2019 IEEE Intelligent Transportation Systems Conference (ITSC)
Auckland, NZ, October 27-30, 2019

Lane-Merging Using Policy-based Reinforcement Learning and Post-Optimization

Patrick Hart[1], Leonard Rychly[1] and Alois Knoll[2]

***Abstract*— Many current behavior generation methods struggle to handle real-world traffic situations as they do not scale well with complexity. However, behaviors can be learned off-line using data-driven approaches. Especially, reinforcement learning is promising as it implicitly learns how to behave utilizing collected experiences. In this work, we combine policy-based reinforcement learning with local optimization to foster and synthesize the best of the two methodologies. The policy-based reinforcement learning algorithm provides an initial solution and guiding reference for the post-optimization. Therefore, the optimizer only has to compute a single homotopy class, e.g. drive behind or in front of the other vehicle. By storing the state-history during reinforcement learning, it can be used for constraint checking and the optimizer can account for interactions. The post-optimization additionally acts as a safety-layer and the novel method, thus, can be applied in safety-critical applications. We evaluate the proposed method using lane-change scenarios with a varying number of vehicles.**

I. Introduction

As autonomous driving strives towards level-five autonomy novel behavior generation methods have to be developed to tackle the remaining challenges. One of these remaining challenges is to plan behaviors for complex traffic scenarios having multiple traffic participants.

Data-driven methods, such as policy-based reinforcement learning can learn how to behave in complex situations utilizing collected experiences. However, since the whole configuration space cannot be explored during training and most approaches use deep artificial neural networks, the reinforcement learning algorithm might fail to interpolate and to generalize well [1]. This can lead to unwanted and unexpected behaviors making reinforcement learning infeasible to use in safety-critical applications.

Contrary to this, conventional approaches, such as optimization and search-based techniques provide optimal and safe solutions given the problem is well defined. However, these approaches might fail to handle complex situations in real-time since they do not scale well with the increasing complexity of the environment. A local non-linear optimizer would have to solve multiple homotopy classes to find the global optimum. Figure 1 shows a scenario where a local optimizer would have to evaluate at least two combinatorial options: merge behind or in front of the red vehicle.

The main contributions of this work are the following:

[1]Patrick Hart and Leonard Rychly are with the fortiss GmbH, An-Institut Technische Universität München, Munich, Germany. Email: patrick.hart@tum.de, rychly@fortiss.org

[2]Alois Knoll is with the Chair of Robotics, Artificial Intelligence and Real-time Systems, Technische Universität München, Munich, Germany

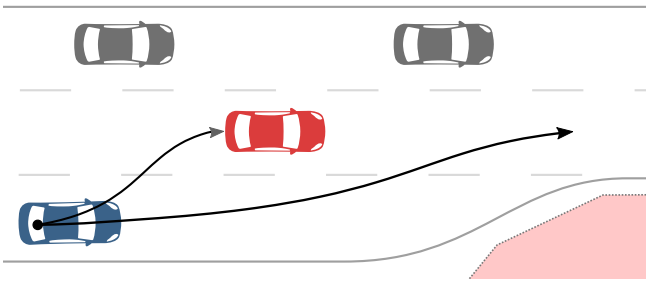

Fig. 1: The blue vehicle wants to merge onto the highway. In this scenario, two homotopy classes exist – merge in front of or behind the red vehicle.

- Obtain a behavior for complex traffic situations using reinforcement learning,
- use this behavior to choose a homotopy class and to include interactions in the optimization and,
- make the reinforcement learning algorithm combined with the post-optimization feasible in safety-critical applications.

In this work, we combine policy-based reinforcement learning with local non-linear optimization to synthesize the best of the two methodologies. The learned policy of the reinforcement learning algorithm is iteratively executed in the environment to generate an initial estimate and a guiding reference trajectory for the optimization. The state history of all vehicles is stored and later used for constraint checking in the optimization. Since all vehicles react to their surroundings according to their intrinsic behavior policy, the interactions between the traffic participants are implicitly included in the optimization. Additionally, due to the initial estimate proposed by the reinforcement learning algorithm, the optimizer only has to solve a single homotopy class – making the optimization also feasible for very complex traffic situations. Moreover, the post-optimization acts as an additional safety-layer after the reinforcement learning algorithm. Therefore, a high level of safety and comfort is guaranteed and the novel approach is also applicable in safety-critical applications.

