LPPG-RL: Lexicographically Projected Policy Gradient Reinforcement Learning with Subproblem Exploration

Ruiyu Qiu[1*], Rui Wang[2*], Guanghui Yang[1,3], Xiang Li[1], Zhijiang Shao[1,3†]

[1]College of Control Science and Engineering, Zhejiang University, Hangzhou, China
[2]School of Mechanical Engineering and Mechanics, Ningbo University, Ningbo, China
[3]Huzhou Institute of Industrial Control Technology, Huzhou, China
ryqiu@zju.edu.cn,wangrui2@nbu.edu.cn,{ghyang,li.xiang,szj}@zju.edu.cn

Abstract

Lexicographic multi-objective problems, which consist of multiple conflicting subtasks with explicit priorities, are common in real-world applications. Despite the advantages of Reinforcement Learning (RL) in single tasks, extending conventional RL methods to prioritized multiple objectives remains challenging. In particular, traditional Safe RL and Multi-Objective RL (MORL) methods have difficulty enforcing priority orderings efficiently. Therefore, Lexicographic Multi-Objective RL (LMORL) methods have been developed to address these challenges. However, existing LMORL methods either rely on heuristic threshold tuning with prior knowledge or are restricted to discrete domains. To overcome these limitations, we propose Lexicographically Projected Policy Gradient RL (LPPG-RL), a novel LMORL framework which leverages sequential gradient projections to identify feasible policy update directions, thereby enabling LPPG-RL broadly compatible with all policy gradient algorithms in continuous spaces. LPPG-RL reformulates the projection step as an optimization problem, and utilizes Dykstra's projection rather than generic solvers to deliver great speedups, especially for small- to medium-scale instances. In addition, LPPG-RL introduces Subproblem Exploration (SE) to prevent gradient vanishing, accelerate convergence and enhance stability. We provide theoretical guarantees for convergence and establish a lower bound on policy improvement. Finally, through extensive experiments in a 2D navigation environment, we demonstrate the effectiveness of LPPG-RL, showing that it outperforms existing state-of-the-art continuous LMORL methods.

Code — https://github.com/qiuruiyu/LPPG-RL

Introduction

Reinforcement learning (RL) has achieved remarkable success in diverse domains, including games (Silver et al. 2014; Mnih et al. 2015), robotics (Tang et al. 2025), and autonomous driving (Coelho, Oliveira, and Santos 2024). In real-world applications, complex tasks are naturally decomposed into multiple subtasks, often organized according to explicit, predefined priorities. For example, in autonomous driving, the lane-changing task can be divided into subtasks with descending priorities: collision avoidance, lane keeping, and speed regulation. These subtasks may conflict with each other, making the overall task challenging, even though each subtask is individually simple.

In such scenarios, safety is typically assigned the highest priority. Prior work on Safe RL formulates constraints within the framework of Constrained Markov Decision Processes (CMDPs) (Altman 2021), where the agent maximize reward while minimizing cost. However, existing approaches exhibit significant limitations. First, policy optimization is generally conducted over an unordered set of constraints, which results in infeasibility when constraints are conflicting. Second, traditional Safe RL methods are not designed to optimize multiple objectives concurrently.

If the safety cost is treated as an additional objective, Safe RL can be regarded as a special case of Multi-Objective RL (MORL). MORL seeks to identify optimal policies that balance trade-offs among multiple objectives (Qiu et al. 2025), typically by approximating the Pareto front. Under this perspective, Safe RL corresponds to selecting a Pareto-optimal policy that satisfies constraints. Although this concept is appealing, MORL methods usually aggregate objectives into a scalar, disregarding explicit priorities. Moreover, extending MORL approaches to densely approximate the Pareto front is computationally expensive, as it requires repeated training with varying weights (Parisi et al. 2014). When priorities are predefined, practitioners must manually assign extreme weights and search through the Pareto front, resulting in sample inefficiency and offering no guarantees that higher-priority objectives are maintained (Liu et al. 2025).

To address these limitations, we focus on directly finding a feasible policy by leveraging Lexicographic Multi-Objective Reinforcement Learning (LMORL). Building upon the framework introduced in (Skalse et al. 2022), we extend it from a policy gradient perspective. Specifically, we propose the Lexicographically Projected Policy Gradient RL (**LPPG-RL**), which sequentially searches for a feasible policy update by identifying the intersection of high-priority gradients, thereby enforcing the priority structure. Furthermore, we introduce several key improvements to enhance both practicality and efficiency. First, recognizing the computational complexity associated with large network parameters, we formulate an optimization problem and employ a problem-specific iterative projection method rather than re-

[*]These authors contributed equally.
[†]Corresponding author.

lying on generic solvers. This approach significantly accelerates gradient projection, particularly in small- to medium-scale problems. Second, to avoid excessive hyperparameters and gradient vanishing, we encourage Subproblem Exploration (SE), which expedites the training and convergence of individual subtasks. Importantly, our method is naturally applicable to continuous spaces with standard policy gradient RL algorithms such as Proximal Policy Optimization (PPO) (Schulman et al. 2017) and Soft Actor-Critic (SAC) (Haarnoja et al. 2018), whereas most previous methods are restricted to discrete domains. Finally, we provide theoretical analysis of the convergence and the policy improvement lower bound for our algorithm.

We evaluate our algorithm using PPO (LPPG-PPO) on variants of a 2D navigation environment. We compare the computational efficiency of our projection method with generic solvers within CVXPY (Diamond and Boyd 2016), and benchmark our performance against two state-of-the-art LMORL methods for continuous spaces: Lexicographic Projection Algorithm (LPA) (Tercan and Prabhu 2024) and Lexicographic PPO (LPPO) (Skalse et al. 2022). Experiments show that our method achieves higher efficiency with equivalent results when the number of objectives is below 100, and LPPG-RL outperforms baseline methods in both performance and stability. Ablation studies further demonstrate the effectiveness of our proposed enhancements.

