Learning Feasible Scalarizations in Constrained Markov Decision Processes Using a Stochastic Meta-Policy

Youssef Al Ozaibi
École centrale d'électronique, LyRIDS Laboratory
Université de Versaille Paris-Saclay, SyMRIC LISV Laboratory
Paris, France
yalozaibi@ece.fr

Maxime Toquebiau
Vrije Universiteit Brussel, Artificial Intelligence Laboratory
Brussels, Belgium
mtoquebiau@ai.vub.ac.be

Manolo Dulva Hina
École centrale d'électronique, LyRIDS Laboratory
Paris, France
manolo-dulva.hina@ece.fr

Amar Ramdane-Cherif
Université de Versaille Paris-Saclay, SyMRIC LISV Laboratory
Vélizy-Villacoublay, France

ABSTRACT

In many safety-critical decision-making problems, it can be difficult to manually tune the reward function to obtain an optimal policy that behaves satisfactorily. Multi-objective reinforcement learning (MORL) offers a way to address this issue by considering policies with different trade-offs on the Pareto Front, allowing multiple policies to be represented and consequently selected after learning. However, the discovered policies are not guaranteed to satisfy constraints. On the other hand, safe RL methods based on Constrained Markov Decision Processes (CMDPs) explicitly incorporate safety requirements into the optimization and can automatically adjust constraint coefficients to ensure safety compliance. Prior work has combined multi-objective reinforcement learning with CMDP formulations to obtain sets of feasible policies under fixed scalarizations. However, constraint guarantees are tied to stationary policies and do not generally extend to dynamic switching between trade-offs during execution. To this end, we propose a hierarchical maximum-entropy framework in which a meta-policy outputs state-dependent, scalarization weights for a low-level controller. Entropy regularization encourages wider distributions over feasible trade-offs, while penalties from CMDP techniques restrict the distributions to scalarizations that satisfy constraints. In this preliminary work, we show that our method maintains high entropy over scalarizations in non-critical states while collapsing distributions over constraint-relevant reward dimensions in critical regions, enabling the discovery of diverse yet feasible behaviors within a CMDP.

KEYWORDS

Reinforcement Learning, Multi-Objective Reinforcement Learning, Constrained Markov Decision Processes

1 INTRODUCTION

In reinforcement learning, an agent seeks to maximize cumulative reward, and its behavior at optimality is therefore entirely determined by the reward function. Reward design thus becomes a difficult problem in complex tasks involving multiple, potentially conflicting objectives and constraints. In such cases, the reward function must carefully represent and balance trade-offs between competing components and reliably produce behaviors that respect the desired constraints. For example, in autonomous driving, constructing a reward that simultaneously captures safe driving, progress, traffic compliance, fuel efficiency, among various other driving criteria remains a challenging problem [1, 16]. Small changes in scalarization coefficients can lead to different behaviors at optimality, making manual tuning brittle and highly task-dependent. Moreover, attempts to induce desired behavior through reward shaping or increasingly complex reward functions can lead agents to exploit loopholes in the designed objectives, producing undesirable outcomes [3, 17].

An alternative to manual scalarization is multi-objective reinforcement learning (MORL) [8, 13, 19]. In MORL, rewards are modeled as vectors and policies are evaluated with respect to multiple objectives simultaneously. Different scalarization coefficients induce different optimal policies, each reflecting a distinct trade-off between reward components. The goal is typically to approximate the Pareto front, representing the set of undominated policies corresponding to these trade-offs [18]. After training, policies can be chosen based on the user's preferences.

While MORL exposes trade-offs between objectives, it does not by itself guarantee constraint satisfaction. One might attempt to enforce safety by selecting a scalarization that heavily weights a safety or penalty component, i.e., choosing policies on the Pareto front with high value along the safety dimension. However, solving scalarized MDPs with such scalarizations does not strictly guarantee that the constraints will be satisfied [23]. This motivates Safe RL formulations based on Constrained Markov Decision Processes (CMDPs), which treat safety-related reward dimensions as constraints [2, 11]. CMDP methods aim to optimize the primary task objectives subject to constraints on the cost dimensions. A popular solution approach is to use Lagrangian primal–dual optimization, which reformulates the constrained problem as a min–max interaction: the policy seeks to maximize the task-related objective, while Lagrange multipliers increase penalties on safety-related cost dimensions whenever constraints are violated [6, 24].

Proc. of the Adaptive and Learning Agents Workshop (ALA 2026), Aydeniz, Delgrange, Mohammedalamen, Yang (eds.), May 25 – 26, 2026, Paphos, Cyprus, https://alaworkshop2026.github.io/. 2026.

Prior work has combined MORL and CMDPs to obtain sets of feasible policies under fixed scalarizations [10, 14]. However, feasibility guarantees in CMDPs apply to stationary policies evaluated under their induced state visitation distributions. If one dynamically switches between scalarizations during execution—for example, prioritizing progress in open space but safety near obstacles—the resulting policy may reach states that were rarely or never visited under a given scalarization. In such states, constraint satisfaction is not guaranteed, since feasibility was established only under the original visitation distribution. Conversely, enforcing feasibility for each scalarization over the entire trajectory may discard trade-offs that are unsafe globally but safe in specific regions of the state space.