We demonstrate and evaluate our novel approach using interactive lane-merging scenarios having a varying number of vehicles. All vehicles in the simulation framework have a behavior policy, such as the Intelligent Driver Model [2, 3]. The ego-vehicle is controlled by the novel behavior generation method presented in this work.

The work is further organized as follows: Section II

Authorized licensed use limited to: Technische Universitaet Muenchen. Downloaded on July 27,2022 at 11:01:52 UTC from IEEE Xplore. Restrictions apply.

gives a quick overview of state-of-the-art algorithms used in behavior generation with a special focus on reinforcement learning. Next, a problem definition for this work is provided. In Section IV, our combined reinforcement and optimization approach is introduced. Section V evaluates the performance of the novel method using a lane-merging scenario. Finally, a conclusion, discussion, and an outlook are provided in Section VI.

II. Related Work

This section gives a brief overview of state-of-the-art reinforcement learning methods – with a special focus on behavior generation for autonomous driving. Moreover, a short overview of optimization-based techniques to generate behaviors for autonomous vehicles is given.

Reinforcement learning can roughly be divided into two categories: policy- and value-based methods [4]. In this work, we focus on policy-based reinforcement learning since it achieves state-of-the-art performance in large configuration spaces and dynamic control problems [5, 6]. In the deterministic case, policy-based methods map states to actions and in the stochastic case, they create distributions over actions for each state.

The Trust Region Policy Optimization (TRPO) is a policy-based method that additionally restrains the next policy to be close to the previous one by using the Kullback-Leibler (KL) divergence [7]. Based on this the Proximal Policy Optimization (PPO) has been developed that also restricts the objective function, but by using a clipping function instead of using the KL divergence [8]. This is computationally more efficient than the TRPO and yields equally good or better results in a wide range of applications [9, 10].

However, the above-presented methods explore on-policy and are, thus, more likely to become stuck in local minima. Recently, policy-based methods have been suggested that explore off-policy and, therefore, mitigate this problem, such as the Maximum a Posteriori Policy Optimization (MPO) [6] and the Soft Actor-Critic (SAC) [11]. To leverage the off-policy exploration advantage, we use the SAC algorithm in this work.

Reinforcement learning has been successfully applied to generate behaviors for autonomous vehicles. Shalev-Shwartz et al. [12] use deep reinforcement learning to find long term driving strategies. Other than using an option-graph, they showed that the long-term driving problem can be solved without assuming the Markov property. However, as most of the reinforcement learning methods use deep neural networks, they cannot be simply applied in safety-critical applications as the interpolation and generalization of these cannot be guaranteed [1].

Therefore, reinforcement learning has been combined with formal verification to ensure that only safe actions are executed by the reinforcement learning algorithm [13]. However, due to the curse of dimensionality in formal verification, these methods become untractable fast with a growing action and configuration space [14].

Another approach trying to make reinforcement learning safe and applicable was introduced by Levine et al. [15]. They use differential dynamic programming in order to generate suitable guiding samples to direct policy learning and to avoid poor local optima. However, with the characteristic of neural networks having a large number of local optima, unwanted behavior still can occur and generalization cannot be guaranteed [16].

Contrary to data-driven reinforcement learning techniques, optimization-based behavior generation methods provide safe and optimal solutions. These are, therefore, already used in safety-critical applications, such as autonomous driving. As shown by Bender et al. [17] all combinatorial options (homotopy classes) have to be evaluated in order to find the global optimum when using local optimization. Therefore, these methods generally do not scale well with the increasing complexity of the environment and become infeasible for real-time applications. In this work, we combine reinforcement learning to propose an initial solution and, thus, also a homotopy class for the optimizer. Therefore, the optimizer only has to solve a single homotopy and local-optimization techniques become feasible in very complex traffic scenarios.

III. Problem Statement

The proposed behavior generation method consists of two main components:

1) An off-line stochastic policy-based reinforcement learning method and,
2) a local non-linear post-optimization.

