Lexicographic Actor-Critic Deep Reinforcement Learning for Urban Autonomous Driving

Hengrui Zhang, Youfang Lin, Sheng Han, and Kai Lv

***Abstract*—Urban autonomous driving is a difficult task because of its complex road scenarios and the interaction between multiple vehicles. Autonomous vehicles need to balance multiple objectives in these complex scenarios, e.g., safety and speed. Traditional reinforcement learning methods deal with the multi-objective problem by optimizing agents with a single objective reward. However, these methods are sensitive to the reward scale and require huge experiments to design reward weights. In this paper, we propose the Lexicographical Proximal Policy Optimization algorithm (LPPO), which can express people's preference relationship through the lexicographical ordering between objectives. The proposed method has two main advantages. *On the one hand,* LPPO has a smaller parameter adjustment space, which makes it easy to find the optimal solution that satisfies the actual problem preference. *On the other hand,* the proposed method is less affected by the reward scale and easy to deploy in various driving scenarios. We evaluate our algorithm in two driving simulation environments, and the results show that the proposed method has better performance in urban driving tasks than previous reinforcement learning algorithms. In addition, we illustrate that the proposed method has better stability even if the reward scale changes.**

***Index Terms*—Reinforcement learning, Urban autonomous driving, Actor-critic, Multi-objective.**

I. Introduction

URBAN autonomous driving is a hot topic that has attracted more and more attention. It enables humans to have an efficient and enjoyable driving experience beyond traditional driving [1], [2], [3]. With the rapid development of deep reinforcement learning, many researchers have tried to integrate reinforcement learning into decision-making systems [4], [5]. However, how to build a safe urban driving decision system remains a huge challenge.

Urban autonomous driving is a challenging task because it has complex road scenarios. e.g., highway and intersection. There exist multiple objectives in these scenarios, and autonomous vehicles have a trade-off within different objectives. For example, when passing through an intersection, an autonomous vehicle has two objectives, i.e., safety and speed. On the one hand, a safety driving indicates that no collision happens and a long distance is desired when crossing the intersection. On the other hand, a vehicle is expected to have a higher speed during the driving process. However, it is difficult to balance the two objectives and optimize them quantitatively.

When solving the multi-objectives decision problem in autonomous driving, we can utilize single-objective reinforcement learning approaches [4], from value function method Deep-Q-Learning (DQN) [5] to various actor-critic methods, e.g., Twin Delayed Deep Deterministic Policy Gradient (TD3) [6], Proximal Policy Optimization (PPO) [7], and Soft Actor-Critic (SAC) [8].

The above methods deal with the multi-objective problem by optimising agents with a single objective. However, single-objective approaches have two problems when solving the multi-objective task reasonably. *On the one hand*, as the single-objective methods blend different objective rewards into one overall reward, it is difficult to independently adjust the importance between multiple objectives, thus making the algorithms difficult to determine the weights between the rewards. *On the other hand*, the overall performance is sensitive to the reward scales of multiple objectives. For example, we assume that the speed reward during driving is 0-120 km/h, and the scale of distance reward is 0-10 km. In this case, we require numerous attempts to determine the parameters of multiple objectives and obtain a trained network. However, in another scenario where the distance scale is 0-100 km, the above parameters would become invalid and must be adjusted by additional experiments.

In order to solve the first problem, we propose to *sequentially* define the importance of different objectives by introducing Thresholded Lexicographical Ordering (TLO) [6], [7], [8]. The lexicographical ordering defines the priority order between multiple objectives. The threshold indicates a trade-off between the primary and secondary objectives. For example, in autonomous driving, we may prefer to allow small deviations in travel distance (the priority objective) with the hopes of obtaining large improvements in speed (the secondary objective). Thus, it is practical to involve TLO into the reinforcement learning framework to solve the multi-objective problem. Specifically, we propose to combine TLO and the actor-critic algorithm to solve the problem of multiple objectives in urban driving.

Actor-critic architecture is widely used in reinforcement learning. It has two networks: actor network and critic network. The actor decides which action should be taken, and the critic informs the actor how good was the action and how it should adjust.

Manuscript received 1 March 2022; revised 17 September 2022 and 3 November 2022; accepted 7 November 2022. Date of publication 5 December 2022; date of current version 18 April 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62206013 and in part by China Postdoctoral Science Foundation under Grant 2022M720391. The review of this article was coordinated by Dr. Ajeya Gupta. *(Corresponding author: Kai Lv.)*

The authors are with the School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China (e-mail: qirenzhr@gmail.com; yflin@bjtu.edu.cn; shhan@bjtu.edu.cn; lvkai@bjtu.edu.cn).

Digital Object Identifier 10.1109/TVT.2022.3226579

Based on TLO and the actor-critic architecture, we then focus on solving the second problem. Note that there exist two different ways depending on whether TLO is deployed in the actor or critic component. If we deploy TLO directly on the critic part, the second problem still exists. In this work, we analyze the difference between the two ways (more details in Section IV-E) and finally deploy TLO in the actor part. We find that deploying TLO in the actor part is scale insensitive for the reward and has better performance in most instances.

In this paper, we propose the Lexicographical Actor-Critic (LAC) architecture and design the corresponding Lexicographical Proximal Policy Optimization (LPPO) algorithm. LPPO is a multi-branch architecture that contains an actor and a critic for multi-objective optimization, and TLO is deployed in the actor part. To utilize off-policy data reasonably, we also propose objective clip and objective V-trace. We set up the corresponding multi-objective tasks in the MetaDrive [9] and SMARTS [10] autonomous driving environments. In these tasks, autonomous vehicles are expected to avoid collisions with other social vehicles relatively quickly as they travel from an initial location to a destination. Experiments show that our method has more stable and better performance than previous approaches.

In summary, this paper has the following contributions.

- We introduce TLO into the actor-critic architecture to sequentially define the importance of multiple objectives. Note that the proposed method can better reflect human preferences.
- We deploy TLO in the actor part, making it scale insensitive for the reward and has better performance. In addition, we analyze the difference between our method and the algorithm that deploys the TLO in the critic part.
- We propose LPPO to solve the multi-objective problem in urban autonomous driving scenes. Experimental results show that LPPO can perform better when solving collision avoidance and crossing intersections in urban driving.

The rest of this article is organized as follows. In Section II, we briefly introduce the related works about reinforcement learning in urban driving and multi-objective reinforcement learning. Section III is the problem formulation about using reinforcement learning in urban autonomous driving. In Section IV, we describe the LAC architecture and the corresponding LPPO algorithm. In Section V, we show our experiments in the simulation environment and analyse the corresponding results. Finally, we conclude this article in Section VI.

II. Related Work

In this section, we review the reinforcement learning methods in urban driving. In addition, we also review the Multi-objective Reinforcement Learning (MORL).

A. Reinforcement Learning in Urban Driving

There exist many tasks, e.g., vehicle navigation and autonomous driving, utilize reinforcement learning to solve the problems in urban driving scenes. Some works [11], [12], [13] train self-driving vehicles to navigate in urban scenes. Toromanoff et al. [11] propose a value-based Rainbow-IQN-Apex and a large conditional network architecture for urban navigation. Note that the Rainbow-IQN-Apex network is trained with an adapted reward.

Controllable Imitative Reinforcement Learning (CIRL) [12] incorporates controllable imitation learning with Deep Deterministic Policy Gradient (DDPG) policy learning to address the challenging problem of vision-based autonomous driving in the high-fidelity car simulator. Huang et al. [13] embed a Deduction Reasoner (DR) into DDPG to foresee future trajectories from the current state and propose a Deductive RL (DeRL) method to address the challenging problems of vision-based autonomous urban driving.

Many efforts [14], [15], [16] also focus on some related tasks in autonomous driving. Fu et al. [14] analyze the vehicle lane-changing and braking processes in detail and designs the autonomous braking systems using DRL based on the AC architecture. Song et al. [15] adopt an end-to-end neural network controller by utilizing task-specific curriculum reinforcement learning and a reward formulation for high-speed autonomous overtaking. Cai et al. [16] utilize the soft actor-critic method and train a closed-loop controller to control the steering angle and throttle of simulated vehicles. Qiao et al. [17] use hierarchical reinforcement learning to train autonomous vehicles to stop at Stop-Line (SSL) and Follow Front Vehicle (FFV).