In this work, we propose a hierarchical framework in which a stochastic meta-policy outputs state-dependent scalarization weights for a low-level controller. Instead of learning separate feasible policies for fixed trade-offs, we learn a state-dependent distribution over scalarizations that is directly influenced by constraint penalties. This allows preferences to change during execution while maintaining safety. Similar to the Option Keyboard [4, 5], scalarizations are treated as abstract actions. Using a maximum-entropy formulation [26], the meta-policy is encouraged to maintain a wide distribution over reward trade-offs, while constraint penalties restrict this distribution to safety-respecting scalarizations. As a result, the scalarization distribution remains broad in non-critical states and narrows to those that respect constraints when safety becomes critical.

We evaluate the proposed approach in a continuous-control navigation environment using the Gymnasium Robotics framework [7, 25]. The reward is decomposed into goal-reaching as the primary task objective, obstacle collisions as a constrained cost dimension, and progression and energy consumption as secondary objectives. Our method is evaluated using both quantitative metrics and qualitative visualizations.

2 BACKGROUND

2.1 Multi-Objective Markov Decision Processes

A multi-objective Markov decision process (MOMDP) extends the standard MDP by replacing the scalar reward with a vector-valued reward function. Formally, an MOMDP is defined by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, \mathbf{r}, \mu, \gamma)$, where $\mathcal{S}$ is a state space, $\mathcal{A}$ an action space, $P(\cdot \mid s, a)$ the transition dynamics, μ an initial state distribution, $\gamma \in [0, 1)$ a discount factor, and $\mathbf{r} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^d$ a vector-valued reward function with dimension d whose components represent distinct objectives.

For a policy π, the vector return is defined as

$$\mathbf{V}^\pi = \mathbb{E}_\pi\left[\sum_{t\geq 0} \gamma^t \mathbf{r}(S_t, A_t, S_{t+1})\right],$$

which captures the discounted accumulation of each objective.

Since vector returns are only partially ordered, policies are typically compared through linear scalarizations. Given preference weights $\mathbf{w} \in \mathbb{R}^d$, a scalar objective is defined as

$$J^\pi(\mathbf{w}) = \mathbf{w}^\top \mathbf{V}^\pi,$$

and an optimal policy for a given $\mathbf{w}$ satisfies

$$\pi^\star_{\mathbf{w}} \in \arg\max_\pi \ \mathbf{w}^\top \mathbf{V}^\pi.$$

Different weight vectors generally induce different optimal behaviors. Multi-objective reinforcement learning (MORL) [8, 19] techniques aim to approximate the set of undominated policies, commonly referred to as the Pareto front. A policy π Pareto-dominates π' if

$$\mathbf{V}^\pi \succeq \mathbf{V}^{\pi'} \quad \text{and} \quad \mathbf{V}^\pi \neq \mathbf{V}^{\pi'},$$

where $\succeq$ denotes component-wise inequality. The Pareto front consists of policies for which no other policy achieves strictly better performance in at least one objective without degrading another.

2.2 Constrained Markov Decision Processes

A constrained Markov decision process (CMDP) augments the primary reward with cost signals and imposes constraints on their expected discounted returns [2].

$$J_r(\pi) = \mathbb{E}_\pi\left[\sum_{t\geq 0} \gamma^t r_t\right], \qquad J_{c_i}(\pi) = \mathbb{E}_\pi\left[\sum_{t\geq 0} \gamma^t c_{i,t}\right],$$

where r_t denotes the task reward and $c_{i,t}$ the i-th cost signal at time t. The CMDP optimization problem is

$$\max_\pi \ J_r(\pi) \quad \text{s.t.} \quad J_{c_i}(\pi) \leq b_i, \quad i = 1, \ldots, m,$$

where b_i are user-specified thresholds. A common approach to solving this constrained problem is Lagrangian primal–dual optimization [6, 11]. Introducing non-negative multipliers $\boldsymbol{\lambda} \in \mathbb{R}^m_+$, the Lagrangian is defined as

$$\mathcal{L}(\pi, \boldsymbol{\lambda}) = J_r(\pi) - \sum_{i=1}^m \lambda_i \big(J_{c_i}(\pi) - b_i\big).$$

The constrained problem is then reformulated as the saddle-point problem

$$\max_\pi \min_{\boldsymbol{\lambda}\geq \mathbf{0}} \ \mathcal{L}(\pi, \boldsymbol{\lambda}),$$

where the policy seeks to maximize task return, while the multipliers increase penalties when constraints are violated.

2.3 Maximum Entropy Reinforcement Learning

Maximum entropy reinforcement learning [12, 26] augments the standard return-maximization objective with an entropy term encouraging stochastic policies while optimizing expected performance:

$$\max_\pi \mathbb{E}\left[\sum_{t\geq 0} \gamma^t \big(r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot \mid s_t))\big)\right], \mathcal{H} = -\log\pi(\cdot \mid s_t),$$

where $\alpha > 0$ controls the trade-off between reward maximization and entropy. In this work, the goal of entropy regularization is to maintain a distribution over possible reward scalarizations under constraints, rather than using it solely for exploration.