The stochastic policy-based reinforcement learning provides an approximately optimal solution for the Markov Decision Process (MDP). This is achieved by interacting with an environment, collecting experiences and improving its stochastic policy π_ϕ parametrized by an artificial neural network having the parameters ϕ. This policy is then iteratively executed to generate a state-space trajectory $\underline{\mathrm{x}}^{\mathrm{RL}} = [\underline{\mathrm{x}}_0, \ldots, \underline{\mathrm{x}}_{\mathrm{N}}]$ and a control input sequence $\underline{\mathrm{a}}^{\mathrm{RL}} = [\underline{\mathrm{a}}_0, \ldots, \underline{\mathrm{a}}_{\mathrm{N}}]$ for the ego-vehicle having N time-steps. The other vehicles behave according to their defined policies $[\pi_0, \ldots, \pi_{\mathrm{M}}]$ for M other vehicles. All agent's state-space trajectories $[\underline{\mathrm{x}}^{\mathrm{other},1}, \ldots, \underline{\mathrm{x}}^{\mathrm{other},\mathrm{M}}]$ are stored as they are later used for constraint checking in the optimization.

The second part consists of the local non-linear post-optimization that uses the generated state-space trajectory $\underline{\mathrm{x}}^{\mathrm{RL}}$ and the control input sequence $\underline{\mathrm{a}}^{\mathrm{RL}}$ as a guiding reference and initial optimization vector, respectively. Due to this, a homotopy class is implicitly chosen for the optimizer and only a single homotopy has to be evaluated by the local-optimization method to find the globally optimal control input sequence $\underline{\mathrm{a}}^* = [\underline{\mathrm{a}}_0, \ldots, \underline{\mathrm{a}}_{\mathrm{N}}]$ that produces the state-space trajectory $\underline{\mathrm{x}}^* = [\underline{\mathrm{x}}_0, \ldots, \underline{\mathrm{x}}_{\mathrm{N}}]$. Constraints and interactive behavior with other traffic participants are integrated into the optimization using the recorded state-space trajectories $[\underline{\mathrm{x}}^{\mathrm{other},1}, \ldots, \underline{\mathrm{x}}^{\mathrm{other},\mathrm{M}}]$.

Authorized licensed use limited to: Technische Universitaet Muenchen. Downloaded on July 27,2022 at 11:01:52 UTC from IEEE Xplore. Restrictions apply.

IV. Our Method

In this section, we introduce the used policy-based reinforcement learning and how we generate behaviors for autonomous vehicles. Moreover, we then go into detail of the post-optimization and how it is combined with the policy-based reinforcement learning algorithm.

A. Policy-based Reinforcement Learning

This section describes the stochastic policy-based reinforcement learning and how the environment's rewards are defined. In the reinforcement learning problem, we try to optimize the expected cumulative future reward for the stochastic policy denoted by π_ϕ parametrized by the neural network parameters ϕ. The basic objective function expressing the expected cumulative future reward can be mathematically denoted as

$$\mathrm{R} = \sum_{\mathrm{t}} \mathbb{E}_{(\underline{\mathrm{s}}_t, \underline{\mathrm{a}}_t) \sim \pi_\phi} [r(\underline{\mathrm{s}}_t, \underline{\mathrm{a}}_t)] \tag{1}$$

with $r(\cdot)$ being the reward function, $\underline{\mathrm{s}}_t$ the concatenated state-space and $\underline{\mathrm{a}}_t$ the action of the ego-vehicle. However, in this work, we use an extended maximum entropy objective as introduced by Haarnoja et al. [11].

The rewards of the environment are designed to avoid crashes, provide a comfortable driving experience and to follow a given reference with a desired speed. Derived from these characteristics the reward function is defined as

$$\mathbf{R}_{\mathrm{cost}} = \mathbf{R}_{\mathrm{ref}} + \mathbf{R}_{\mathrm{comfort}} + \mathbf{R}_{\mathrm{vel}} + \mathbf{R}_{\mathrm{col}} \tag{2}$$

with $\mathbf{R}_{\mathrm{ref}}$ being the deviation to the reference path, $\mathbf{R}_{\mathrm{comfort}}$ rating the jerk of the trajectory, $\mathbf{R}_{\mathrm{vel}}$ being the deviation to the desired velocity and $\mathbf{R}_{\mathrm{col}}$ the penalty for collisions. All rewards besides the collision-term are modeled as

$$\mathbf{R} = \mathrm{r}_{\max} - \Delta^2_{\mathrm{desired}} \tag{3}$$

with $\Delta^2_{\mathrm{desired}}$ being the squared deviation to a desired value, such as the speed or reference path and $\mathrm{r}_{\max}$ being the maximum achievable reward. Equation 3 will assume its maximum value if the deviation to the desired values is zero. The penalty for a collision $\mathbf{R}_{\mathrm{col}}$ is selected to be a large negative scalar value.