Many works pay attention to how to safely pass through unprotected intersections, where most accidents occur [18]. The difficulty lies in crossing unprotected intersections is the absence of lights and hard-to-predict interactions among possibly many road participants. Isele et al. [19] propose Time-to-Go DQN to demonstrate the effectiveness of deep learning techniques on the intersection handling problem. Mahamed et al. [20] develop a benchmark called ULTRA for robust agents to train in the unprotected left turn. During a left-turn scenario, Shu et al. [21] introduce hierarchical left-turn planning for autonomous vehicles to perform safe and efficient planning behaviors as a driving expert. The above previous works only consider one objective, named safety or success rate, during autonomous driving. However, when crossing intersections, many competing objectives should be involved in the driving process, and a vehicle should safely pass through an intersection at a relatively high speed.

B. Multi-Objective Reinforcement Learning

The goal of MORL is to select one policy to satisfy a specific preference from Pareto front. Pareto front is a set of nondominated solutions, being chosen as optimal, if no objective can be improved without sacrificing at least one other objective. Existing MORL algorithms can be roughly divided into two categories regarding the number of learned policies [22]. The first category of MORL methods is the single-policy method, which are designed to learn one policy that best satisfies a set of preferences between objectives. The second category of MORL approaches is the multi-policy method, and the policies are expected to approximate the Pareto front.

Compared with multi-policy methods, single-policy algorithms use less memory and are easier to implement. In addition, instead of finding all feasible solutions using a multi-policy

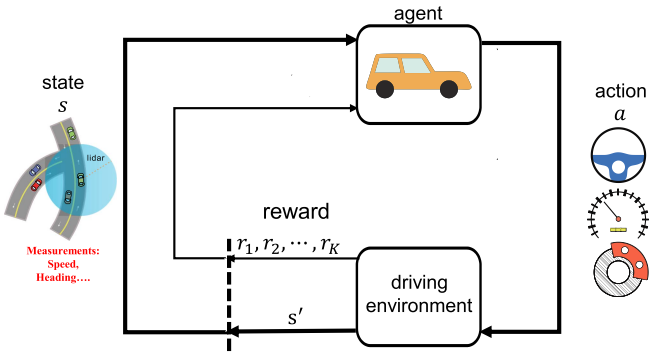

Fig. 1. The elements of a reinforcement learning model in urban driving. The vehicle agent interacts with the driving environment and performs actions (throttle, brake, and steering angle) based on the current state.

approach, single-policy only needs to find the optimal solution that matches people's preferences. Thus, single-policy methods are more suitable for urban driving tasks.

Many single-policy approaches [23], [24] utilize a linear scalarization function to better express people's preferences in some fields. However, in many cases this linear function cannot adequately represent the preferences for multiple objectives [25]. Meanwhile, many non-linear methods, such as thresholded lexicographic methods [7], [8], [26] and Chebyshev scalarization function [27], are proposed to better represent objective preferences. Multi-objective Maximum a Posteriori Policy Optimization (MO-MPO) [28] imposes a constraint on each objective when updating the policy. However, to realize the scalarization and the constraint, previous works are parameter sensitive, which means that a small difference of two parameter settings may dramatically influence the importance of multiple objectives. In other words, to achieve a desired objective preference, these approaches require huge experiments to determine the relative weights of the objectives, making these approaches infeasible to deploy in real scenarios. Hence, Li et al. [26] propose lexicographic DQN to define the objective preference by introducing dynamic thresholded lexicographic order. However, lexicographic DQN is sensitive to the scale of the reward. In this work, our method LPPO is nonlinear and can better express people's preference for multi-objectives relative to linear methods. In addition, LPPO is parameter insensitivity and reward scale insensitivity, making it suitable to deploy in real urban driving scenarios.

III. Problem Formulation

This section formalizes urban autonomous driving into a reinforcement learning problem. As shown in Fig. 1, at each driving step, the agent needs to decide the control strategy according to the current state. The vehicle executes the control action to enter the next state, and the agent gets a new action according to the new state. The whole process loops back and forth until the vehicle reaches the endpoint.

As there exist multiple rewards in training, the above learning process can be regarded as a Multi-objective Markov Decision Process (MOMDP). Assume that there exist K objectives in a task. A typical MOMDP consists of state $s \in \mathcal{S}$, action $a \in \mathcal{A}$, reward $\{r_k(s,a) \in \mathbb{R}\}_{k=1}^{K}$ per objective k, transition probabilities $p(s'|s,a)$, and a discount factor $\gamma \in [0,1)$. $\mathcal{S}$ and $\mathcal{A}$ correspond to state set and action set, respectively. For the k-th objective, the reward can be represented as $r_k : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, mapping a state s and action a to a reward $r_k(s,a)$. $p(s'|s,a)$ is the probability of changing from state s_t to s_{t+1} when taking action a_t. In the simulation environment, the state-to-state transition is determined by the simulator. For each objective k, we define the corresponding Q function and V function as follows:

$$Q_k^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_k(s,a)\right], \tag{1}$$

$$V_k^{\pi}(s) = \mathbb{E}_{\pi}[Q_k^{\pi}(s,a)]. \tag{2}$$

We solve this sequential decision problem by employing reinforcement learning methods. Reinforcement learning aims to control an agent to maximize the reward function. In order to learn the optimal strategy in urban driving, we use Actor-Critic as the infrastructure. In the Actor-Critic architecture, the actor network outputs a policy $\pi(a|s)$ that represents the behavior of the agent in the current state, and the critic network outputs a value estimate $V(s)$ to evaluate the current policy of the agent. The overall update rules rely on policy gradient theory [29] as follows:

$$\nabla_\theta J(\theta) = E_{s_t,a_t}[Adv^{\pi_\theta}(s_t,a_t)\nabla_\theta log\pi_\theta(a_t|s_t)], \tag{3}$$

$Adv^{\pi_\theta}(s_t,a_t)$ is the advantage function, measures whether or not the action is better or worse than the policy's default behavior. The definition of the advantage function is as follows:

$$Adv^{\pi_\theta}(s_t,a_t) = Q^{\pi_\theta}(s_t,a_t) - V^{\pi_\theta}(s_t), \tag{4}$$

Positive $Adv^{\pi_\theta}(s_t,a_t)$ means that the action was better than average and minimizing the loss function will increase the probability of sampling the same action again.

For the multi-reward situation in autonomous driving, a natural idea is to combine multiple sub-rewards according to their weights w_k into a total reward, and finally maximize the Q-value corresponding to the total reward,

$$r(s,a) = \sum_{k=1}^{K} w_k * r_k(s,a), \tag{5}$$

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s,a)\right], \tag{6}$$

A more direct way is to perform a weighted combination of Q to get the final Q function as:

$$Q^{\pi}(s,a) = \sum_{k=1}^{K} w_k * Q_k^{\pi}(s,a), \tag{7}$$

However, the weight combination method is sensitive to the reward scale and requires massive experiments to determine the weight coefficient.

IV. METHOD

A. Thresholded Lexicographic Order

In the autonomous driving scenarios, humans always consider the quality of the second objective on the premise of ensuring that the first objective meets the requirements. For example, when people consider the driving time and the expense as two objectives, they will balance these two objectives based on their preferences. If they prefer the previous objective, they will first choose the collection of alternatives to meet the minimum driving time requirement and then select the best option from the alternatives according to the expense objective. Inspired by the above process, we utilize Thresholded Lexicographical Ordering (TLO) to express the preference between multiple objectives.

Formally, lexicographic ordering $1, 2, \ldots, k, \ldots, K$ is on the K objectives of MOMDP (S, A, P, R, γ), and τ'_k is a threshold that specifies the minimum admissible value for each objective. Thresholded lexicographic Q-learning finds K sets of policies $\Pi_k, k = 1, 2, \ldots, K$ that maximize $\{\hat{Q}^*_1(s,a), \hat{Q}^*_2(s,a), \ldots, \hat{Q}^*_K(s,a)\}$ in lexicographic order:

$$\Pi_k = \{\pi_k \epsilon \Pi_{k-1} | \pi_k(s) = argmax_{a \epsilon \pi_{k-1}(s)} \hat{Q}^*_k(s,a), \forall s \epsilon S\}, \tag{8}$$

$\hat{Q}^*_k(s,a)$ is the Q function rectified to τ'_k :

$$\hat{Q}^*_k(s,a) = \min(\tau'_k, Q^*_k(s,a)), \tag{9}$$

Note that the static threshold τ'_k requires strong prior knowledge and is difficult to extend to more practical problems.