3 ALGORITHM

We consider a Markov decision process with vector-valued rewards $\mathbf{r}(s, a, s') \in \mathbb{R}^d$, where each component r_i corresponds to a distinct objective. The reward vector includes both task-related objectives and constraint-related costs. We denote by d_r the number of task dimensions and by d_c the number of cost dimensions, with $d_r + d_c = d$.

As previously discussed, feasibility and desirable trade-offs may depend on the current state. To this end, we adopt a hierarchical setup. Here, the high-level meta-policy determines, based on the current state, which trade-off between objectives should govern behavior until the next high-level decision. Its role is to select scalarization vectors that induce feasible behaviors under the constraints while promoting task performance. Consequently, the low-level policy executes these decisions. It receives the scalarization specified by the high level and acts to realize the corresponding behavior in the environment (Figure 1). Note that the low level is completely blind with regards to the task objective or the constraints; it is trained only to optimize the objective defined by the current scalarization. This separation of roles assigns the high-level meta-policy with learning *what* behaviors best satisfy the task and the constraints given the current state, and the low-level with learning *how* to act to execute the desired behaviors.

3.1 The High-Level Meta-Policy

Meta-Policy Critic. The meta-policy is implemented using an actor–critic off-policy formulation. We learn a vector-valued action-value function $\psi_{\phi_{hi}}(s, \mathbf{z}) \in \mathbb{R}^d$, parameterized by ϕ_{hi}, where the scalarization vector $\mathbf{z}$ represents the high-level action given by the actor ω_θ. The critic is implemented as a single neural network whose output dimension matches the reward dimension d. Each output component estimates the expected discounted return of the corresponding reward component when a fixed scalarization $\mathbf{z}$ is selected. The critic treats all dimensions identically; the distinction between indices related to task or cost is used only when forming the actor objective.

When the high level selects $\mathbf{z}$, the low-level policy executes for K steps, after which a new scalarization is sampled from the high-level actor. The critic therefore accounts for the accumulated reward over the K-step interval and the discounted value at the next high-level decision state. This corresponds to a semi-Markov decision process (SMDP) backup over decision intervals [22]. High-level transitions are collected as $(s_t, \mathbf{z}_t, \mathbf{r}_t^{(K)}, s_{t+K}, \delta_t)$, where $\mathbf{r}_t^{(K)} \in \mathbb{R}^d$ denotes the undiscounted accumulated reward vector over the K-step interval and δ_t indicates termination within that interval. The vector Bellman target is defined as

$$\mathbf{y}_t = \mathbf{r}_t^{(K)} + (1 - \delta_t)\gamma^K \, \mathbb{E}_{\mathbf{z}' \sim \omega_\theta(\cdot|s_{t+K})} \left[\psi_{\phi_{hi}^-}(s_{t+K}, \mathbf{z}') \right], \tag{1}$$

where $\psi_{\phi_{hi}^-}$ denotes a slowly updated target critic. The critic parameters are updated using a mean-squared error loss over vector targets,

$$\mathcal{L}_{\text{critic}} = \mathbb{E}\left[\left\| \psi_{\phi_{hi}}(s_t, \mathbf{z}_t) - \mathbf{y}_t \right\|_2^2 \right]. \tag{2}$$

To mitigate overestimation bias, we employ standard double-critic and target-network mechanisms commonly used in off-policy actor–critic methods [9, 12].

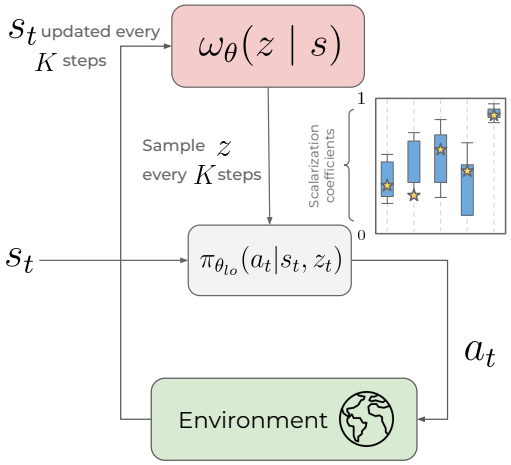

Figure 1: Hierarchical architecture with stochastic reward scalarization. The meta-policy $\omega_\theta(z \mid s)$ samples a preference vector (shown by stars) from its distributions z (shown by blue bars) every K environment steps, which conditions the low-level policy $\pi_{\theta_{lo}}(a_t \mid s_t, z_t)$ during execution. The low-level policy interacts with the environment at every step, while the meta-policy operates on a slower time scale

Meta-Policy Actor. The high-level actor $\omega_\theta(\mathbf{z} \mid s)$ is an entropy-regularized stochastic policy parameterized by θ. It outputs a continuous scalarization vector $\mathbf{z} \in [0, 1]^d$ conditioned on the current state. At each decision interval of length K, a scalarization sampled from this distribution is passed to the low-level policy (see Figure 1).