The input state consists of the concatenated states of all vehicles $\underline{\mathrm{s}}_t = [\underline{\mathrm{x}}_0, \ldots, \underline{\mathrm{x}}_{\mathrm{M}}]$ with M being the number of vehicles, t the time-index and $\underline{\mathrm{x}}_i$ the state of the i-th vehicle. On the output layer of the policy network, we utilize a normal distribution $\pi_\phi(\underline{\mathrm{s}}_t) \sim \mathrm{N}(\cdot)$ whose mean is scaled to limit the output space. For scaling the mean to a defined range of minimum and maximum control values ($\underline{\mathrm{a}}_{\min}$ and $\underline{\mathrm{a}}_{\max}$), we use a hyperbolic tangent scaling function that can be denoted as

$$\underline{\mathrm{a}}_{\mathrm{t}} = \frac{(\underline{\mathrm{a}}_{\max} + \underline{\mathrm{a}}_{\min})}{2} + \frac{(\underline{\mathrm{a}}_{\max} - \underline{\mathrm{a}}_{\min})}{2} \tanh \pi_\phi(\underline{\mathrm{s}}_{\mathrm{t}}). \tag{4}$$

After training has been concluded, the policy $\pi_\phi(\underline{\mathrm{s}}_{\mathrm{t}})$ is greedily executed iteratively N-times in order to create the state-space trajectory $\underline{\mathrm{x}}^{\mathrm{RL}} = [\underline{\mathrm{x}}_0, \ldots, \underline{\mathrm{x}}_{\mathrm{N}}]$ and the control input sequence $\underline{\mathrm{a}}^{\mathrm{RL}} = [\underline{\mathrm{a}}_0, \ldots, \underline{\mathrm{a}}_{\mathrm{N}-1}]$. During the greedy execution of the stoachastic policy π_ϕ the maximum values of the distribution are being used in each time-step maximizing Equation 1. All used hyper-parameters, reward values and other characteristics are explained in detail in Section V.

B. Non-linear Post-Optimization

The non-linear optimization utilizes the behavior policy of the reinforcement learning algorithm to obtain an initial optimization vector and a guiding reference. With the assumption that reinforcement learning approximates an optimal solution for the Markov Decision Process (MDP), the optimized solution is enforced to be close to it. Thus, a homotopy class for the optimization is implicitly selected and also complex planning problems become feasible to be solved online and in real-time using local optimization. Additionally, the optimization checks constraints and smoothens the control inputs $\underline{\mathrm{a}}^{\mathrm{RL}}$ of the reinforcement learning.

The optimization problem can be mathematically denoted as

$$\text{minimize } \mathrm{J}(\underline{\mathrm{x}}) \tag{5}$$

$$\text{subject to } \dot{\underline{\mathrm{x}}}(t) = F(\underline{\mathrm{x}}(t), \underline{\mathrm{a}}(t)) \tag{6}$$

with J being the objective function and $F(\underline{x}(t), \underline{u}(t))$ the dynamic model of the ego-vehicle. The dynamic model $\dot{\underline{\mathrm{x}}}(t) = F(\underline{\mathrm{x}}(t), \underline{\mathrm{a}}(t))$ is forward integrated N-times and a state-space trajectory $\underline{\mathrm{x}}^{\mathrm{Opt.}}$ is obtained having the same length as $\underline{\mathrm{x}}^{\mathrm{RL}}$. Thus, a dynamically feasible state-space trajectory is guaranteed. In this work, we use a simplified single track model as presented in [18]. The function $\mathrm{J}(\underline{\mathrm{x}})$ is defined as

$$\mathrm{J}(\underline{\mathrm{x}}) = \mathbf{J}_{\mathrm{jerk}} + \mathbf{J}_{\mathrm{col}} + \sum_{\mathrm{i}=0}^{\mathrm{N}} (\underline{\mathrm{x}}_{\mathrm{i}}^{\mathrm{Opt.}} - \underline{\mathrm{x}}_{\mathrm{i}}^{\mathrm{RL}}) \tag{7}$$

with $\mathbf{J}_{\mathrm{jerk}}$ being the squared jerk cost of the optimized trajectory $\underline{\mathrm{x}}^{\mathrm{Opt.}}$, $\mathbf{J}_{\mathrm{col}}$ being the collision cost and the last term being the distance of the optimized trajectory to the reinforcement learning algorithm's solution.