An alternative solution [26] is to deploy a dynamic threshold. Specifically, this method [26] ignores $\hat{Q}^*_k(s,a)$ and estimates $Q^*(s,a)$ directly through the following Bellman equation:

$$Q^*_k(s,a) = r_k(s,a) + \gamma_k \sum_{s'} P(s'|s,a) \max_{a' \in \{\pi_{k-1}(s) | \pi_{k-1} \in \Pi_{k-1}\}} Q^*_k(s', a'), \tag{10}$$

where Π_k is redefined as:

$$\Pi_k = \{\pi_k \epsilon \Pi_{k-1} | Q^*(s, \pi_k(s)) \geq \max_{a \epsilon \pi_{k-1}} Q^*(s,a) + \tau_k \quad or \quad \pi_k(s) = argmax_{a \epsilon \pi_{k-1}(s)} \hat{Q}^*_k(s,a), \forall s \epsilon S\}, \tag{11}$$

Note that the static threshold has been replaced by a dynamic threshold that depends on the learned Q function. Here, τ_k means how much worse than the best action is considered acceptable in each state.

In this paper, to reduce the impact of the reward scale on the algorithm performance, we define the threshold as the policy threshold without directly relying on the Q-function.

B. Thresholded Lexicographical Ordering Actor-Critic

1) Architecture: In this paper, we combine Thresholded Lexicographical Ordering (TLO) and the actor-critic algorithm to propose the Lexicographical Actor-Critic (LAC) architecture that consists of an *actor network* and *a critic network*. The architecture of LAC is illustrated in Fig. 2. *On the one hand,*

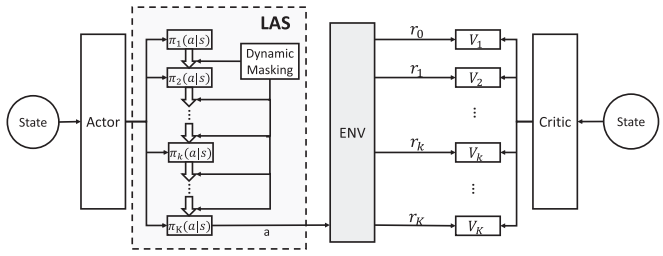

Fig. 2. Architecture of Lexicographic Actor-Critic (LAC). LAC is a multi-branch architecture that contains an actor and a critic for multi-objective optimization. The architecture utilizes Lexicographical Action Selection (LAS) to choose an action based on multiple objectives.

the actor network is a multi-branch architecture that contains multiple policy distributions, and the policy distribution of the objective k is represented by $\pi_k(a|s,\theta)$. Note that $\pi_k(a|s,\theta)$ is based on the previous objective policy distribution $\pi_{k-1}(a|s,\theta)$. *On the other hand*, the critic network is also a multi-branch architecture that has several value functions. In this work, the value function V_k corresponds to policy distribution π_k.

2) Lexicographical Action Selection: Different from the typical actor-critic algorithm that directly adopts the actor output as action a, LAC utilizes Lexicographical Action Selection (LAS) to choose an action based on multiple objectives. LAS is a process that shrinks the initial action space $A_0(s)$ to the final action space $A_K(s)$, and the process can be written as $A_0(s) \to A_1(s) \to \cdots \to A_k(s) \to \cdots \to A_K(s)$. Note that transferring $A_{k-1}(s)$ to $A_k(s)$ is called *dynamic action masking*, which eliminates actions that are not satisfied with objective k. Finally, the proposed method selects action a from $A_K(s)$ as the output action of this interaction. Algorithm 1 illustrates the LAS process.

Consider $[l_1, l_2, \ldots, l_n]_k$ as the output logits of previous policy distribution $\pi_{k-1}(a|s,\theta)$. Note that n equals the number of actions. The dynamic action masking operation D_k for objective k can be written as:

$$D_k(l_i) = \begin{cases} l_i, & a_i \in A_k(s) \\ M, & a_i \notin A_k(s) \end{cases} \tag{12}$$

where $i = 1, 2, \cdots n$, M is a large negative number. $A_k(s)$ is an action space that determined by state s and can be written as:

$$A_k(s) = \{a \in A_{k-1}(s) | \pi_k(a|s) \geq \max_{a' \in A_{k-1}(s)} \pi_k(a'|s) - \tau_k\}, \tag{13}$$

where τ_k is the hyperparameter that specifies how much worse than the best action is considered acceptable in each state under objective k. $\max_{a' \in A_{k-1}(s)} \pi_k(a'|s) - \tau_k$ is the dynamic threshold that changes during optimization process.

After obtaining the action space A_k, we reorganize the distribution of the actions. For action $a \notin A_k$, the proposed method sets the corresponding action logit to M, while the action logit of action $a \in A_k$ remains unchanged. Finally, we utilize softmax to obtain the probabilities of all actions.

During LAS, there are three different probability distributions: $\pi_k(\cdot|s)$, $\pi_{k'}(\cdot|s)$, and $\pi_{k^*}(\cdot|s)$. $\pi_k(\cdot|s)$ is the distribution of k-th actor branch. After the dynamic action masking operation D_k, $\pi_k(\cdot|s)$ is converted to $\pi_{k'}(\cdot|s)$. $\pi_{k^*}(\cdot|s)$ indicates the final distribution of sampling actions which interact with the environment.

The distributions are as follows:

$$\pi_k(\cdot|s) = \text{softmax}([l_1, l_2, \ldots, l_n]_k), \tag{14}$$

$$\pi_{k'}(\cdot|s) = \text{softmax}(D_k([l_1, l_2, \ldots, l_n]_k)), \tag{15}$$

$$\pi_{k^*}(\cdot|s) = \pi_{(K)'}(\cdot|s). \tag{16}$$

After the action selection stage, we get state-action pairs, that are produced when interacting with the environment. The state-action pairs are utilized to update the actor-critic network. Specifically, the K branches of the critic network are updated under the guidance of K objective rewards.

C. Thresholded Lexicographical Ordering Proximal Policy Optimization

As the proposed method can be applied to actor-critic algorithms, we use Proximal Policy Optimization (PPO) [30] for illustration.

After the sample interaction, we calculate the advantage value of the corresponding state-action according to Generalized Advantage Estimator (GAE) [31]:

$$\delta_{t,k} = r_{t,k} + \gamma V_k(s_{t+1}) - V_k(s_t), \tag{17}$$

$$Adv_{t,k} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l,k}, \tag{18}$$

where t is the current step, $\gamma \in [0, 1)$ is a discount, and λ is a generalized advantage estimator that makes a compromise between bias and variance.

The advantage function $Adv_{t,k}$ measures whether or not the action is better or worse than the policy's default behavior corresponds to the objective k. GAE is a simple but effective advantage estimator, which significantly reduces variance empirically while maintaining a tolerable level of bias.Then, we normalize the advantage value of each objective k to reduce the influence caused by the advantage scale.

With the advantage value $Adv_{t,k}$ and the current critic network estimation $V_k(s_t)$, we can calculate the target value of the k-th critic branch as follows:

$$V_k^{target}(s_t) = Adv_{t,k} + V_k(s_t). \tag{19}$$

We can update the critic network, and the critic loss is as follows:

$$L_v(w) = \sum_{s_t} \sum_{k=1}^{K} (V_k(s_t, w) - (V_k^{target}(s_t)))^2, \tag{20}$$

where w is the parameter of the critic network.

Algorithm 1: Lexicographic Action Select.