The high-level actor is trained to (i) maximize task performance over the task-related reward components, (ii) maintain diversity over scalarizations through entropy regularization, and (iii) satisfy constraints over the cost-related components using Lagrange multipliers. Using the vector critic defined above, the scalar objective is

$$J_{\boldsymbol{\lambda}}(s, \mathbf{z}) = \mathbf{w}_{\text{task}}^\top \psi_r(s, \mathbf{z}) - \boldsymbol{\lambda}^\top \big(\psi_c(s, \mathbf{z}) - \mathbf{b} \big), \tag{3}$$

where $\mathbf{w}_{\text{task}}$ weights the task-related components of the critic output, $\boldsymbol{\lambda}$ are non-negative multipliers applied to the cost-related components, and $\mathbf{b}$ denotes constraint thresholds. The optimization problem is therefore :

$$\min_{\boldsymbol{\lambda} \geq 0} \max_{\theta} \; \mathbb{E}_{s,\, \mathbf{z} \sim \omega_\theta} \left[J_{\boldsymbol{\lambda}}(s, \mathbf{z}) \right] + \alpha \, \mathbb{E}_s \left[\mathcal{H}(\omega_\theta(\cdot \mid s)) \right], \tag{4}$$

Under entropy-regularized optimization, the optimal policy corresponds to a Boltzmann distribution over the scalar objective,

$$\omega_\theta(\mathbf{z} \mid s) \propto \exp\left(\frac{1}{\alpha} J_{\boldsymbol{\lambda}}(s, \mathbf{z}) \right), \tag{5}$$

where α controls the strength of entropy regularization. This form highlights how the shape of $J_{\boldsymbol{\lambda}}(s, \mathbf{z})$ determines the spread of the distribution. If the objective varies smoothly across scalarizations in a given state, differences in value remain small, and the exponential weighting yields a broad distribution. Entropy regularization therefore promotes diversity whenever multiple scalarizations achieve similar value. The Lagrange multipliers weight the cost-related components of the critic output. When the critic predicts that certain

scalarizations incur substantially larger costs, the weighted penalty term creates larger differences in $J_{\boldsymbol{\lambda}}(s, \mathbf{z})$. Through the exponential weighting, these differences are amplified, and thus entropy is restricted in safety-critical states.

Lagrangian Multipliers Update. To solve the min–max problem in Eq. 4, the Lagrange multipliers should be updated to increase penalties when constraints are violated. Rather than performing simple gradient ascent on the constraint violation [24], we adopt the responsive PID update proposed by Stooke et al. [21], which improves stability and responsiveness of the multipliers.

3.2 Low-Level Policy

The low-level policy $\pi_{\theta_{\text{lo}}}(a \mid s, \mathbf{z})$ is conditioned on the scalarization $\mathbf{z}$ sampled from the meta-policy, $\mathbf{z} \sim \omega_\theta(\cdot \mid s)$. It is trained using an off-policy continuous-control algorithm (TD3 [9] in our implementation).

Low-Level Critic. The low-level critic is a vector-valued action-value function

$$\psi_{\phi_{\text{lo}}}(s, \mathbf{z}, a) \in \mathbb{R}^d, \tag{6}$$

where each output dimension corresponds to the expected discounted return of one reward component under the low-level policy conditioned on $\mathbf{z}$. For a given scalarization, a scalar value is obtained via the dot product

$$\mathbf{z}^\top \psi_{\phi_{\text{lo}}}(s, \mathbf{z}, a). \tag{7}$$

Low-Level Actor. The low-level actor produces a deterministic action $a = \pi_{\theta_{\text{lo}}}(s, \mathbf{z})$ conditioned on the current state and scalarization. It is trained to maximize the scalarized value induced by $\mathbf{z}$. Concretely, the actor update optimizes

$$\max_{\theta_{\text{lo}}} \mathbb{E}\Big[\mathbf{z}^\top \psi_{\phi_{\text{lo}}}\big(s, \mathbf{z}, a\big)\Big], \quad \text{with } a = \pi_{\theta_{\text{lo}}}(s, \mathbf{z}), \tag{8}$$

which follows a deterministic actor–critic formulation, with the scalarized dot product replacing the usual single-objective value.

Off-policy Preference Relabeling. Since a transition collected under one scalarization vector remains informative for others, we exploit this to improve sample efficiency. Given a batch of transitions $(s_t, a_t, \mathbf{r}_t, s_{t+1}, \mathbf{z}_t)$ sampled from the low-level replay buffer $\mathcal{D}_{\text{LL}}$, we extend each transition by pairing it with additional scalarization vectors. For every sampled transition, we retain the original tuple $(s_t, a_t, \mathbf{r}_t, s_{t+1})$ and associate it with a set of vectors $\tilde{\mathbf{z}}$ that includes the original $\mathbf{z}_t$ as well as additional vectors sampled uniformly from scalarizations stored in $\mathcal{D}_{\text{LL}}$.

For each $\tilde{\mathbf{z}}$, scalar rewards are computed as $\tilde{\mathbf{z}}^\top \mathbf{r}_t$, and the critics and actor are updated using all resulting pairs. Training on both the original preference that induced the transition and additional off-policy scalarizations improves sample efficiency and encourages the critics and policy to accurately learn the value associated with each scalarization. The low-level controller thus acts as a universal value function approximator [20] conditioned over scalarizations.