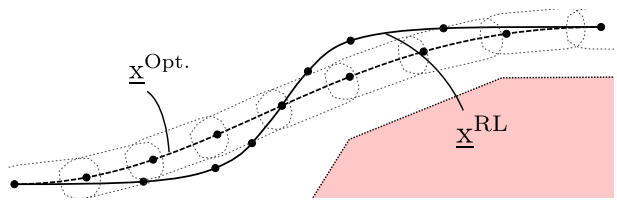

Fig. 2: State-space trajectory provided by the reinforcement learning algorithm and the trajectory generatey by the post-optimization.

Collisions with other vehicles, as well as with boundaries, are included in the collision term $\mathbf{J}_{\mathrm{col}}$. The collision term uses the state history of all agents $[\underline{\mathrm{x}}^{\mathrm{other},1}, \ldots, \underline{\mathrm{x}}^{\mathrm{other,M}}]$ that have been recorded stepping the environment during reinforcement learning. Thus, the interaction with other vehicles is implicitly integrated into the local optimization

Authorized licensed use limited to: Technische Universitaet Muenchen. Downloaded on July 27,2022 at 11:01:52 UTC from IEEE Xplore. Restrictions apply.

method. Collision costs are imposed as soon as the ego-vehicle crosses a pre-defined safety-distance threshold.

The optimization outputs an optimal control input sequence $\underline{a}^* = [\underline{a}_0, \ldots, \underline{a}_N]$ that generates the optimal state-space trajectory $\underline{x}^* = [\underline{x}_0, \ldots, \underline{x}_N]$. After the post-optimization has finished an additional collision-check is performed to guarantee a high level of safety.

V. Results and Evaluation

In this section, we first present the used simulation framework and scenario chosen for training, applying and, evaluating the novel approach. Next, details of the used stochastic policy-based reinforcement learning algorithm – the Soft Actor-Critic (SAC) – are presented. Finally, we show and discuss the results of the novel approach that combines the reinforcement learning algorithm with the post-optimization.

A. Simulation and Scenario

A semantic simulation framework capable of simulating multiple agents has been developed and released in the course of this work [19]. The framework is capable of simulating structured and unstructured environments and, thus, enables simulating a wide range of scenarios. Each agent in the environment is either controlled internally using a behavior model or can be controlled externally by providing system inputs. In this work, all agents but the ego-agent are controlled internally and drive with a constant velocity or behave according to the Intelligent Driver Model (IDM). The ego-vehicle is controlled externally and uses a single-track model that requires the steering angle and acceleration as system inputs ($\underline{a} = [\delta, a]$).

The initial states of all agents are uniformly sampled and, thus, the initial position, velocity, and angle in every episode are varied according to the specified range. In this work, we sample the speed in a range of $v \in [4m/s, 6m/s]$. The desired speed for all agents is set to $5m/s$. Furthermore, each agent has a pre-defined reference path that implicitly motivates each agent's goal.

The simulation framework is executed in an episodical fashion. An episode is counted as successful once the maximum number of steps has been reached without the ego-vehicle causing a collision with any other object. Furthermore, an observer assigns rewards to all events that occur during simulation and also calculates a concatenated state-space $\underline{s}_t = [(x, y, v_x, v_y)_0, \ldots, (x, y, v_x, v_y)_M]$ with M being the number of vehicles. In this work, the concatenated state space includes the Cartesian coordinates x, y as well as the velocities v_x, v_y of each vehicle. The rewards are designed to avoid collisions, generate a comfortable driving behavior and to be close to the original reinforcement learning solution. To model the aforementioned characteristics, the reward values are set to

- $r_{\mathrm{max,vel}} = 10$ for the deviation to the desired velocity,
- $r_{\mathrm{max,ref}} = 10$ for the deviation to the reference path,
- $r_{\mathrm{col}} = -100$ for collisions with other objects and,
- $r_{\mathrm{jerk}} = -0.1$ for the jerk of the trajectory.