Input: State s, Objectives number K,
initial action set $A_0(s) = \mathcal{A}$
Parameter: θ in $\pi_k(a|s, \theta)$
Output: Final action set

```
for k in {1, 2, ..., K} do
    Obtain A_k(s) according to (13)
    if A_k(s) is NULL then
        A_k(s) = A_{k-1}(s)
    end if
    if objective k is chosen to be explored then
        return A_{k-1}(s)
    end if
end for
return A_K(s)
```

Similar to the clip loss of PPO, we use the following multi-objective clip loss to update the actor network:

$$L(\theta) = \sum_{(s_t,a_t)} \sum_{k=1}^{K} \left[\min \left(\frac{\pi_k(a_t|s_t,\theta)}{\pi_{k^*}(a_t|s_t,\theta_{old})} Adv_{t,k}, \right. \right.$$
$$\left. \left. \text{clip}\left(\frac{\pi_k(a_t|s_t,\theta)}{\pi_{k^*}(a_t|s_t,\theta_{old})}, 1-\epsilon, 1+\epsilon \right) Adv_{t,k} \right) \right], \tag{21}$$

where ϵ is a hyperparameter, say, $\epsilon = 0.2$. $\text{clip}(\frac{\pi_k(a_t|s_t,\theta)}{\pi_{k^*}(a_t|s_t,\theta_{old})}, 1-\epsilon, 1+\epsilon)$ removes the incentive for moving probability ratio $\frac{\pi_k(a_t|s_t,\theta)}{\pi_{k^*}(a_t|s_t,\theta_{old})}$ outside of the interval $[1-\epsilon, 1+\epsilon]$. If $x > 1+\epsilon$, $\text{clip}(x, 1-\epsilon, 1+\epsilon) = 1+\epsilon$; if $x < 1-\epsilon$, $\text{clip}(x, 1-\epsilon, 1+\epsilon) = 1-\epsilon$; if $1-\epsilon < x < 1+\epsilon$, $\text{clip}(x, 1-\epsilon, 1+\epsilon) = x$.

D. Off-Policy Update

In the LAC architecture, there is only one sampling distribution for each interaction with the environment. However, each actor branch needs to update in the learning process, which means off-policy data exists during the training process. Off-policy data may cause unstable in the training process. In this part, we propose two training techniques: objective clip and objective V-trace.

Objective clip means that only part of interactive data is utilized in the training process. In the original on-policy setting, the algorithm sample the output distribution and utilize the sampling results to update the same output distribution. As there exist multiple objective distributions in our method, we need to determine the corresponding data to update the corresponding distribution.

In the training process, there exist two kinds of distribution: the distribution of actual action and the distribution of the actions from actor branches. We utilize the probability ratio of the two distributions on the state-action pair to judge whether the actor branch should be updated. If the ratio is smaller than the hyperparameter d, we use the corresponding state-action data to

Algorithm 2: LPPO.

Input: Actor network $[\pi_1(a|s), \pi_2(a|s), \ldots, \pi_K(a|s)]$
Critic network $[V_1(s), V_2(s), \ldots, V_K(s)]$
Parameter: θ, w, β
for each iteration **do**
  Collect dataset $\{s_i, a_i, r_i\}$ with Actor network and Algorithm1
  According (18), (19) to calculate advantage $Adv_{t,k}$ and target value $V_{t,k}$
  for each gradient step **do**
    According (20) to calculate critic loss $L_v(w)$
    According (22) to choose the corresponding (s, a, k) to calculate actor loss $L(\theta)$ (21)
    Compute the final objective function $J(\theta, w) = L_v(w) + L(\theta)$
    update $(\theta', w') \leftarrow (\theta, w) + \beta \nabla_{\theta,w} J(\theta, w)$
  end for
end for

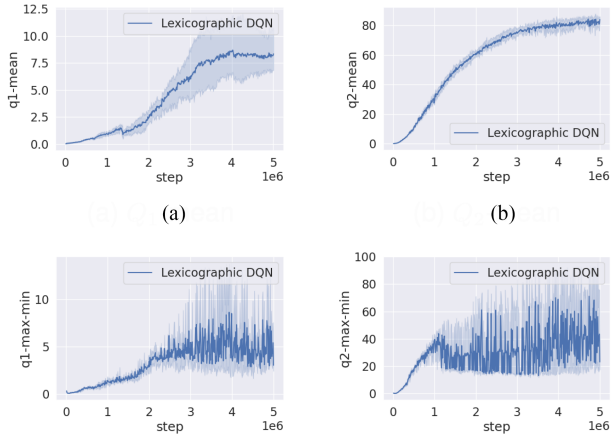

Fig. 3. The Q-value of Lexicographic DQN during training with two objectives. Q_1 and Q_2 represent the Q-value of different objectives. Note that Q_1 and Q_2 have different reward scale. (a) Q_1-mean. (b) Q_2-mean. (c) Q_3-maxmin. (d) Q_4-maxmin.

update this actor branch. The objective clip is defined as:

$$abs\left(\frac{\pi_k^*(a|s)}{\pi_k(a|s)} - 1\right) \leq d. \tag{22}$$

When updating the value network, we propose Objective V-trace, which is similar to impala V-trace [32] and GAE V-trace [33], to reduce the variance of the value estimation when the distribution is inconsistent. Specifically, the advantage function $Adv_{t,k}$ and the value function $V_{t,k}$ are described as follows:

$$Adv_{t,k} = \sum_{l=t}^{T-1}\left(\prod_{i=t+1}^{l} \min\left(1.0, \frac{\pi_k(a_i|s_i)}{\pi_{k^*}(a_i|s_i)}\right)\right)(\gamma\lambda)^{l-t}\delta_{l,k}, \tag{23}$$

$$V_{t,k} = \min\left(1.0, \frac{\pi_k(a_t|s_t)}{\pi_{k^*}(a_t|s_t)}\right) Adv_{t,k} + V_k(s_t). \tag{24}$$

E. Discussions

1) What the difference is between dynamic action masking and static operation: Recent works suppose to utilize a technique known as invalid action masking [34], [35] to avoid repeatedly sampling invalid actions in full discrete action spaces. This action masking technique is static because the current state of invalid actions is known to the environment. In other words, invalid action masking that defines whether an action is invalid relies on prior knowledge. However, different from invalid action masking, dynamic action masking eliminates actions by estimating the importance of different objectives. we can imagine that after the previous k-1 actor branches are trained to convergence, the trained branches can be regarded as prior knowledge, thus the training of the k-th branch approximates to invalid action masking. Thus, the training for multi-branch actors is a process of finding inappropriate actions by learning prior knowledge.

2) Why thresholded lexicographic order is deployed in actor: In the multi-branch actor-critic architecture, the TLO is deployed both in the actor and in the critic. We name the method that deploys TLO in the actor part as the policy threshold, while the method that deploys TLO in the critic part is the value threshold. There are three reasons for utilizing the policy threshold instead of the value threshold. First, the policy threshold can reduce the tuning space.

The range of the policy threshold is $[0, 1]$ (when the action space is discrete), while the range of the value threshold is $[0, \infty)$

Second, the value function changes dramatically as training progresses, and the value threshold cannot adapt to this change and cause failure. Third, the value function is closely related to the reward value. If objective rewards change in scale, the defined value threshold needs to be adjusted. In other words, the value threshold is not scale invariant.

LDQN [26] is regarded as a value threshold method. Fig. 3(a) and Fig. 3(b) show the average curve of LDQN. Q_1 and Q_2 are two objective Q-functions. On the one hand, the value function Q_1 and Q_2 both increases dramatically when training. On the other hand, when the objective scales vary, we can observe that the Q-function scales of the two objectives are different. Hence, it is challenging to design the value threshold for each objective. Fig. 3(c) and Fig. 3(d) show the Q-maxmin curve of LDQN. Q-maxmin represents the value of $\max(Q) - \min(Q)$. It shows that the gap between the maximum Q-value and the minimum Q-value is gradually increasing during the learning process, which also affects the adjustment of the value threshold.

V. Experiments and Results

In this section, experiments on three kinds of simulation platforms are conducted to evaluate the effectiveness of these algorithms. One experimental platform is a simple environmental Safety-gym [36], and the rest are two autonomous driving simulation platforms named MetaDrive [9] and SMARTS [10].

TABLE I
PARAMETERS FOR PPO, PPOLC, LPPO

Parameter Names	Parameter Values
γ(discount Factor)	0.995
λ(for GAE)	0.95
ϵ(PPO's Clipping Coefficient)	0.2
η(Entropy Regularization Coefficient)	0.01
ω(Gradient Norm)	0.5
K(Number of PPO Update Iteration Per Epoch)	10
l(Learning Rate)	0.0001

TABLE II
PARAMETERS FOR LDQN

Parameter Names	Parameter Values
discount Factor	0.995
Learning starts	10000
buffer size	1000000
target network update freq	10000
train batch size	32
exploration initial epsilon	1
exploration final epsilon	0.02
epsilon timesteps	200000

We design multiple tasks to demonstrate that LPPO is not limited to a particular scenario or objective setting.

