in *AAAI*, 2021.

[22] ——, "Safety-constrained reinforcement learning with a distributional safety critic," *Machine Learning*, pp. 1–29, 2022.

[23] A. Ray, J. Achiam, and D. Amodei, *Benchmarking Safe Exploration in Deep Reinforcement Learning*, https://cdn.openai.com/safexp-short.pdf, 2019.

[24] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *ICML*, PMLR, 2017, pp. 22–31.

[25] Y. Liu, J. Ding, and X. Liu, "Ipo: Interior-point policy optimization under constraints," in *AAAI*, 2020, pp. 4940–4947.

[26] T.-Y. Yang, J. Rosca, K. Narasimhan, and P. J. Ramadge, "Projection-based constrained policy optimization," in *ICLR*, 2020.

[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal Policy Optimization Algorithms*, arXiv:1707.06347, 2017.

[28] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust Region Policy Optimization," in *ICML*, JMLR.org, 2015, pp. 1889–1897.

[29] D. P. Bertsekas, *Constrained optimization and Lagrange multiplier methods*. Academic press, 1982, vol. 1.

[30] E. Leurent, *An environment for autonomous driving decision-making*, `https://github.com/eleurent/highway-env`, 2018.

[31] M. Treiber, A. Hennecke, and D. Helbing, "Congested Traffic States in Empirical Observations and Microscopic Simulations," *Phys. Rev. E*, vol. 62, pp. 1805–1824, 2 2000.

[32] T. Liu, R. Zhou, D. Kalathil, P. R. Kumar, and C. Tian, "Learning Policies with Zero or Bounded Constraint Violation for Constrained MDPs," in *NeurIPS*, 2021, pp. 17 183–17 193.

[33] T. D. Simão, N. Jansen, and M. T. J. Spaan, "AlwaysSafe: Reinforcement Learning Without Safety Constraint Violations During Training," in *AAMAS*, IFAAMAS, 2021, pp. 1226–1235.

[34] Q. Bai, A. S. Bedi, M. Agarwal, A. Koppel, and V. Aggarwal, "Achieving Zero Constraint Violation for Constrained Reinforcement Learning via Primal-Dual Approach," in *AAAI*, AAAI Press, 2022, pp. 3682–3689.