Since $r_{\mathrm{max,vel}}$ and $r_{\mathrm{max,ref}}$ are the only positive rewards, the maximum achievable reward per epidose can be calculated using $r_{\mathrm{total}} = \sum_{\mathrm{i=0}}^{\mathrm{N}} r_{\mathrm{max,vel}} + r_{\mathrm{max,ref}}$ with N being the number of steps in a single episode. With the rewards presented above, the maximum achievable reward per episode with a hundred steps is $r_{\mathrm{total}} = 2000$. However, if there is an initial deviation to the reference path or to the desired velocity, the maximum achievable reward is smaller.

B. Policy-based Reinforcement Learning

In this work, we use the Soft Actor-Critic (SAC) policy-based reinforcement learning algorithm. It uses a distributional actor neural network having three layers (512, 512, 128) and a deterministic critic network also having three layers (256, 256, 128). The distributional network uses a normal distribution on the output layer and all other layers use the standard ReLU-activation function. In order to limit the input space for the dynamic model to feasible ranges, we limit the steering-angle $\delta \in [-0.2, 0.2]$ and the acceleration $a \in [-1, 1]$. To evaluate and quantify the results obtained by the SAC algorithm, we train and evaluate the following scenarios:

1) Two-lane merging having two other vehicles,
2) two-lane merging having three other vehicles and,
3) highway merging having five other vehicles.

Figure 3, shows the average reward obtained during training plotted over the episode numbers. It can be seen, that the reward increases for all of the above-presented scenarios and that the algorithm can handle all scenarios well.

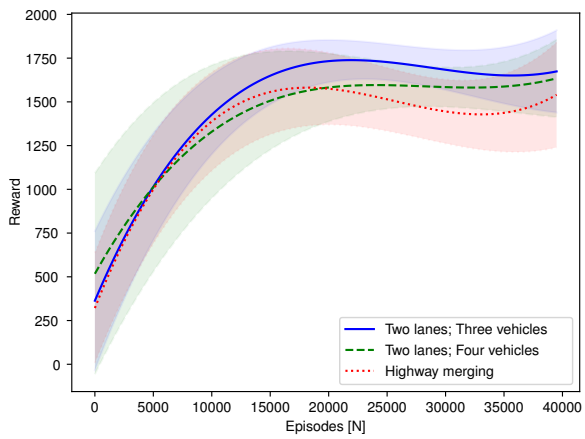

Fig. 3: The average rewards obtained during training by the SAC algorithm.

Quantitative results of applying the SAC algorithm to the above-presented scenarios are shown in Table I. As expected the agent performs best, the less complex the scenario is. Despite this, the SAC algorithm is able to perform well in more complex scenarios, such as the highway merging scenario having many vehicles and multiple lanes.

In Figure 5, a spatial plot of the highway scenario is shown with the ego-vehicle being controlled by the policy-based

Authorized licensed use limited to: Technische Universitaet Muenchen. Downloaded on July 27,2022 at 11:01:52 UTC from IEEE Xplore. Restrictions apply.

Scenario	Success-rate [%]	Avg. Reward
Two-lane merging (Three vehicles)	99.99 %	1816.70
Two-lane merging (Four vehicles)	97.31 %	1827.36
Highway merging	96.05 %	1722.91

TABLE I: For evaluation, the scenarios were run $1,000$ times.

reinforcement learning algorithm (SAC) and the other agents using the Intelligent Driver Model (IDM) as behavior policy.

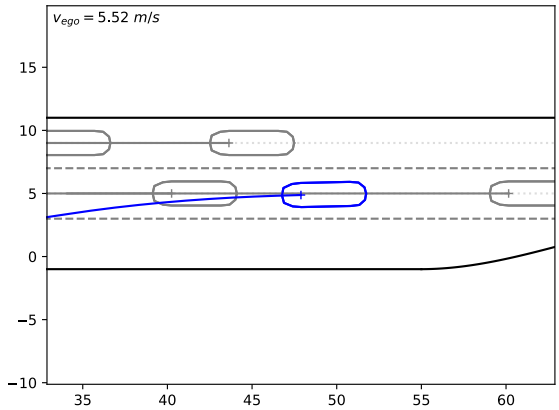

Fig. 4: Resulting behavior produced by the policy-based reinforcement learning approach is depicted in blue. All other vehicles behaving according to the IDM are shown in gray.