The comparison approaches consist of single-objective and multi-objective algorithms. The compared single-objective algorithm is PPO(PPO Reward Shaping, PPORS). When realizing PPO, we combine multiple weighted sub-rewards into a total reward. Multi-objective algorithms include PPO Linear Combination (PPOLC) and Thresholded Lexicographical Ordering DQN (LDQN). PPOLC is a multi-branch critic architecture that calculates the advantages for each objective. Then, the advantages are normalized and finally combined to output a total advantage. LDQN is a value-based lexicographical method that utilizes the Q-value threshold.

We deploy PPO, PPOLC, LPPO, and LDQN based on Ray [37] and RLlib [38]. With the distributed characteristics of the ray framework, we use six workers to train each algorithm in parallel. Each algorithm requires five random seeds, so we need 30 workers in parallel for one experiment. We perform experiments with Intel(R) Xeon(R) CPU E5-2620 v4 and a NVIDIA 2080 GPU. The CPU E5-2620 v4 has 32 threads, which can support our efficient testing of an algorithm. Other 32-thread CPUs can also be used to realize our experiments. Related parameters are illustrated in Table I and Table II.

A. Toy Experiments in Safety-Gym

In order to show the solution sets of LPPO and other algorithms under different preference parameters, we conduct experiments in safety-gym [36] environment. Safety-gym is a robot control environment and focuses on safety exploration. We evaluate the algorithms in the CarGoal1 task that involves moving the robot car to a progression of goal positions. At each step, the robot car receives a penalty signal if it encounters an obstacle.

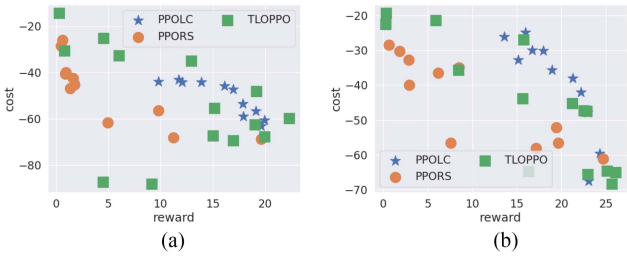

Fig. 4. Pareto fronts found by LPPO and other methods for safety-gym CarGoal1 task. Each point represents the average results under five random seeds. (a) mid-training. (b) end of training.

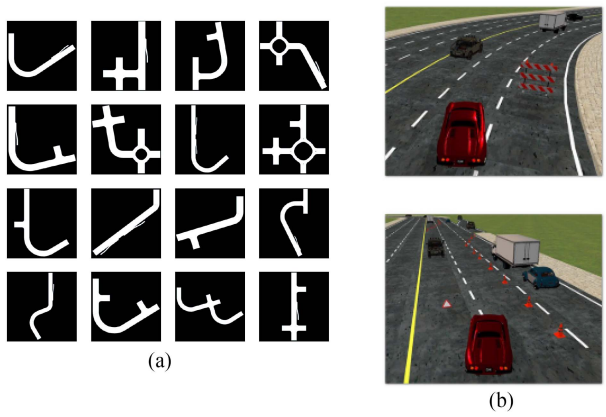

Fig. 5. (a) Example maps in the MetaDrive environment. (b) The interface of the environment from MetaDrive. In the environment, there exist different kinds of obstacles, i.e., warning triangle, cone, or other vehicles. (a) Map. (b) Interface.

We take the car's forward reward and collision penalty as two research objectives in the experiment. We use different preference parameters to observe the Pareto fronts found by the approaches.

Fig. 4 shows that the Pareto front found by LPPO is superior to the one found by PPORS. In particular, LPPO finds more policies that perform well with respect to both objectives, i.e., green dots are in the upper right portion of the Pareto front relative to the yellow point. In addition, the results also illustrate that LPPO outperforms PPOLC in this experiment.

B. Obstacle Avoidance Experiments in MetaDrive

We conduct vehicle obstacle avoidance experiments in the MetaDrive environment. MetaDrive is a driving simulation platform that can compose a wide range of scenarios with various road networks and traffic flow. In this experiment, we use 100 scene maps for training, some example maps are shown in Fig. 5(a). At the beginning of each episode, the map is randomly selected from the 100 scene maps. We randomly initialize the positions of the static and dynamic obstacles in the scene, as

shown in Fig. 5(b), the car is expected to reach the end point from the starting point without colliding with the obstacles.

1) State Space: The state space of the environment contains Lidar-like cloud points, the state of the target vehicle, and the navigation information. The Lidar-like cloud points are represented by a 240-dimensional vector with 50 m maximum detecting distance centering at the target vehicle. The target vehicle's state consists of steering, heading, velocity, and relative distance to the left and right boundaries. The navigation information guides the target vehicle toward the destination.

2) Reward Design: The purpose of the experiment is to make the self-driving vehicle reach the end point from the starting point. During driving, the vehicle is also designed to avoid collisions with warning triangle, cone or other vehicles.

In this experiment, we design the task reward and punishment cost to satisfy the above purpose. The task reward is composed of three parts as follows:

$$R_{task} = c_1 R_{disp} + c_2 R_{speed} + c_3 R_{term}, \tag{25}$$

where $R_{disp} = d_t - d_{t-1}$ and $R_{speed} = v_t / v_{max}$. R_{disp} represents displacement reward. d_t and d_{t-1} denote the longitudinal coordinates of the target vehicle in the current lane of two consecutive time steps. R_{speed} encourages the agent to drive fast. v_t and v_{max} are the current velocity and the maximum velocity (120 km/h). R_{term} is a sparse reward, which is non-zero only at the last time step. The punishment cost is -1 when the agent collides with vehicles, obstacles, sidewalks, and buildings, or when the car leaves the road.

3) Evaluation Metrics: We use success rate as a final evaluation metric. It defines the ratio of episodes where the agent arrives at the destination in multiple episodes. In order to achieve a high success rate, we need to deal with the balance between task reward and cost, and in other words, we need to have a suitable preference coefficient.

4) Parameter Details: We experiment with a series of preference parameters on various algorithms. For PPO, we first fix the c_{task} to 1, while $c_{cost} \in [0.0, 1.0, 3.0, 5.0, 10.0, 15.0]$. For PPOLC, the preference coefficient is used to encode the relationship between the two advantage values. We fix the c_{task} to 1, while $c_{cost} \in [0.0, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2]$. For LDQN, we take task reward as the first priority and cost as the second priority, while $\tau \in [0.1, 0.2, 0.5, 1.0, 2.0]$. For LPPO, we also take task reward as the first priority and cost as the second priority, while $\tau \in [0.05, 0.1, 0.15, 0.2, 0.3, 0.4]$.

5) Result Analysis: We compare LPPO and other algorithms according to the parameter settings in the previous section. The relevant experimental results are shown in Fig. 6. The right images in Fig. 6 show that as the preference parameters increase, the final success rate gradually increases and then decreases. Note that the preference parameter is relevant to the importance of the cost. At the beginning when the preference parameters are small, it indicates that the cost is not important, thus resulting in a high possibility of collision and a low success rate. As the preference parameters gradually increase, the success rate also increases accordingly. However, large preference parameters would make the vehicle too cautious and difficult to move forward, thus leading to a decrease in the success rate. We can find that the success rate of LPPO exceeds the performance of other algorithms under some preference parameters when $\tau \in [0.1, 0.2, 0.3]$. At the same time, the preference parameter adjustment space of LPPO is smaller than other algorithms. For PPO, PPOLC and LDQN, the coefficient adjustment space is $[0, \infty]$, while LPPO is [0,1]. In summary, LPPO needs fewer experiments to reflect humans preferences and can be better deployed in practical problems.