Our contributions of this work are three-fold:

1. We propose LPPG-RL, a novel LMORL algorithm for continuous spaces that guarantees strict priority ordering while training a policy end-to-end. With an ultra-light Dykstra's projection core, we achieve up to $20\times$ greater computational efficiency compared to generic solvers.
2. We introduce Subproblem Exploration (SE), a uniform rollout scheduler that dynamically focuses on the active priority layer, requiring no handcrafted weights or prior knowledge. SE achieves superior performance and training stability compared to existing baseline methods.
3. We provide theoretical guarantees of convergence, including a principled stepsize bound that ensures monotonic policy improvement, and establish a lower bound on performance improvement at each update.

Related Works

Safe RL

Safe RL is typically formulated as a constrained optimization problem, and various techniques have been developed to handle safety constraints. Primal-dual (Lagrangian) methods are widely used by introducing penalty terms with tunable coefficients (Tessler, Mankowitz, and Mannor 2018; Stooke, Achiam, and Abbeel 2020). However, selecting appropriate coefficients remains challenging in practice. Constrained Policy Optimization (CPO) (Achiam et al. 2017) addresses this issue by solving a trust-region problem at each step, but relies on accurate cost estimation and brings high computational complexity. To reduce this burden, Projection-based CPO (PCPO) (Yang et al. 2020) simplifies the process by directly projecting policy gradients onto the feasible constraint set. In addition, explicit Lyapunov functions (Chow et al. 2018, 2019) and control barrier functions (Dawson, Gao, and Fan 2023; Deng et al. 2024) can be employed to provide formal safety guarantees. Despite the significant progress in Safe RL, explicitly ordered constraints remain largely unsolved. In such cases, constraints must be handled sequentially, which substantially increases computational costs.

Multi-Objective RL

MORL methods can be categorized into single-policy and multi-policy approaches. Single-policy methods combine multiple objectives into a single scalar reward, using weighted sums (Roijers, Whiteson, and Oliehoek 2014; Abels et al. 2018), which require significant domain knowledge and extensive tuning (Van Moffaert, Drugan, and Nowé 2013). Multi-policy methods approximate the Pareto front by repeatedly applying single-policy methods with different weight configurations (Mossalam et al. 2016; Zuluaga, Krause, and Püschel 2016). To improve efficiency, evolutionary algorithms (Xu et al. 2020) and Pareto Set Learning (Liu et al. 2025) have also been explored to help approximate the Pareto front. However, when explicit priorities are required, these methods often fall short. Lexicographic optimality demands filtering policies along a densely sampled and accurate Pareto front, which leads to computational and scalability issues as the number of objectives increases (Parisi et al. 2014; Pirotta, Parisi, and Restelli 2015). Therefore, existing MORL methods excel at balancing objectives but remain inefficient for strictly prioritized problems.

Lexicographic Multi-Objective RL

LMORL was first explored through lexicographic optimization by (Gábor, Kalmár, and Szepesvári 1998). Later, a unified thresholded framework with theoretical analysis was proposed for both value-based and policy-based RL methods (Skalse et al. 2022). However, the value-based framework—followed by works in autonomous driving(Li and Czarnecki 2019; Zhang et al. 2023)—filters actions by enumerating all candidates in each subtask, which restricts applications to discrete action spaces. (Rietz et al. 2024) extended the policy-based framework to Q-value, but this approach can fail when subtask Q-values differ in scale, and actually relies on a costly grid search to estimate continuous values. Recently, (Tercan and Prabhu 2024) introduced Lexicographic Projection Algorithm (LPA) to address continuous problems. However, to our understanding, both LPA and the original policy-based framework requires setting optimization and criteria thresholds according to the reward scale, demanding prior knowledge and repeated tuning, which makes methods quite sensitive and less practical. In contrast, we overcome these limitations by recasting lexicographic learning as a sequence of constrained gradient projections solved with an efficient Dykstra–style update, and by introducing SE to automatically balance subtask convergence instead of tuning thresholds with prior knowledge.

Problem Definition and Preliminaries

Suppose there are M subtasks, including objectives and constraints, defined in a lexicographic order and denoted by the

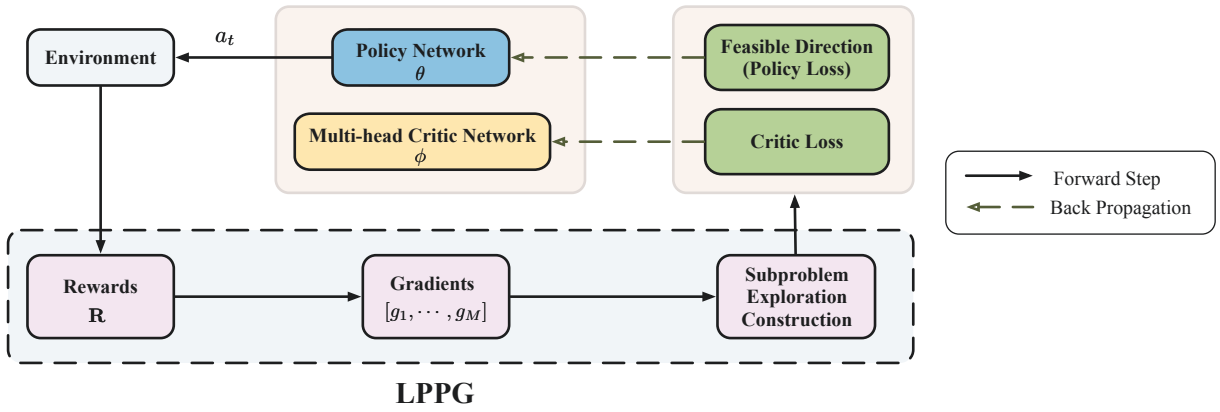


Figure 1: The overall workflow of LPPG-RL. The agent consists of a policy network and a multi-head critic network, where the policy network learns the global optimal lexicographic action. The lower part is LPPG. Policy gradients are calculated each time with rollouts, and a subproblem is drawn to get a feasible update direction so that each subtask can be trained uniformly.

task set $\mathcal{K} = \{K_1, \cdots, K_M\}$, where the subscript indicates priority (with K_1 the highest priority).