The inference time executing the actor-network a single time on an i7-6700HQ processor having 2.60GHz and using Ubuntu 16.04 takes on average about $2.24ms$. The next sub-section will present the results of combining the policy-based reinforcement learning with the post-optimization method.

C. Non-linear Post-Optimization

The results of the novel approach combining policy-based reinforcement learning with a post-optimization are presented and discussed in this sub-section. We use two of the above-presented scenarios for the evaluation – the two-lane with four vehicles and the highway scenario. The optimizer we use in this work is an unconstrained least-squares solver [20].

In Figure 5, a spatial plot of the highway scenario is shown. The reinforcement learning algorithm's solution is depicted in blue and the optimized solution in green. Due to the additional safety-distance in the optimization, the trajectory is pushed further away from the other vehicle. The control input sequence produced by the learned policy of the reinforcement learning algorithm as well as the control input sequence of the optimization is shown in Figure 6. Due to the additional jerk term in the objective function of the optimization, the resulting input sequences of the optimization are smoothened.

Quantitative results of the novel approach are shown in Table II. Due to the optimizer pushing the ego-vehicle away

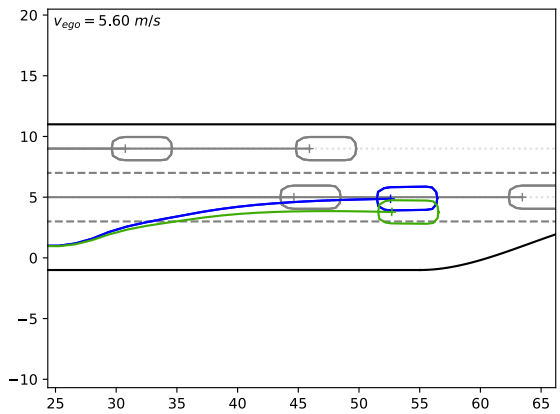

Fig. 5: The reinforcement learning algorithm's solution is depicted in blue and the optimizer's solution in green. The optimizer pushes the trajectory away from the other vehicle and additionally smoothens the trajectory.

from other vehicles and boundaries, the collision rate drops to zero for the give time-horizon. Additionally, the jerk values are reduced and, thus, the comfort of the passengers is increased significantly. In case of a non-convergence of the optimizer, an emergency maneuver, such as emergency-braking could be performed.

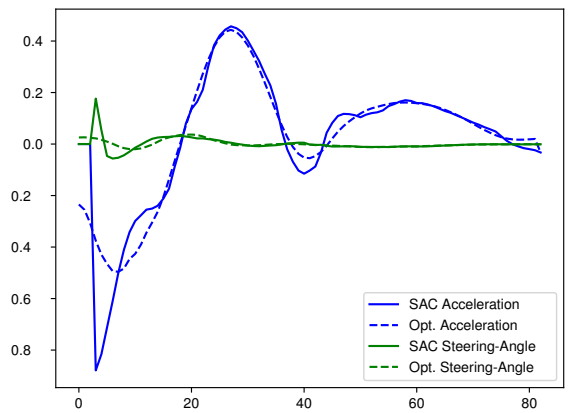

Fig. 6: The control input sequences produced by the reinforcement learning algorithm and the optimization.

As shown in the previous sub-section, the inference-time of the policy-network is on average about $2.24ms$. Therefore, it would take around $112ms$ to produce a trajectory having 50 time-steps. Contrary to this it takes the non-linear local optimizer on average about $206ms$ to solve a single homotopy class with $N = 40$ time-steps [18]. Thus, choosing a homotopy class using reinforcement learning and obtaining an approximately optimal solution for the Markov decision process is more efficient than solving multiple homotopy

Authorized licensed use limited to: Technische Universitaet Muenchen. Downloaded on July 27,2022 at 11:01:52 UTC from IEEE Xplore. Restrictions apply.

Scenario	Jerk. Opt./Jerk SAC [%]	Jerk Opt.
Two-lane merging	0.026	9.7
Highway merging	0.034	7.75

TABLE II: Results of the novel approach applied to the two-lane and highway scenario.

classes. Furthermore, by using Graphics Processing Units (GPUs) the speed of both methods could be further improved – during training as well as during application.