We select various algorithms corresponding to the preference parameters with the best success rate to compare task reward and

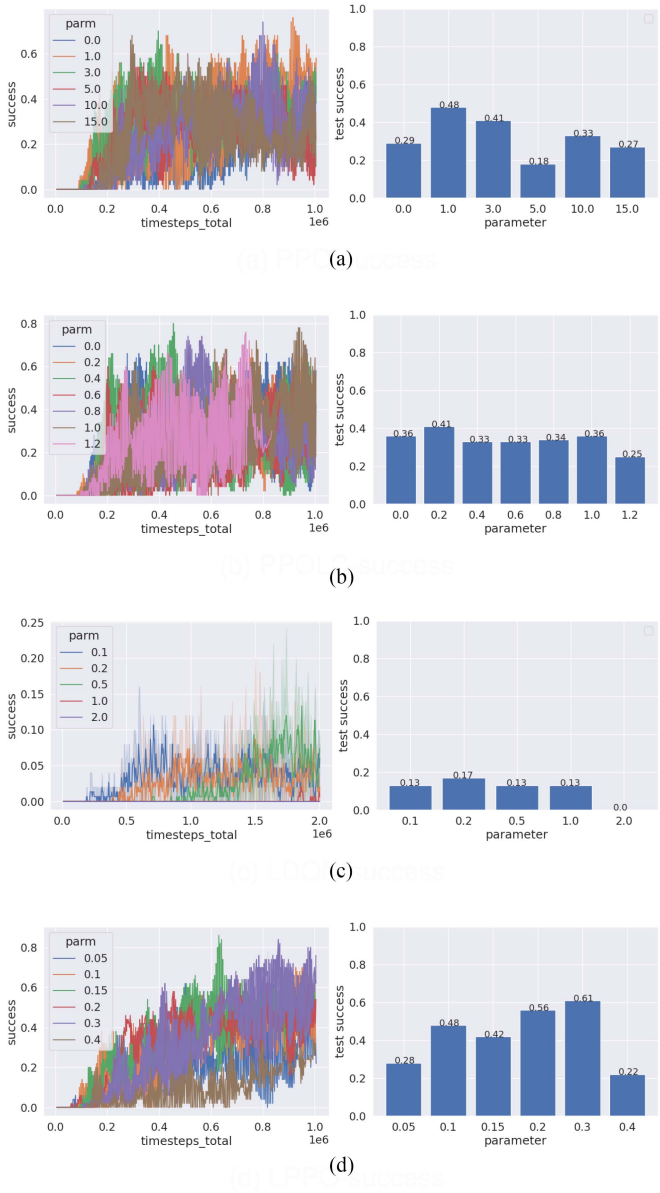

Fig. 6. The success rate of different algorithms in the metadrive environment. The left figures represent the change curve of the success rate during the training period under different preference parameters, and the right figures are the corresponding test success rate. (a) PPO-success. (b) PPOLC-success. (c) LDQN-success. (d) LPPO-success.

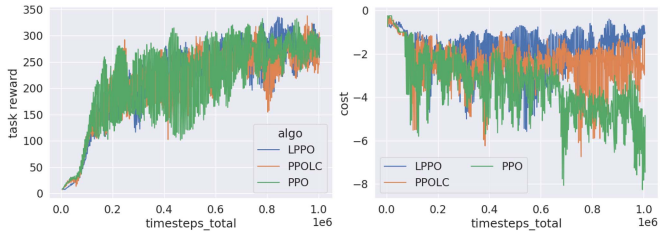

Fig. 7. The results of different hyperparameter settings on algorithm performance. We ignore LDQN because it achieves low success rate in Fig. 6.

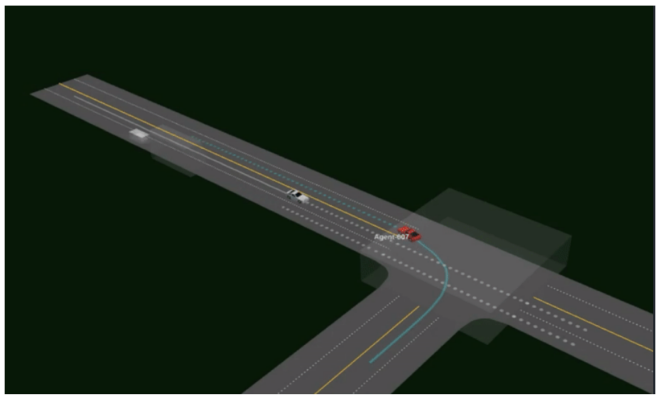

Fig. 8. Unprotected left Intersection in SMARTS. The red agent car is forced to turn left onto a major road by crossing a two-way four-lane highway.

TABLE III
DISCRETE CONTROL

Discrete control name	throttle	brake	steer
Coast	0	0	0
Turn Left	0	0	-0.35
Turn Right	0	0	0.35
Forward	0.85	0	0
Brake	0	0.35	0
Bear Left and accelerate	0.7	0	-0.35
Bear Right and accelerate	0.7	0	0.35
Bear Left and decelerate	0	0.6	-0.35
Bear Right and decelerate	0	0.6	0.35

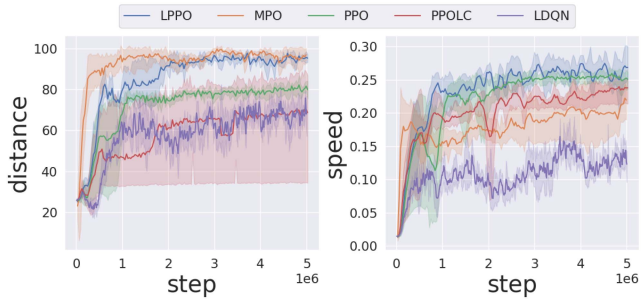

Fig. 9. The x-axis represents step, and the y-axis represents distance or speed. The blue line represents our method LPPO, which finds the most optimum solution for both distance and speed objectives compared to other algorithms. Each algorithm trains through four random seed. Error bars show standard error across these seeds.

cost as shown in Fig. 7. The figure shows that various algorithms have similar performance in task rewards, but LPPO is better and more stable in terms of the cost objective.

C. Experiments in SMARTS

In the SMARTS simulation environment, we construct a unprotected intersection scene as shown in Fig. 8, a line merging scene and a straight scene as shown in Fig. 15. When a self-driving vehicle passes through an unprotected intersection, it needs to pay attention to the surrounding environment to avoid debuting and collision with other social vehicles. Under the premise of ensuring safety, it is hoped that it can pass quickly without causing detention and affecting the passage of other vehicles. In the single-direction scene, the agent is forced to turn left onto a major road by crossing a two-way four-lane highway and reaching the rightmost lane with no traffic light regulation. We consider two competing objectives: travel distance and speed. That is to say, when the vehicle cares more about reaching the destination safely, it will reduce its speed and increase the driving distance respectively, and vice versa.

1) State Space and Action Space in SMARTS: The state space can be divided into three categories: ego information, map information, and surrounding social vehicle information. The ego information includes the current speed and steering angles of the vehicle. While the map information contains the distance from the center line, the way point in front of the lane, and the angle deviation from the predetermined route. The surrounding vehicle information includes the distance of the nearest vehicle in each lane, the time of possible collision, the speed of social vehicles, and a bird's-eye view gray-scale image with the agent at the center. For the action space, it receives actions for throttle, brake, and steering angle where the steering of the vehicle is provided by an Ackermann steering mechanism. As shown in Table III, in order to reduce the action space so that the agent can learn quickly, we discretize the continuous control variables of steering wheel, accelerator, and brake into nine actions to form the action space.

Fig. 9 shows that LPPO achieves good performance at the two competing objectives, while the rest cannot. Although MPO [39] can output competitive results at the first objective, LPPO outperforms MPO by a large margin at the second objective. PPO has a similar performance, doing well on the second objective but not so good on the first objective. The reason is that as a single-objective method, PPO and MPO over-optimizes one objective during the training process but ignores the other one. In addition, multi-objective methods cannot solve the tasks properly either. LDQN does not work well on the distance and speed objective because the scope of the Q-value changes dramatically during training, and the predefined threshold value cannot match these changes well enough. PPOLC also cannot balance the relative importance of the two objectives when facing conflicting objectives.

2) Evaluation When Reward Scale Changes: In this experiment, we change the reward scale by multiplying the speed reward by 10 while leaving the distance reward unchanged.

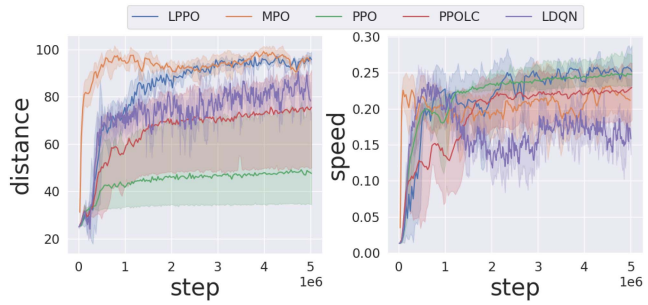

Fig. 10. Comparison of the performance of LPPO and other algorithms under unbalanced rewards.