3.3 Environment Interaction Loop

The agent operates on two time scales. Every K environment steps, the meta-policy samples a new preference vector z_t, which is held fixed while the low-level policy executes actions conditioned on z_t. The case $K = 1$ corresponds to fully reactive reweighting, while larger K (we use $K = 5$) induces temporally extended behavioral modes. Algorithm 1 summarizes the full interaction and training loop.

Algorithm 1: Hierarchical Training Loop

Input: Horizon K, replay buffers $\mathcal{D}_{\text{HL}}, \mathcal{D}_{\text{LL}}$
Initialize ω_θ, $\psi_{\phi_{\text{hi}}}$, π_ψ, $\psi_{\phi_{\text{lo}}}$
Initialize multipliers $\boldsymbol{\lambda} \geq 0$
for *each episode* **do**
  Observe initial state s
  while *episode not terminated* **do**
    Sample $\mathbf{z} \sim \omega_\theta(\cdot \mid s)$
    $s_{\text{start}} \leftarrow s$, $\mathbf{r}^{(K)} \leftarrow \mathbf{0}$
    for $k = 0$ **to** $K - 1$ **do**
      Sample $a \sim \pi_{\theta_{lo}}(\cdot \mid s, \mathbf{z})$
      Execute a, observe s', $\mathbf{r}(s, a, s')$, termination d
      Store $(s, \mathbf{z}, a, \mathbf{r}(s, a, s'), s', \delta)$ in $\mathcal{D}_{\text{LL}}$
      Update low-level actor/critics from $\mathcal{D}_{\text{LL}}$
      $\mathbf{r}^{(K)} \leftarrow \mathbf{r}^{(K)} + \mathbf{r}(s, a, s')$
      $s \leftarrow s'$
      if d **then**
        break
    Store $(s_{\text{start}}, \mathbf{z}, \mathbf{r}^{(K)}, s, \delta)$ in $\mathcal{D}_{\text{HL}}$
    Update high-level critic from $\mathcal{D}_{\text{HL}}$
    Update high-level actor using Eq. (4)
    Update multipliers $\boldsymbol{\lambda}$ using PID Lagrangian [21]

3.4 Other Implementation Details

Jointly training the high-level and low-level policies introduces non-stationarity, since changes in the low-level policy alter the transition dynamics seen by the meta-policy as the low-level policy learns. To mitigate this issue, we adopt a three-stage training procedure.

Stage 1 (Low-level warm-up). We first pre-train the low-level policy independently of the meta-policy. During this phase, scalarizations $\mathbf{z}$ are sampled uniformly from $[0, 1]^d$, allowing the low-level controller to experience diverse trade-offs across reward components. This stage trains the low-level policy to approximate different behaviors before introducing hierarchical optimization.

Stage 2 (Joint training). After warm-up, both levels are trained simultaneously. Because the low-level policy has already learned to respond to different scalarizations, subsequent updates are more gradual, reducing instability in the high-level optimization. The joint training also allows the low-level to further specialize in behaviors commonly chosen by the high-level. Note that constraints are not yet applied at this stage of training.

Stage 3 (Constraints activation). The low-level policy is frozen at the final stage, while training continues for the high-level meta-policy. During this phase, constraints are activated and the Lagrange multipliers begin to be updated, allowing the meta-policy to refine

the distribution over feasible scalarizations while employing a fixed low-level controller.

4 EXPERIMENTS

We evaluate our method in PointObstacle, a continuous-control goal-reaching environment with obstacles from Gymnasium Robotics [7]. The agent controls a 2D point mass and must reach a goal region while avoiding collisions. The reward is vector-valued,

$$\mathbf{r}(s, a, s') = \left[r_{\text{goal}},\ r_{\text{progress}},\ r_{\text{energy}},\ r_{\text{collision}}\right] \in \mathbb{R}^4,$$

decomposing sparse goal rewards, progress toward the goal, energy usage, and collision penalties. The state at time t is given by

$$s_t = \left[x_t,\ y_t,\ \dot{x}_t,\ \dot{y}_t,\ \Delta g_t,\ \text{lidar}_t,\ \Delta\text{lidar}_t\right] \in \mathbb{R}^{70},$$

where (x_t, y_t) and $(\dot{x}_t, \dot{y}_t)$ denote the agent's position and velocity (4 dimensions), Δg_t is the relative goal position (2 dimensions), $\text{lidar}_t \in \mathbb{R}^{32}$ are distance readings from 32 lidar rays, and $\Delta\text{lidar}_t \in \mathbb{R}^{32}$ are their temporal changes. Including both lidar values and their changes ensures the state representation remains Markovian.

For evaluation, we use a fixed task scalarization $\mathbf{w}_{\text{task}} = [1, 0, 0, 0]$, corresponding to goal-reaching as the sole task objective. Under this scalarization, progress and energy terms do not directly affect the task return, and collision penalties are handled via constraint multipliers. Any behavior that successfully reaches the goal is therefore task-optimal, regardless of how it trades off speed of progress, energy expenditure, or safety margins. This setup intentionally creates a setting in which multiple distinct behaviors can achieve comparable task performance. In the absence of constraints, maximum-entropy optimization would permit a wide range of scalarizations, since the task objective is agnostic to secondary objectives and safety considerations.