An LMORL problem can be formulated with Multi-Objective Markov Decision Processes (MOMDPs) (Skalse et al. 2022), represented as $\mathcal{M} := \langle S, A, \mathbf{R}, P, \boldsymbol{\gamma} \rangle$. Here, S and A denote the state and action spaces, respectively, and P is the state transition probability. The reward function $\mathbf{R} : S \times A \times S \to \mathbb{R}^M$ produces a rewards vector $[r_1, \cdots, r_M]^T$ of $\mathcal{K}$. $\boldsymbol{\gamma}$ is the vector of discount factors.

Following the definition, lexicographic relationships between subtasks are enforced through constraints. Formally, a policy π is said to be lexicographically ϵ-optimal within the policy set Π_i, if subtasks are finished and it satisfies:

$$\Pi_{i+1} := \left\{ \pi \in \Pi_i \mid \max_{\pi' \in \Pi_i} J_i(\pi') - J_i(\pi) \leq \epsilon_i \right\} \tag{1}$$

where Π_0 is the initial policy set and Π_i is the feasible policy set for subtasks $\{K_1, \ldots, K_i\}$. When optimizing a subtask K_{i+1}, all higher priorities $\{K_1, \cdots, K_i\}$ must remain sufficiently close to their previous optima within a threshold ϵ_i. For any subtask index i and policy π, $J_i(\pi)$ is the performance criterion, such as the value function. It is important to note that, directly computing Π_{i+1} via Equation 1 is generally intractable, especially in continuous domains.

LPPG-RL

In this section, we present LPPG-RL, an efficient end-to-end framework for solving LMORL problems in continuous spaces. The overall workflow of LPPG-RL is illustrated in Figure 1. Given an LMORL problem with an ordered subtask set $\mathcal{K}$, our framework proceeds as follows:

- **Actor-Critic Architecture**: We adopt the actor-critic structure, where a policy network generates the final feasible action and a multi-head critic network estimates the value for each subtask.
- **Data Collection**: Rollouts are collected following standard RL procedures, with the reward for each subtask stored separately.
- **Subtask Gradients**: The policy gradient for each individual subtask is computed separately.
- **Subproblem Extraction**: To promote systematic exploration and expedite learning, SE is introduced as a uniform rollout scheduler to extract a subproblem and allocate trajectories evenly across priority levels, reducing the chance of local optima.
- **Feasible Direction and Update**: A feasible policy update direction d^* is computed based on the SE scheduler to update the policy. Instead of invoking generic solvers, we employ a Dykstra projection loop, an ultra-light iterative algorithm that provably converges to the optimum with significantly reduced computational burden.

In the following, we first construct the LMORL problem with policy gradients. Next, we discuss critical challenges during training and introduce techniques to address issues and enhance performance. Finally, we instantiate our framework with LPPG-PPO and provide theoretical analysis.

Projected Gradient for Lexicographic Tasks

Policy gradient methods in standard RL aim to maximize the expected cumulative reward by repeatedly estimating the gradient, which can be formulated as (Schulman et al. 2015):

$$g = \mathbb{E}\left[\sum_{t=0}^{\infty} \Psi_t \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right] \tag{2}$$

where Ψ_t can be the reward, Q-value, advantage, etc. A key advantage of using gradients is that it requires only the evaluation and differentiation of the policy log-density rather than enumerating all possible actions. This property facilitates a natural extension to multi-objective settings by directly operating on subtask-specific gradients. Since all gradients share a same parameter space, lexicographic order over the objectives can be straightforwardly imposed.

For each subtask $K_i \in \mathcal{K}$, let J_i denote its optimization objective and $g_i = \nabla_\theta J_i$ the corresponding gradient. At each iteration, the policy parameters update as follows:

$$\theta_{k+1} = \theta_k + \alpha d \tag{3}$$

where $\alpha > 0$ is the learning rate and d is the chosen update direction. Then, interpreting Equation 1 from a policy gradient perspective, we require that the objective J_i does not decrease when optimizing for lower-priority subtasks. This can be captured by a first-order (linear) approximation:

$$\begin{aligned} & J_i(\theta_{k+1}) \approx J_i(\theta_k) + \alpha g_i^T d \geq J_i(\theta_k) - \alpha\epsilon_i \\ \Rightarrow \quad & g_i^T d \geq -\epsilon_i \quad (\epsilon_i = 0 \text{ by default}) \end{aligned} \tag{4}$$

where θ_k is the network parameter at iteration k, and $\epsilon_i \geq 0$ serves as a small threshold for allowable degradation and helping convergence in the i-th subtask (Rietz et al. 2024). Clearly, if the chosen update direction d has a negative component along the original gradient g_i, the corresponding performance may deteriorate. Therefore, to achieve a lexicographically optimal update, we seek a direction d that satisfies as many conditions in Equation 4 as possible. Note that, we set $\epsilon_i = 0$ by default and rely on other techniques to maintain convergence. The threshold ϵ_i is only adjusted in specific cases where relaxing the constraint to address practical considerations.

Let $\mathcal{C}_M$ denote the intersection of half-spaces defined by Equation 4 for $i = \{1, \cdots, M\}$:

$$\mathcal{C}_M = \bigcap_{i=1}^{M} \left\{ d \mid g_i^T d \geq -\epsilon_i \right\} \tag{5}$$

Here, $\mathcal{C}_M$ forms a (polyhedral) cone, and any direction d within this cone is considered feasible. As long as the intersection set $\mathcal{C}_M$ is not reduced to the trivial space $\{\mathbf{0}\}$, there always exists $d \in \mathcal{C}_M$ serving as a feasible update direction.