VI. Conclusion

In this work, we showed that a stochastic policy-based reinforcement learning algorithm is able to solve complex scenarios, such as merging lanes with multiple other vehicles present. It learns implicitly and without any prior knowledge how to behave and which homotopy class to choose based on its collected experiences. In order to improve the solution provided by the reinforcement learning and to make it applicable in safety-critical applications, we used a non-linear post-optimization. By using the solution of the reinforcement learning algorithm in the optimization, a homotopy class is implicitly chosen. Thus, only a single objective has to be solved using local optimization instead of all in order to find the global optimum. Moreover, the comfort of the reinforcement learning algorithm's solution is improved due to the post-optimization. Furthermore, the optimizer's convergence can be evaluated for the use in safety-critical applications.

In future research should be conducted on the generalization capabilities of the reinforcement learning to novel scnearios and to a varying number of vehicles. Moreover, occlusions and uncertainties could be modeled in the environment as well as more sophisticated behaviors of the other agents.

References

[1] Justin A. Boyan and Andrew W. Moore. "Generalization in Reinforcement Learning: Safely Approximating the Value Function". In: *Advances in Neural Information Processing Systems 7*. 1994, pp. 369–376.

[2] Rajesh Rajamani. *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[3] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. "Congested traffic states in empirical observations and microscopic simulations". In: *Physical review E* 62.2 (2000), p. 1805.

[4] Kai Arulkumaran et al. "A Brief Survey of Deep Reinforcement Learning". In: *CoRR* abs/1708.05866 (2017).

[5] Timothy P. Lillicrap et al. "Continuous control with deep reinforcement learning". In: *4th International Conference on Learning Representations*. 2016.

[6] Abbas Abdolmaleki et al. "Maximum a Posteriori Policy Optimisation". In: *6th International Conference on Learning Representations*. 2018.

[7] John Schulman et al. "Trust Region Policy Optimization". In: *Proceedings of the 32nd International Conference on Machine Learning*. 2015, pp. 1889–1897.

[8] John Schulman et al. "Proximal Policy Optimization Algorithms". In: *CoRR* abs/1707.06347 (2017).

[9] Martin Klissarov et al. "Learnings Options End-to-End for Continuous Action Tasks". In: *CoRR* abs/1712.00004 (2017).

[10] Nicolas Heess et al. "Emergence of Locomotion Behaviours in Rich Environments". In: *CoRR* abs/1707.02286 (2017).

[11] Tuomas Haarnoja et al. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. 2018, pp. 1856–1865.

[12] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. "Safe, Multi-Agent, Reinforcement Learning for Autonomous Driving". In: *CoRR* abs/1610.03295 (2016).

[13] Branka Mirchevska et al. "High-level Decision Making for Safe and Reasonable Autonomous Lane Changing using Reinforcement Learning". In: *21st International Conference on Intelligent Transportation Systems*. 2018, pp. 2156–2162.

[14] Nikolaus Correll. *Distributed Autonomous Robotic Systems: The 14th International Symposium*. Springer, 2019, pp. 266–267.

[15] Sergey Levine and Vladlen Koltun. "Guided Policy Search". In: *Proceedings of the 30th International Conference on Machine Learning*. 2013, pp. 1–9.

[16] Grzegorz Swirszcz, Wojciech Marian Czarnecki, and Razvan Pascanu. "Local minima in training of neural networks". In: *arXiv preprint arXiv:1611.06310* (2016).

[17] Philipp Bender et al. "The combinatorial aspect of motion planning: Maneuver variants in structured environments". In: *2015 IEEE Intelligent Vehicles Symposium*. 2015, pp. 1386–1392.

[18] Patrick Hart and Alois Knoll. "Kinodynamic Motion Planning Using Multi-Objective optimization". In: *2018 IEEE Intelligent Vehicles Symposium*. 2018, pp. 1185–1190.

[19] Patrick Hart. *Divine-RL: an autonomous driving environment for applied reinforcement learning*. `https://github.com/patrickhart/divine-rl`. 2019.

[20] Sameer Agarwal, Keir Mierle, et al. *Ceres Solver*. `http://ceres-solver.org`.

Authorized licensed use limited to: Technische Universitaet Muenchen. Downloaded on July 27,2022 at 11:01:52 UTC from IEEE Xplore. Restrictions apply.