We pose the following two questions to evaluate whether our method achieves the desired behavior. **Q1: Does the stochastic meta-policy satisfy constraints while maintaining entropy?** We measure episodic task return, collision-related constraint violations, and the differential entropy of the high-level policy during training. Together, these metrics assess whether the learned meta-policy maintains safety while preserving stochasticity over trade-offs. **Q2: Do the learned scalarizations adapt near safety-critical states?** We analyze how the meta-policy adapts scalarizations in regions where obstacles are present. Specifically, we examine whether the scalarization distribution changes as the agent approaches obstacles and becomes more flexible once it clears them.

The experimental setup is shown in Figure. 2. Start and goal locations are sampled from both ends of a corridor. When the agent reaches a goal, a new one is sampled on the other end. Episodes last 1000 steps, during which the agent must reach as many goals as possible while avoiding obstacles. Obstacles are concentrated near the center and move slowly with Brownian motion.

This corridor configuration creates varying degrees of safety risk for the agent. Since goals are placed at opposite ends, the agent must traverse the obstacle cluster in the middle during each episode. As it approaches and navigates through this region, careful collision avoidance becomes necessary. Once the agent clears the obstacle-dense center, the environment becomes low-risk again, allowing greater flexibility in scalarization choices. This structure makes the environment well suited for evaluating Q1 and Q2: constraints must be satisfied throughout training (Q1), while the learned scalarizations should contract when passing through the obstacle cluster and relax once the agent returns to safer regions (Q2).

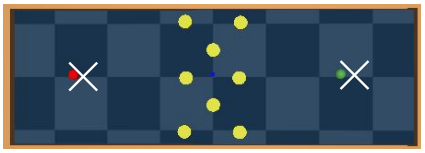

Figure 2: Corridor configuration of PointObstacle. The agent (green) navigates toward goals (red) while avoiding slowly moving obstacles (yellow). White crosses indicate possible start and goal locations.

4.1 Task Performance, Constraint Violations, and Meta-Policy Entropy (Q1)

Undiscounted episodic task returns, collision costs, and the corresponding meta-policy entropy during training are shown in Figure 3. The constraint is defined over the undiscounted cumulative collision cost per episode. Each collision incurs a unit cost $c = 1$, and we impose a threshold $b = 2$ for the Lagrangian multiplier updates. A reward of $\mathbf{r}_{\text{goal}} = 100$ is given once the agent reaches a goal.

During Stage 1 (low-level warmup) and Stage 2 (joint training), task return steadily improves, indicating that the hierarchical agent learns reliable goal-directed behavior. Over the same period, collision costs fluctuate and spike frequently above the threshold, as no constraint regularization is yet applied. The spikes are caused by the agent acting recklessly in the obstacle zone and occasionally incurring multiple collision costs.

When Stage 3 begins and constraint regularization becomes active, collision cost decreases and stabilizes below the imposed threshold. This transition is accompanied by a reduction in task return, reflecting the expected trade-off between performance and safety. Importantly, task performance does not completely collapse, indicating that constraint enforcement introduces caution without totally eliminating goal-directed behavior.

To address whether safety is achieved without collapsing behavioral diversity, we examine the differential entropy of the high-level meta-policy. To provide context for the reported entropy values, we contrast the default setting, $\alpha = 2$, with a lower coefficient, $\alpha = 0.2$. Both settings exhibit similarly high entropy at the onset of Stage 2, as the high-level is still largely untrained and produces near-uniform scalarizations. After constraint activation, the lower-α variant contracts more noticeably, whereas $\alpha = 2$ retains higher stochasticity throughout Stage 3 while still satisfying the collision constraint. This suggests that stronger entropy regularization can help limit collapse of the scalarization distribution under constraint pressure in this setting.

Overall, these results support Q1: the hierarchical meta-policy satisfies the smoothed episodic collision constraint while maintaining stochasticity over scalarization weights. Constraint satisfaction appears to arise through selective restriction of unsafe regions of the distribution rather than deterministic convergence to a single trade-off.