In practice, instead of simply selecting an arbitrary feasible direction, we formulate the following convex optimization problem to determine the optimal direction d^*. Specifically, we seek the direction closest to the gradient of the lowest-priority subtask, g_M, with all higher-priority subtasks within their prescribed thresholds:

$$\begin{aligned} \min_{d} \quad & \|d - g_M\|^2 \\ \text{s.t.} \quad & d \in \mathcal{C}_M \end{aligned} \tag{6}$$

An illustrative example of the optimal solution is shown in Figure 2.

Existing Problems and Enhancements

Optimization Efficiency One common approach to directly solve problem 6 is to employ standard convex optimization solvers, such as those provided in CVXPY. However, repeatedly invoking these solvers during update steps can incur significant computational burden.

To address this issue, we propose replacing general-purpose solvers with **Dykstra's projection algorithm** (Boyle and Dykstra 1986), an efficient iterative method that, to our knowledge, has not been previously applied in LMORL. Dykstra's projection extends the classical Sequential Orthogonal Projection (SOP) method. SOP is typically employed to find a feasible $d \in \mathcal{C}$ by iteratively removing components that violate the constraints, as follows:

$$d_{i+1} = \begin{cases} d_i - \dfrac{g_i^T d_i}{\|g_i\|^2} g_i & \text{if} \quad g_i^T d_i \leq -\epsilon_i \\ d_i & \text{else} \end{cases} \tag{7}$$

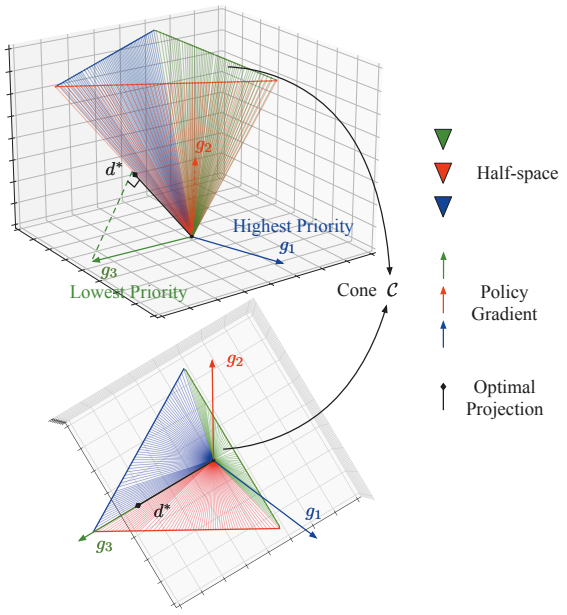


Figure 2: Illustration of the optimal solution of Equation 6 with three gradients. The blue, red and green vectors represent the policy gradients g_1, g_2, g_3, ordered from high to low priority, serving as the normal vectors of the respective half-spaces. The intersection of these half-spaces forms a cone. The optimal solution, shown as the black arrow d^*, is the vector within the intersection $\mathcal{C}$ that is closest to g_3.

In contrast to SOP, Dykstra's projection algorithm iteratively projects the gradient g_M onto the intersection set, introducing auxiliary variables r_i to track the residuals from previous projections. This mechanism ensures the final convergence to the optimal solution of Proposition 6. An iterative update proceeds as follows:

$$\begin{aligned} d_{i+1}^{(t)} &= d_i^{(t)} - \frac{g_i^T (d_i^{(t)} + r_i^{(t)})}{\|g_i\|^2} g_i \\ r_i^{(t)} &= d_i^{(t)} + r_i^{(t)} - d_{i+1}^{(t)} \end{aligned} \tag{8}$$

where t is the iteration number. It has been proven that the sequence $\{d_i^{(t)}\}$ generated by Dykstra's projection converges to the solution of Equation 6 as long as the intersection set is non-empty (Gaffke and Mathar 1989). As noted previously, $\mathbf{0}$ constitutes a trivial solution in our context, which means the convergence of Dykstra's projection is always guaranteed for our optimization.

Gradient Vanishing Another challenge during training is gradient vanishing, which arises under two different conditions. First, as illustrated in Figure 2, when the optimal direction d^* is on the boundary of $\mathcal{C}$, there exists at least one higher-priority policy gradient g_i such that $g_i^T d^* = 0$, causing some subtasks trapped in the local optima. Second, when $\mathcal{C}$ contains only the trivial solution $\mathbf{0}$, d^* makes no contribution to any subtask, and training stagnates.

To address gradient vanishing, we introduce **SE** as a rollout scheduler during training. Specifically, we extract a sub-

problem containing the top-N subtasks:

$$\begin{aligned} \min_{d} \quad & \|d - g_N\|^2 \\ \text{s.t.} \quad & d \in \mathcal{C}_N, \quad 1 \leq N \leq M \end{aligned} \tag{9}$$

where $N \sim \text{Uniform}(\{1, \cdots, M\})$ is sampled uniformly. For any N, if $\|d^*\| = 0$, we recursively set $N \leftarrow N - 1$ and solve the higher-level subproblem, until a non-zero feasible solution is found. This approach ensures each subtask is selected and trained equally often, thereby reducing the risk of local optima caused by gradient vanishing. Besides, a major advantage over previous methods is that SE does not require accurate prior knowledge or additional hyperparameters.

Finally, the general LPPG framework for policy gradient RL algorithms is summarized in Algorithm 1. Specific instantiations of LPPG-PPO and LPPG-SAC, are provided in Appendix A.