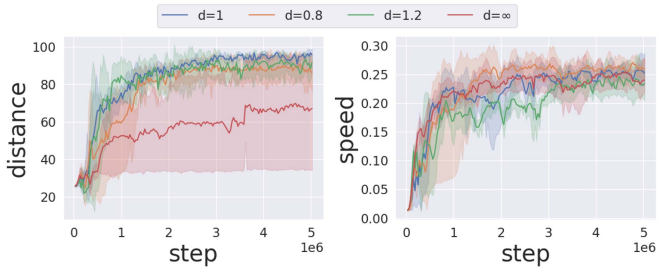

Fig. 11. The results of different objective clip hyperparameter settings on algorithm performance.

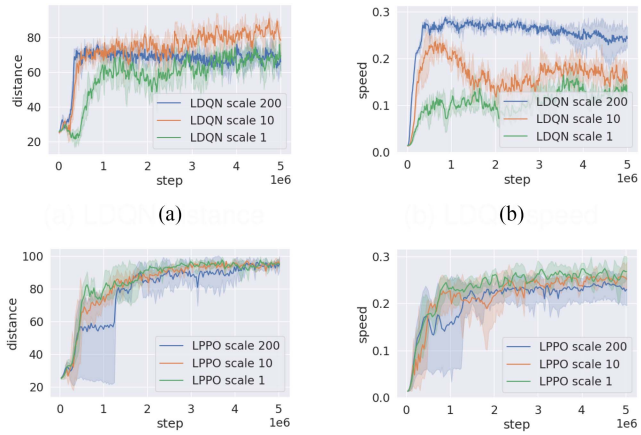

Fig. 12. The performance of LPPO and LDQN under different sacle of the speed reward. (a) LDQN-distance. (b) LDQN-speed. (c) LPPO-distance. (d) LPPO-speed.

TABLE IV
TEST PERFORMANCE OF DIFFERENT ALGORITHMS IN SINGLE-DIRECTION INTERSECTION

Algorithm	LPPO	LDQN	PPOLC	PPO	MPO
Distance	97.15	81.3	89.47	88.34	96.46
Speed	0.281	0.278	0.273	0.256	0.222

Intuitively, the agent tends to output a shorter distance when the speed objective is fed with a larger reward.

When comparing Fig. 9 and Fig. 10, we can draw the conclusion that our method is insensitive to the reward scale. The distance curve of LPPO remains almost unchanged when the speed reward scale changes, while the distance of PPO decreases dramatically. In terms of different speed reward scales, the comparison methods have different results. The figures illustrate that PPOLC is also insensitive to reward scale. The reason is that PPOLC tries to reduce the influence of the scale by normalizing the advantage value. However, it cannot achieve competitive results on both distance and speed objectives. The distance curve of MPO also keeps stable, but the speed curve is at a lower level. We deem that when MPO mixes distance and speed rewards, the incentive of speed rewards is negligible. Even if the speed scale increases, the distance reward remains to dominate the total reward. The curves of LDQN show a downward trend on the speed objective. The reason is that the original threshold setting of LDQN is related to the reward scale. When the reward scale changes, the threshold hyperparameter needs to be adjusted.

3) Evaluation on Objective Clip: To verify the effectiveness of objective clip and parameter d, we experiment with the training effect of LPPO under d of 0.8, 1, 1.2, and ∞. Note that ∞ means that objective clip is not deployed.

Fig. 11 shows objective clip is necessary for our method. When compared with the methods that adopt objective clip, the setting $d = \infty$ has a high variance in the driving distance curve. The reason is that the generated action distribution, which derives from interacting with the environment, is different from the output distribution of some actor branches.

When d is set to different values, the distance and speed curve of these settings are similar. The results also show that the hyperparameter d is not sensitive to the final performance within a specific reasonable range.

4) Comparison With Value Threshold: LDQN is a value threshold approach. Fig. 12 shows the performance of LDQN and LPPO under different speed reward scales $(1, 10, 200)$. We find that the performance of LDQN differs greatly under different speed reward scales. The designed threshold for the Q value is not applicable under some speed reward scales by observing the learning curve appears to decrease. However, the performance of LPPO is relatively stable.

5) Test Set Evaluation: We train 5 million steps for each algorithm and use the final trained network to test in an unprotected single-direction intersection scene. In order to avoid the impact of the randomness of the scene, each algorithm tests in 30720 steps. The final average performance result shows in Table IV. We can find that LPPO outperforms the other methods in retaining a long distance and a high speed.

6) Comparison in Multi-Direction Intersection: In order to test the robustness of our algorithm, we train in parallel at three directions in the multi-direction scene. Fig. 13(b) shows that the vehicles have three fixed directions: left, front, and right in the multi-direction scene.

Fig. 14 shows the training performance of different algorithms in multi-direction intersection scenarios. We can find that in addition to LDQN, other algorithms achieve similar and excellent results in distance objectives, and in terms of speed objectives, LPPO is better than other algorithms.

Fig. 13. Two intersection scenes. Note that compared with the single-direction scene, multi-direction scene is a more challenging task. (a) single-direction. (b) multi-direction.

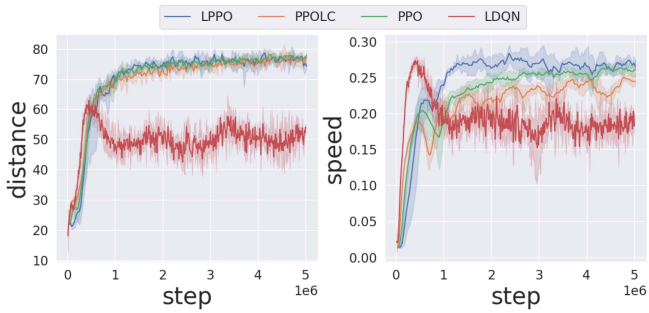

Fig. 14. The performance of different algorithms in Multi-direction Intersection.

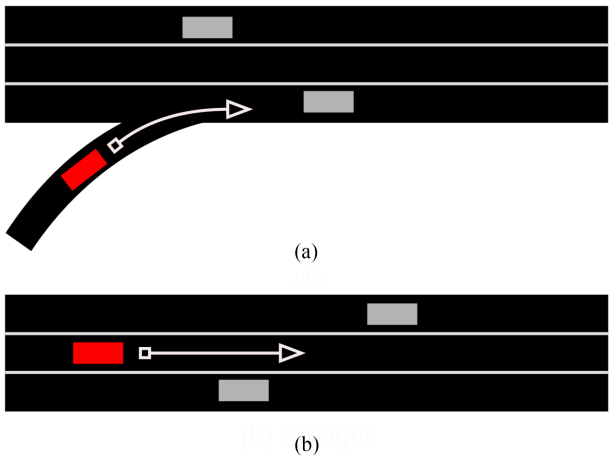

Fig. 15. Merge scene and straight scene in SMARTS. The red agent car is forced to cross a three-lane highway and reach the destination. (a) Merge. (b) Straight.

7) Comparison in Lane Merging and Straight Scenes: To further demonstrate the effectiveness of our algorithm, we also conduct experiments in the lane merging and straight scenes. As shown in Fig. 15, the agent can change lanes at any time during driving. We consider two objectives: travel distance and off-center distance. The agent vehicle should pass quickly without causing detention and affecting the passage of other vehicles. At the same time, the vehicle needs to follow the center-line of the road to avoid frequent lane changes.

Fig. 16 shows that LPPO can achieve the longest driving distance and the smallest off-center distance compared to other

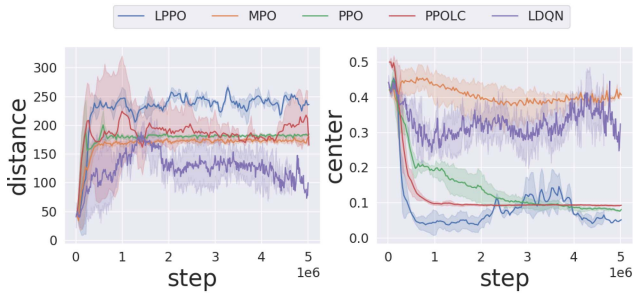

Fig. 16. The x-axis represents step, and the y-axis represents distance or off-center distance. For distance indicators, the larger, the better; for center indicator, the smaller, the better.

algorithms. This means LPPO can better balance driving distance and following lanes.