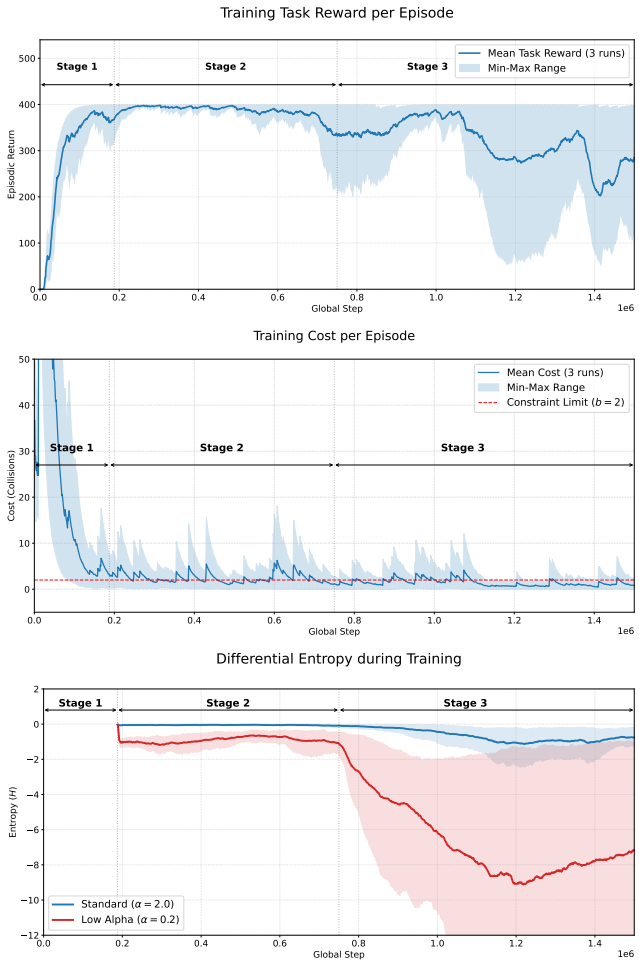

Figure 3: Top: Undiscounted episodic task return. Middle: Episodic collision cost (dashed line: threshold $b = 2$). Bottom: Meta-policy differential entropy during training. Shaded regions indicate min–max range across 3 runs.

4.2 Qualitative Analysis of Meta-Policy Under Constraints (Q2)

To address Q2, we visualize how the meta-policy adapts scalarizations in constraint-relevant states in Figure 4. The figure shows representative scalarization distributions before and after interacting with obstacle-dense regions.

As the agent approaches an obstacle, the meta-policy assigns more concentrated weights to safety-relevant objectives, reducing variability in directions that would induce constraint violations. This reflects selective pruning of high-risk trade-offs rather than uniform contraction across all objectives. Once the obstacle is cleared and the agent re-enters safer regions, variability increases and the distribution relaxes, allowing a broader set of trade-offs to be expressed. Importantly, these adjustments are localized: the meta-policy does not permanently collapse to a conservative scalarization, but instead modulates its distribution according to the surrounding state context. This behavior supports our interpretation

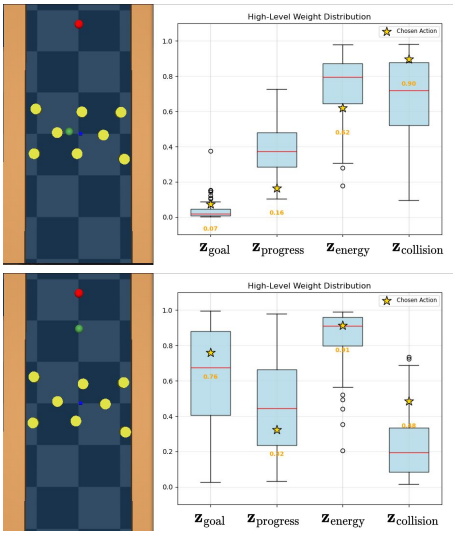

Figure 4: Top: Before passing the obstacle, the meta-policy places more importance on the scalarization corresponding to collision. Right: After clearing the obstacle, scalarization over the collision dimension decrease, indicating reduced relevance of collision avoidance.

that constraint penalties shape the structure of the scalarization distribution in a state-dependent manner, enabling near-stationary trade-offs to be maintained where feasible and selectively restricted where necessary.

5 CONCLUSION

We introduced a hierarchical maximum-entropy framework for constrained multi-objective reinforcement learning in which a stochastic meta-policy generates state-dependent scalarization weights for a low-level controller. Empirically, we showed that collision constraints can be satisfied while preserving non-trivial entropy over scalarizations, with distributions contracting in safety-critical regions and relaxing in open space.

This preliminary study highlights several open challenges. While our experiments suggest that locally adaptive adjustments can preserve useful trade-offs, we also observe that the meta-policy can behave partially as a corrective controller, highlighting the need to formally distinguish stable trade-offs from continual correction. A central open question is how to define feasibility when scalarizations are allowed to change over time. In stationary constrained MORL, scalarizations are either globally feasible or discarded, potentially excluding trade-offs that are safe in most but not all states.

Future work will formalize what it means for a trade-off to be "near-optimal" under constraints, including how much performance it retains relative to fully feasible stationary policies and how frequently the meta-policy must intervene to maintain safety. In particular, we aim to quantify whether a candidate scalarization can remain fixed across extended time periods of an episode, requiring only occasional meta-policy adjustments. We will compare the feasible Pareto front under stationary constrained MORL with the near-optimal trade-offs achievable under limited, state-dependent

switching in our framework, to assess whether controlled switching expands the set of safe trade-offs. In addition, we would like to analyze simplified tabular settings to obtain theoretical insight and validate the framework in safety benchmarks such as Safety Gymnasium [15].

More broadly, our framework shifts the focus from requiring scalarizations to be globally feasible at all times to allowing trade-offs that are safe in most states and selectively restricted in critical regions. By keeping scalarizations fixed where possible and intervening only when necessary, this approach can enable a wider range of feasible behaviors while reducing the need to carefully hand-design reward coefficients to ensure safety.

REFERENCES

[1] Ahmed Abouelazm, Jonas Michel, and J Marius Zöllner. 2024. A review of reward functions for reinforcement learning in the context of autonomous driving. In *2024 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 156–163.