Algorithm 1: LPPG

Require: : policy network $\pi(\cdot|s)$, prioritized subtasks set $\mathcal{K}$

Compute gradient g_i for each subtask
Sample subtask index $N \sim \text{Uniform}(\{1, \ldots, |\mathcal{K}|\})$
for $n = N, \cdots, 1$ **do**
  $\mathcal{C}_n \leftarrow \bigcap_{i=1}^{n} \{\, d \,|\, g_i^\top d \geq -\epsilon_i \,\}$
  if $\mathcal{C}_n = \{\mathbf{0}\}$ **then**
    continue
  else
    break
  end if
end for
Solve $d \in \mathcal{C}_n$ closest to g_n,
return Optimal gradient d^*

Theoretical Analysis

In this subsection, we provide a theoretical analysis of the convergence properties of LPPG-RL and establish an exact lower bound for each subtask during a policy update.

Theorem 1. *Consider an LMORL problem with subtask set $\mathcal{K} = \{K_1, \cdots, K_M\}$ and LPPG-RL algorithm in a general actor-critic framework. Let θ_t be the actor parameters and ϕ_t be the multi–head critic parameters. Assume,*

1. *Two–timescale stepsizes. The actor and critic learning rates $\alpha_{\theta,t}, \alpha_{\phi,t} > 0$ satisfy $\sum_t \alpha_{\theta,t} = \sum_t \alpha_{\phi,t} = \infty$, $\sum_t (\alpha_{\theta,t}^2 + \alpha_{\phi,t}^2) < \infty$, $\lim_{t\to\infty} \alpha_{\theta,t}/\alpha_{\phi,t} = 0$*
2. *L–smooth objectives. Each sub-objective $J_i(\theta)$ is twice differentiable w.r.t θ, with Hessian L_i-Lipschitz continuous.*

Then, we have our parameters converge to a local or global lexicographic optimum $(\theta^*, \phi^*(\theta^*))$ with a stepsize $\alpha_\theta \leq \min\left\{2g_i^T d_i/(L_i\|d\|^2), i \in \{1, \cdots, M\}\right\}$

The proof of Theorem 1 is provided in Appendix E.

Theorem 2. *Consider an LMORL problem, with the policy update direction d by LPPG, and the stepsize α_θ. Then, for each policy update from π_θ to $\pi_{\theta'}$, we have the lower bound for the improvement of each performance J_i,*

$$J_i(\pi') - J_i(\pi) \geq \frac{\alpha_\theta \delta_i}{1-\gamma} - \frac{2\gamma C_i^{\pi',\pi}}{(1-\gamma)^2}\sqrt{\frac{\eta}{2}}$$

The proof of Theorem 2 is provided in Appendix F.

Experiments

In this section, we present our experimental setup and results. We focus on two main aspects: (1) demonstrating that LPPG-RL achieves zero constraint violation with high performance in safety-critical tasks. (2) illustrating explicitly how LPPG-RL distinguishes and handles different priorities when permuting lexicographic orders of subtasks.

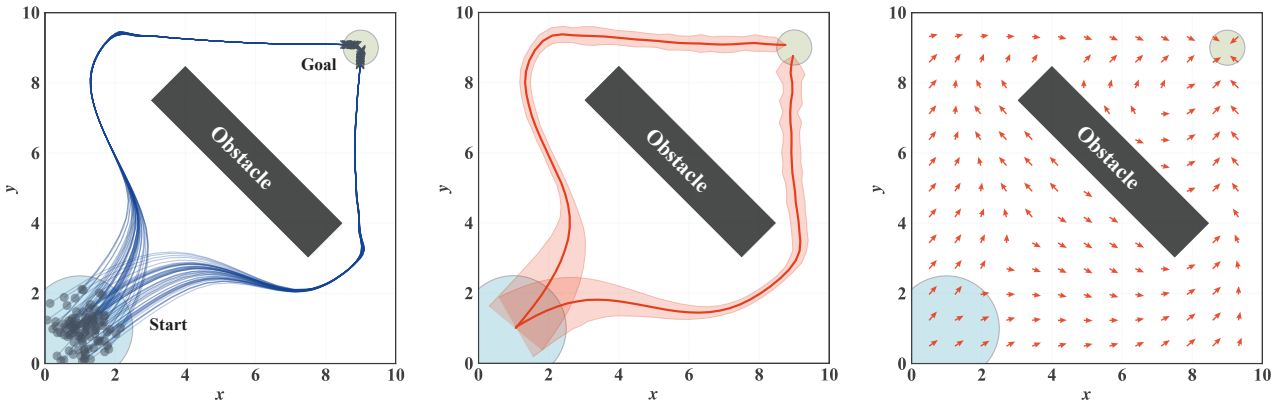
All experiments are conducted using LPPG-PPO in a 2D navigation environment with varying numbers of prioritized subtasks. We consider three scenarios: **(1) Nav2D-1G**: The agent is required to navigate to a goal located behind a rectangular obstacle. **(2) Nav2D-2G**: Two goals are placed symmetrically behind the obstacle. The agent must reach the green goal first, followed by the red goal. **(3) Nav2D-2G-rev**: The priorities of the two goals is reversed. In all scenarios, two common high-priority constraints are imposed: the agent must stay within the map boundary, and must not enter the obstacle area.

Nav2D with 1 Goal

In the following experiment, a navigation subtask with two strict constraints are given. As shown in Figure 3, a map is setup, with a start region and a goal region locating on opposite sides of an obstacle. The set $\mathcal{K}$ contains three elements for a time-limited episode. K_1: "In Boundary", K_2: "Avoid Collision", K_3: "Reach Green Goal". The initial location is sampled from a Gaussian distribution, $x \sim \mathcal{N}(1, 0.5), y \sim \mathcal{N}(1, 0.5)$. Figure 3(a) displays 50 Monte-Carlo simulation trajectories with two different seeds. Due to the symmetry, the agent explores both sides of the obstacle with equal probability. In Figure 3(b), we introduce $\mathcal{N}(0, 0.1)$ Gaussian noise to the state, and plot the 95% confidence corridor, indicating consistent safety performance under perturbations. Figure 3(c) illustrates the deterministic policy in the map, where the arrow length denotes action magnitude. The results demonstrate that our policy reliably navigates around the obstacle and towards the goal region with high speed.