VI. Conclusion

In this paper, we propose the LAC architecture and design the corresponding LPPO algorithm for balancing the multiple rewards of autonomous vehicles. LPPO introduces TLO to sequentially define the importance of different objectives and utilizes the LAS mechanism to generate actions that interact with the environment. At the same time we propose objective clip and objective V-trace to update the networks stably. We find that LPPO requires less experiments to reflect humans preferences and can be better deployed in practical problems. Experimental comparisons show that our method outperforms the other methods in two autonomous driving simulation platforms named MetaDrive and SMARTS. Meanwhile, our method can also output competitive results when the reward scale changes.

A limitation of this work is that LPPO can only be used in discrete action spaces. In the future, we will try to utilize TLO for the actor-critic algorithm under continuous action space. Then, we will attempt to solve more complex and challenging problems.

References

[1] H. Shu, T. Liu, X. Mu, and D. Cao, "Driving tasks transfer using deep reinforcement learning for decision-making of autonomous vehicles in unsignalized intersection," *IEEE Trans. Veh. Technol.*, vol. 71, no. 1, pp. 41–52, Jan. 2022.

[2] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3826–3839, Sep. 2020.

[3] S. Chen, M. Wang, W. Song, Y. Yang, Y. Li, and M. Fu, "Stabilization approaches for reinforcement learning-based end-to-end autonomous driving," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 4740–4750, May 2020.

[4] C. Huang, R. Mo, and C. Yuen, "Reconfigurable intelligent surface assisted multiuser miso systems exploiting deep reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 8, pp. 1839–1850, Aug. 2020.

[5] C. Huang et al., "Multi-hop RIS-empowered terahertz communications: A DRL-based hybrid beamforming design," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 6, pp. 1663–1677, Jun. 2021.

[6] Z. Gábor, Z. Kalmáe, and C. Szepesvári, "Multi-criteria reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 1998, pp. 197–205.

[7] L. E. Pineda, K. H. Wray, and S. Zilberstein, "Revisiting multi-objective MDPs with relaxed lexicographic preferences," in *Proc. AAAI Fall Symp. Ser.*, 2015, pp. 63–68.

[8] K. H. Wray and S. Zilberstein, "Multi-objective POMDPs with lexicographic reward preferences," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, 2015, pp. 1719–1725.
[9] Q. Li, Z. Peng, Z. Xue, Q. Zhang, and B. Zhou, "Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2022.
[10] M. Zhou et al., "Smarts: Scalable multi-agent reinforcement learning training school for autonomous driving," in *Proc. Conf. Robot Learn.*, 2020, pp. 1–20.
[11] M. Toromanoff, E. Wirbel, and F. Moutarde, "End-to-end model-free reinforcement learning for urban driving using implicit affordances," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 7153–7162.
[12] X. Liang, T. Wang, L. Yang, and E. Xing, "CIRL: Controllable imitative reinforcement learning for vision-based self-driving," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 584–599.
[13] C. Huang et al., "Deductive reinforcement learning for visual autonomous urban driving navigation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 12, pp. 5379–5391, Dec. 2021.
[14] Y. Fu, C. Li, F. R. Yu, T. H. Luan, and Y. Zhang, "A decision-making strategy for vehicle autonomous braking in emergency via deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 6, pp. 5876–5888, Jun. 2020.
[15] Y. Song, H. Lin, E. Kaufmann, P. Duerr, and D. Scaramuzza, "Autonomous overtaking in gran turismo sport using curriculum reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 9403–9409.
[16] P. Cai, X. Mei, L. Tai, Y. Sun, and M. Liu, "High-speed autonomous drifting with deep reinforcement learning," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 1247–1254, Apr. 2020.
[17] Z. Qiao, Z. Tyree, P. Mudalige, J. Schneider, and J. M. Dolan, "Hierarchical reinforcement learning method for autonomous vehicle behavior planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 6084–6089.
[18] F. M. Favaró, N. Nader, S. O. Eurich, M. Tripp, and N. Varadaraju, "Examining accident reports involving autonomous vehicles in California," *PLoS one*, vol. 12, no. 9, 2017, Art. no. e0184952.
[19] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura, "Navigating occluded intersections with autonomous vehicles using deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 2034–2039.
[20] E. Mohamed, H. Kimia, N. M. Nguyen, and A. Montgomery, "Ultra: A reinforcement learning generalization benchmark for autonomous driving," in *Proc. Adv. Neural Inf. Process. Syst. Auton. Driving Workshop*, 2020.
[21] K. Shu et al., "Autonomous driving at intersections: A critical-turning-point approach for left turns," in *Proc. IEEE 23rd Int. Conf. Intell. Transp. Syst.*, 2020, pp. 1–6.
[22] P. Vamplew, R. Dazeley, A. Berry, R. Issabekov, and E. Dekker, "Empirical evaluation methods for multiobjective reinforcement learning algorithms," *Mach. Learn.*, vol. 84, no. 1, pp. 51–80, 2011.
[23] A. Abels, D. Roijers, T. Lenaerts, A. Nowé, and D. Steckelmacher, "Dynamic weights in multi-objective deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 11–20.
[24] N. D. Nguyen, T. T. Nguyen, P. Vamplew, R. Dazeley, and S. Nahavandi, "A prioritized objective actor-critic method for deep reinforcement learning," *Neural Comput. Appl.*, pp. 1–15, 2021.
[25] P. Vamplew, J. Yearwood, R. Dazeley, and A. Berry, "On the limitations of scalarisation for multi-objective reinforcement learning of pareto fronts," in *Proc. Australas. Joint Conf. Artif. Intell.*, 2008, pp. 372–378.
[26] C. Li and K. Czarnecki, "Urban driving with multi-objective deep reinforcement learning," in *Proc. Int. Conf. Auton. Agents Multiagent Syst.*, 2019, pp. 359–367.
[27] K. Van Moffaert, M. M. Drugan, and A. Nowé, "Scalarized multi-objective reinforcement learning: Novel design techniques," in *Proc. IEEE Symp. Adaptive Dyn. Program. Reinforcement Learn.*, 2013, pp. 191–199.
[28] A. Abdolmaleki et al., "A distributional view on multi-objective policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 11–22.
[29] R. S. Sutton et al., "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Neural Inf. Process. Syst.*, 1999, vol. 99, pp. 1057–1063.
[30] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[31] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proc. Int. Conf. Learn. Representations*, 2016.
[32] L. Espeholt et al., "Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1407–1416.
[33] S. Han and Y. Sung, "Dimension-wise importance sampling weight clipping for sample-efficient reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2586–2595.
[34] D. Ye et al., "Mastering complex control in moba games with deep reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, pp. 6672–6679.
[35] S. Huang and S. Ontañón, "A closer look at invalid action masking in policy gradient algorithms," in *Proc. 35th Int. Florida Artif. Intell. Res. Soc. Conf.*, 2022.
[36] A. Ray, J. Achiam, and D. Amodei, "Benchmarking safe exploration in deep reinforcement learning," 2019, *arXiv:1910.01708*.
[37] P. Moritz et al., "Ray: A distributed framework for emerging {AI} applications," in *Proc. 13th {USENIX} Symp. Operating Syst. Des. Implementation ({OSDI} 18)*, 2018, pp. 561–577.
[38] E. Liang et al., "RLlib: Abstractions for distributed reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 3053–3062.
[39] A. Abdolmaleki, J. T. Springenberg, Y. Tassa, R. Munos, N. Heess, and M. Riedmiller, "Maximum a posteriori policy optimisation," in *Proc. Int. Conf. Learn. Representations*, 2018.

Hengrui Zhang received the B.S. degree in computer science and technology from Liaoning Technical University, Liaoning, China, in 2018. He is currently working toward the Ph.D. degree in computer science with the School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China.

His research interests include reinforcement learning and urban driving.

Youfang Lin received the Ph.D. degree in signal and information processing from Beijing Jiaotong University, Beijing, China, in 2003.

He is currently a Professor with the School of Computer and Information Technology, Beijing Jiaotong University. His main research interests include Big Data technology, intelligent systems, complex networks, and traffic data mining.

Sheng Han received the master's degree in software engineering from Beijing Jiaotong University, Beijing, China, in 2006.

He is currently an Associate Professor with the School of Computer and Information Technology, Beijing Jiaotong University. His main research interests include Big Data technology, reinforcement learning, and intelligent system.

Kai Lv received the Ph.D. degree from the School of Computer Science and Engineering, Beihang University, Beijing, China, in 2021. He is currently a Postdoctoral Researcher with the School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China. His main research interests include computer vision, reinforcement learning, and machine learning. He is also the corresponding author of this paper.