[2] Eitan Altman. 2021. *Constrained Markov decision processes*. Routledge.

[3] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. 2016. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565* (2016).

[4] Andre Barreto, Diana Borsa, Shaobo Hou, Gheorghe Comanici, Eser Aygün, Philippe Hamel, Daniel Toyama, Jonathan hunt, Shibl Mourad, David Silver, and Doina Precup. 2019. The Option Keyboard: Combining Skills in Reinforcement Learning. In *Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/hash/251c5ffd6b62cc21c446c963c76cf214-Abstract.html

[5] André Barreto, Shaobo Hou, Diana Borsa, David Silver, and Doina Precup. 2020. Fast reinforcement learning with generalized policy updates. *Proceedings of the National Academy of Sciences* 117, 48 (Dec. 2020), 30079–30087. https://doi.org/10.1073/pnas.1907370117

[6] Yi Chen, Jing Dong, and Zhaoran Wang. 2021. A primal-dual approach to constrained markov decision processes. *arXiv preprint arXiv:2101.10895* (2021).

[7] Rodrigo de Lazcano, Kallinteris Andreas, Jun Jet Tai, Seungjae Ryan Lee, and Jordan Terry. 2024. *Gymnasium Robotics*. http://github.com/Farama-Foundation/Gymnasium-Robotics

[8] Florian Felten, El-Ghazali Talbi, and Grégoire Danoy. 2024. Multi-objective reinforcement learning based on decomposition: A taxonomy and framework. *Journal of Artificial Intelligence Research* 79 (2024), 679–723.

[9] Scott Fujimoto, Herke Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*. PMLR, 1587–1596.

[10] Shangding Gu, Bilgehan Sel, Yuhao Ding, Lu Wang, Qingwei Lin, Alois Knoll, and Ming Jin. 2025. Safe and balanced: A framework for constrained multi-objective reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2025).

[11] Shangding Gu, Long Yang, Yali Du, Guang Chen, Florian Walter, Jun Wang, and Alois Knoll. 2024. A review of safe reinforcement learning: Methods, theories, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46, 12 (2024), 11216–11235.

[12] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. https://doi.org/10.48550/arXiv.1801.01290 arXiv:1801.01290 [cs].

[13] Conor F Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M Zintgraf, Richard Dazeley, Fredrik Heintz, et al. 2022. A practical guide to multi-objective reinforcement learning and planning: CF Hayes et al. *Autonomous Agents and Multi-Agent Systems* 36, 1 (2022), 26.

[14] Sandy Huang, Abbas Abdolmaleki, Giulia Vezzani, Philemon Brakel, Daniel J Mankowitz, Michael Neunert, Steven Bohez, Yuval Tassa, Nicolas Heess, Martin Riedmiller, et al. 2022. A constrained multi-objective reinforcement learning framework. In *Conference on Robot Learning*. PMLR, 883–893.

[15] Jiaming Ji, Borong Zhang, Jiayi Zhou, Xuehai Pan, Weidong Huang, Ruiyang Sun, Yiran Geng, Yifan Zhong, Josef Dai, and Yaodong Yang. 2023. Safety Gymnasium: A Unified Safe Reinforcement Learning Benchmark. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. https://openreview.net/forum?id=WZmlxIuIGR

[16] W. Bradley Knox, Alessandro Allievi, Holger Banzhaf, Felix Schmitt, and Peter Stone. 2023. Reward (Mis)design for autonomous driving. *Artificial Intelligence* 316 (March 2023), 103829. https://doi.org/10.1016/j.artint.2022.103829

[17] Andrew Y Ng, Daishi Harada, and Stuart Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, Vol. 99. Citeseer, 278–287.

[18] Mathieu Reymond and Ann Nowé. 2019. Pareto-DQN: Approximating the Pareto front in complex multi-objective decision problems. In *Proceedings of the Adaptive and Learning Agents Workshop 2019 (ALA-19) at AAMAS*.

[19] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley. 2013. A Survey of Multi-Objective Sequential Decision-Making. *Journal of Artificial Intelligence Research* 48 (Oct. 2013), 67–113. https://doi.org/10.1613/jair.3987

[20] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. 2015. Universal value function approximators. In *International conference on machine learning*. PMLR, 1312–1320.

[21] Adam Stooke, Joshua Achiam, and Pieter Abbeel. 2020. Responsive safety in reinforcement learning by pid lagrangian methods. In *International conference on machine learning*. PMLR, 9133–9143.

[22] Richard S Sutton, Doina Precup, and Satinder Singh. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence* 112, 1-2 (1999), 181–211.

[23] Csaba Szepesvári. 2020. Constrained MDPs and the reward hypothesis. https://readingsml.blogspot.com/2020/03/constrained-mdps-and-reward-hypothesis.html

[24] Chen Tessler, Daniel J Mankowitz, and Shie Mannor. [n.d.]. Reward Constrained Policy Optimization. In *International Conference on Learning Representations*.

[25] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 5026–5033.

[26] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. 2008. Maximum entropy inverse reinforcement learning.. In *Aaai*, Vol. 8. Chicago, IL, USA, 1433–1438.