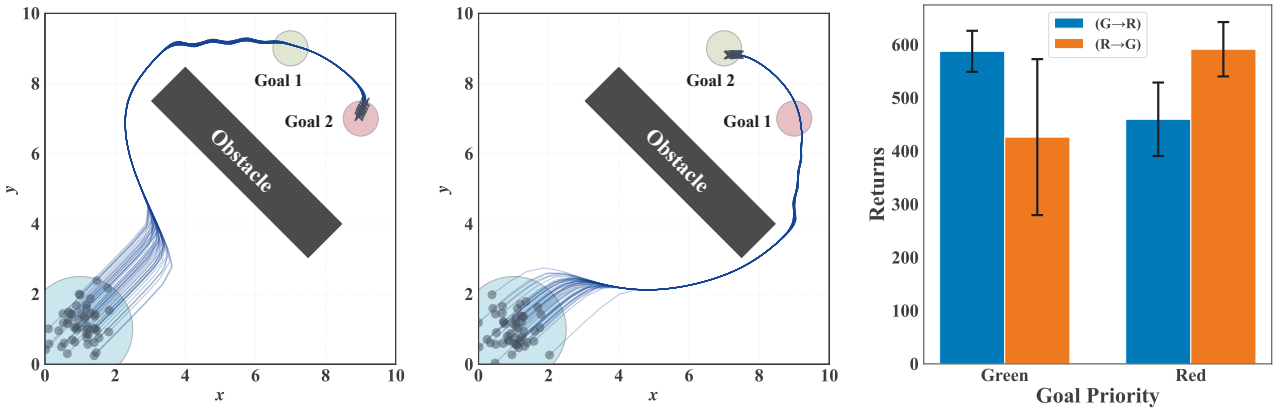
Nav2D with 2 Goals

In the following experiment, two different goal regions are set symmetrically behind the obstacle, and two comparative conditions are designed to evaluate the ability to distinguish priority orderings. The subtask set $\mathcal{K}$ contains four elements, with K_1 and K_2 same as before. In the condition 1, the remaining priorities are, K_3: "Reach Green Goal", K_4: "Reach Red Goal". In condition 2, K_3 and K_4 are reversed. Figure 4(a) shows the simulation trajectories of condition 1 (**Nav2D-2G**), where the agent tends to navigate around the obstacle from the left side to reach the green goal region more efficiently, as it is relatively closer. In contrast, Figure 4(b) presents the trajectories of condition 2 (**Nav2D-2G-**



(a) Simulation trajectories (b) 95% Confidence corridor with noise (c) $\pi^*(a \mid s)$

Figure 3: **1 Goal experiment** in the 2D navigation environment. 3(a) shows the map and agent trajectories under 50 Monte-Carlo simulations with different seeds. 3(b) displays the 95% statistical confidence corridor of trajectories under $\mathcal{N}(0, 0.1)$ Gaussian noise applied to the state. 3(c) illustrates of the deterministic policy direction for different agent locations in the map, where the length of arrows corresponds to the action magnitude.



(a) Green→Red (b) Red→Green (c) Returns comparison

Figure 4: **2 Goal experiment** in the 2D navigation environment. The start region is initialized as before and two symmetric goal regions are designated as different priority subtasks. Figure 4(a) and Figure 4(b) show the trajectories under 50 Monte-Carlo simulations with two different subtask priority configurations. The green goal region has higher priority in Figure 4(a) while it has lower priority in Figure 4(b). As shown in Figure 4(c), a comparison of the returns for the two goal regions under two different priority configurations is presented, demonstrating that our method strictly preserve lexicographic priorities. The legend "G→R" and "R→G" correspond to Figure 4(a) and Figure 4(b), respectively.

Algorithm	**Nav2D-1G (ms)**	**Nav2D-10G (ms)**	**Nav2D-20G (ms)**	**Nav2D-50G (ms)**	**Nav2D-100G (ms)**
OSQP	18.30±1.61	41.46±3.56	87.37±14.66	701.61±158.83	**2907.73±801.74**
SCS	21.13±2.31	44.47±12.04	113.97±25.54	803.13±255.26	3910.14±926.18
CLARABEL	26.79±1.41	79.17±3.91	154.44±10.83	994.35±120.79	4162.98±554.81
Dykstra(Ours)	**0.93±0.34**	**5.33±2.38**	**22.15±10.46**	**685.61±146.68**	2983.20±370.53

Table 1: Ablation study of Dykstra's projection method with varying numbers of goal regions in the 2D navigation environment. All results are reported in milliseconds (ms). For **Nav2D-nG** environment, we generate **n** different goal regions as lexicographic subtasks. Each setting is trained with 100k steps, 5 seeds, starting with the lowest priority subtask K_M. We compare the average runtime of policy-gradient search between Dykstra and several commonly used CVXPY solvers. All solvers yields the same projected gradient within a 10^{-6} relative error. See Appendix C for additional details.

Algorithm	Nav2D-1G						Nav2D-2G								Nav2D-2G-rev							
	K_1		K_2		K_3		K_1		K_2		K_3		K_4		K_1		K_2		K_3		K_4	
LPA	100	✓	-19	×	714	✓	98	✓	0	✓	631	✓	246	×	97	✓	0	✓	603	✓	-39	×
LPPO	100	✓	-20	×	740	✓	100	✓	-10	×	753	✓	382	×	100	✓	-10	×	785	✓	197	×
LPPG-PPO w/o SE	100	✓	-12	×	704	✓	98	✓	0	✓	199	×	604	✓	98	✓	0	✓	258	×	612	✓
LPPG-PPO (Ours)	98	✓	0	✓	427	✓	96	✓	0	✓	587	✓	459	✓	97	✓	0	✓	591	✓	426	✓

Table 2: Comparison of baseline algorithm LPA (Tercan and Prabhu 2024) and LPPO (Skalse et al. 2022), as well as an ablation study of SE (LPPG-PPO w/o SE), on performances in three environments under 10 different seeds. The metric is the average returns after 1M steps, where higher values are better. ✓and ×, indicate whether the subtask is completed. For each environment, the lexicographic order is verified first. The light blue block marks the optimal policy that satisfies the priority order and achieves the highest returns. The completed result table with standard deviations is provided in Appendix C.

rev), where switching the priority causes the agent to prefer the path around the obstacle from the right side. Figure 4(c) compares the returns from both goal regions under two conditions. In condition 1 ("G→R"), the agent achieves higher returns from the green, whereas in condition 2 ("R→G"), the agent obtains higher returns from the red. These results demonstrate that our trained agents are able to solve the LMORL problem with priorities strictly satisfied.

It is important to note that, to ensure the comparative experiments are not influenced by environmental factors (such as state or reward function design), all things are kept identical for the green and red goal regions.

Comparison and Ablation Study

Experiments above demonstrate the effectiveness of LPPG-RL, which employs Dykstra's projection and SE scheduler. This motivates three key questions: (1) How does Dykstra improve the efficiency of policy-gradient search compared to other solvers? (2) How does the SE scheduler help avoid local optimal? (3) How does LPPG-RL perform relative to baseline methods in LMORL?

We conduct two comparative studies to answer these questions. (1) we generate **n** different goal regions as lexicographic subtasks in **Nav2D-nG** environment, and compare the average runtime of policy-gradient search between Dykstra and other solvers. (2) We adapt the same experimental settings as before and compare the final performance of four methods: LPA (Tercan and Prabhu 2024), LPPO (Skalse et al. 2022), LPPG-PPO w/o SE, and full LPPG-PPO.

The first set of results is presented in Table 1. where we compare Dykstra with several common solvers in CVXPY, including OSQP, SCS, and CLARABEL. The results show that, for problems of practical size ($\mathbf{n} \leq 50$), Dykstra achieves up to $20\times$ faster than generic solvers. The gap gradually diminishes with **n** increasing. For larger problems, CVXPY solvers begin to match Dykstra's performance when $\mathbf{n} \approx 100$. Importantly, all solvers converge to the same solution within a relative error of 10^{-6}, resulting in indistinguishable final policy performances.

The second set of results, presented in Table 2, includes a comparison with the baseline algorithms LPA and LPPO, as well as an ablation study on SE (LPPG-PPO w/o SE). For each environment, we report the returns and subtask completion status after 1M training steps.

For Nav2D-1G, we have $K_1 - K_3$. All other methods achieves higher returns for K_3, but fail to avoid the obstacle, as they primarily focus on K_3 during early training and get trapped in suboptimal policies. In contrast, our method ensures uniform sampling, and successfully completes all subtasks. Note that, our method achieves an average score of 98 on K_1, slightly below perfect due to the residual exploration noise in limited training steps. This noise introduces an irreducible variance floor without affecting the optimality of the policy. This observation also applies to subsequent results.

For Nav2D-2G and Nav2D-2G-rev, we have $K_1 - K_4$. LPA get stuck in K_3, due to the difficulty of setting appropriate thresholds without prior knowledge. LPPO still fails to complete K_2, because the performance is sensitive to Lagrange multiplier hyperparameters. LPPG-PPO w/o SE terminates with K_4 higher than K_3. This is because, in the early stage, rollouts cover only limited states, resulting in unreliable policy gradients due to biased advantage estimations, and rendering practically infeasible gradients feasible. Therefore, the agent collects rollouts primarily near the low-priority region, further reinforcing biased estimations and leaving high-priority areas unexplored. SE addresses this issue by forcing the agent to visit high-priority states, thereby reducing bias and restoring the intended lexicographic order.

Conclusion

In this paper, we introduce **LPPG-RL**, which reformulates LMORL problem as a convex optimization of gradients. By finding the intersection of higher-priority half-spaces, LPPG-RL obtains a lexicographically feasible policy gradient direction. We also propose using Dykstra's projection to accelerate gradient search for small- to medium-scale problems, and employ SE scheduler to avoid local optima and facilitate convergence. Experimental results demonstrate that LPPG-RL outperforms previous methods. Notably, our approach does not require additional hyperparameters or prior knowledge, making it more adaptive and practical. Furthermore, we provide theoretical guarantees on convergence and a lower bound for policy updates. A limitation of our work is that we only consider a fixed priority order among subtasks. In future work, it would be interesting to explore the use of transfer learning or experience replay buffers to handle dynamic or changing priorities.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (62120106003, 62173301). We thank the anonymous reviewers for their constructive suggestions that improved this paper.

References

Abels, A.; Roijers, D. M.; Lenaerts, T.; Nowé, A.; and Steckelmacher, D. 2018. Dynamic Weights in Multi-Objective Deep Reinforcement Learning. https://arxiv.org/abs/1809.07803v2.

Achiam, J.; Held, D.; Tamar, A.; and Abbeel, P. 2017. Constrained Policy Optimization. In *Proceedings of the 34th International Conference on Machine Learning*, 22–31. PMLR.

Altman, E. 2021. *Constrained Markov Decision Processes*. New York: Routledge. ISBN 978-1-315-14022-3.

Boyle, J. P.; and Dykstra, R. L. 1986. A Method for Finding Projections onto the Intersection of Convex Sets in Hilbert Spaces. In Brillinger, D.; Fienberg, S.; Gani, J.; Hartigan, J.; Krickeberg, K.; Dykstra, R.; Robertson, T.; and Wright, F. T., eds., *Advances in Order Restricted Statistical Inference*, volume 37, 28–47. New York, NY: Springer New York. ISBN 978-0-387-96419-5 978-1-4613-9940-7.

Chow, Y.; Nachum, O.; Duenez-Guzman, E.; and Ghavamzadeh, M. 2018. A Lyapunov-Based Approach to Safe Reinforcement Learning. arXiv:1805.07708.

Chow, Y.; Nachum, O.; Faust, A.; Duenez-Guzman, E.; and Ghavamzadeh, M. 2019. Lyapunov-Based Safe Policy Optimization for Continuous Control. arXiv:1901.10031.

Coelho, D.; Oliveira, M.; and Santos, V. 2024. RLfOLD: Reinforcement Learning from Online Demonstrations in Urban Autonomous Driving. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(10): 11660–11668.

Dawson, C.; Gao, S.; and Fan, C. 2023. Safe Control With Learned Certificates: A Survey of Neural Lyapunov, Barrier, and Contraction Methods for Robotics and Control. *IEEE Transactions on Robotics*, 39(3): 1749–1767.

Deng, Y.; Gao, J.; Xiao, J.; and Feroskhan, M. 2024. Ensuring Safety in Target Pursuit Control: A CBF-Safe Reinforcement Learning Approach. arXiv:2411.17552.

Diamond, S.; and Boyd, S. 2016. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83): 1–5.

Gábor, Z.; Kalmár, Z.; and Szepesvári, C. 1998. Multi-Criteria Reinforcement Learning. In *ICML*, volume 98, 197–205.

Gaffke, N.; and Mathar, R. 1989. A Cyclic Projection Algorithm via Duality. *Metrika*, 36(1): 29–54.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *International Conference on Machine Learning*, 1861–1870. PMLR.

Li, C.; and Czarnecki, K. 2019. Urban Driving with Multi-Objective Deep Reinforcement Learning. arXiv:1811.08586.

Liu, E.; Wu, Y.-C.; Huang, X.; Gao, C.; Wang, R.-J.; Xue, K.; and Qian, C. 2025. Pareto Set Learning for Multi-Objective Reinforcement Learning. arXiv:2501.06773.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-Level Control through Deep Reinforcement Learning. *Nature*, 518(7540): 529–533.

Mossalam, H.; Assael, Y. M.; Roijers, D. M.; and Whiteson, S. 2016. Multi-Objective Deep Reinforcement Learning. https://arxiv.org/abs/1610.02707v1.

Parisi, S.; Pirotta, M.; Smacchia, N.; Bascetta, L.; and Restelli, M. 2014. Policy Gradient Approaches for Multi-Objective Sequential Decision Making. In *2014 International Joint Conference on Neural Networks (IJCNN)*, 2323–2330.

Pirotta, M.; Parisi, S.; and Restelli, M. 2015. Multi-Objective Reinforcement Learning with Continuous Pareto Frontier Approximation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1).

Qiu, R.; Yang, G.; Xu, Z.; and Shao, Z. 2025. Optimizing Weights to Fit Parametric Operation Policies for Generalized Working Conditions in Linear Systems Using Deep Reinforcement Learning. *IEEE Transactions on Industrial Informatics*, 21(4): 3186–3195.

Rietz, F.; Schaffernicht, E.; Heinrich, S.; and Stork, J. A. 2024. Prioritized Soft Q-Decomposition for Lexicographic Reinforcement Learning. arXiv:2310.02360.

Roijers, D. M.; Whiteson, S.; and Oliehoek, F. A. 2014. Linear Support for Multi-Objective Coordination Graphs. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems*, AAMAS '14, 1297–1304. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-1-4503-2738-1.

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; and Abbeel, P. 2015. High-Dimensional Continuous Control Using Generalized Advantage Estimation. *arXiv preprint arXiv:1506.02438*.

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

Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; and Riedmiller, M. 2014. Deterministic policy gradient algorithms. In *International conference on machine learning*, 387–395. Pmlr.

Skalse, J.; Hammond, L.; Griffin, C.; and Abate, A. 2022. Lexicographic Multi-Objective Reinforcement Learning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, 3430–3436.

Stooke, A.; Achiam, J.; and Abbeel, P. 2020. Responsive Safety in Reinforcement Learning by PID Lagrangian Methods. arXiv:2007.03964.

Tang, C.; Abbatematteo, B.; Hu, J.; Chandra, R.; Martín-Martín, R.; and Stone, P. 2025. Deep Reinforcement Learning for Robotics: A Survey of Real-World Successes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(27): 28694–28698.

Tercan, A.; and Prabhu, V. S. 2024. Thresholded Lexicographic Ordered Multiobjective Reinforcement Learning. arXiv:2408.13493.

Tessler, C.; Mankowitz, D. J.; and Mannor, S. 2018. Reward Constrained Policy Optimization. https://arxiv.org/abs/1805.11074v3.

Van Moffaert, K.; Drugan, M. M.; and Nowé, A. 2013. Scalarized Multi-Objective Reinforcement Learning: Novel Design Techniques. In *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, 191–199.

Xu, J.; Tian, Y.; Ma, P.; Rus, D.; Sueda, S.; and Matusik, W. 2020. Prediction-Guided Multi-Objective Reinforcement Learning for Continuous Robot Control. In *Proceedings of the 37th International Conference on Machine Learning*, 10607–10616. PMLR.

Yang, T.-Y.; Rosca, J.; Narasimhan, K.; and Ramadge, P. J. 2020. Projection-Based Constrained Policy Optimization. arXiv:2010.03152.

Zhang, H.; Lin, Y.; Han, S.; and Lv, K. 2023. Lexicographic Actor-Critic Deep Reinforcement Learning for Urban Autonomous Driving. *IEEE Transactions on Vehicular Technology*, 72(4): 4308–4319.

Zuluaga, M.; Krause, A.; and Püschel, M. 2016. E-PAL: An Active Learning Approach to the Multi-Objective Optimization Problem. *Journal of Machine Learning Research*, 17(104): 1–32.