

# Closing the *intent-to-behavior gap* via Fulfillment Priority Logic

B. El Mabsout<sup>\*1</sup>, A. Abdelgawad<sup>\*2</sup>, and R. Mancuso<sup>1</sup>

**Abstract**—Practitioners designing reinforcement learning policies face a fundamental challenge: translating intended behavioral objectives into representative reward functions. This challenge stems from behavioral intent requiring simultaneous achievement of multiple competing objectives, typically addressed through labor-intensive linear reward composition that yields brittle results. Consider the ubiquitous robotics scenario where performance maximization directly conflicts with energy conservation. Such competitive dynamics are resistant to simple linear reward combinations. In this paper, we present the concept of objective fulfillment upon which we build Fulfillment Priority Logic (FPL). FPL allows practitioners to define logical formulae representing their intentions and priorities within multi-objective reinforcement learning. Our novel Balanced Policy Gradient algorithm leverages FPL specifications to achieve up to 500% better sample efficiency compared to Soft Actor Critic. Notably, this work constitutes the first implementation of a non-linear utility scalarization design, intended explicitly for continuous control problems.

## I. INTRODUCTION

Reward design in reinforcement learning is a nuanced and intricate process that presents the complex challenge of aligning agent behavior with intended objectives [1]. Recent work by [2], supported by findings from [3], has exposed significant limitations in current practices, revealing that 92% of surveyed RL experts rely on trial-and-error approaches, leading to overfitted and inadequate reward functions. This issue fundamentally stems from the disconnect between human reward conceptualization and RL optimization mechanisms [2], creating what we term the *intent-to-behavior gap*—the disparity between practitioners’ intended behavioral objectives and the actual behaviors exhibited by policies after optimization. We argue that traditional trial-and-error reward weight tuning in response to observed policy behaviors [4], [5] is inherently flawed and propose a systematic solution.

Researchers have approached this *intent-to-behavior gap* from various perspectives. Formal methods researchers adopted structured specifications such as temporal logic for reward signals, providing a principled framework to express temporal controller behavior precisely [6]. Others focused on reward engineering, exploiting their domain knowledge as demonstrated by [7] in their quadrupedal locomotion work. They meticulously crafted a multi-component reward function (combining linear velocity, angular velocity, base motion stability, foot clearance, collision avoidance, trajectory smoothness, and torque minimization) with carefully balanced weights to achieve robust locomotion over challenging terrain. Other methods avoid this troublesome process altogether

through inverse RL [8], while more recent work delegates the whole reward design problem to Large Language Models [9], [10].

Recognizing scalar rewards’ inherent limitations for expressing multifaceted intentions, multi-objective reinforcement learning (MORL) emerged to represent distinct objectives through vector rewards [11], [12]. While there is support for Sutton’s reward hypothesis, suggesting that maximization of any goals can theoretically reduce to maximizing a scalar signal [11], [13], [14], later contributions [15], [16] show that there are cases where this does not hold. Indeed, MORL research reveals that per-step scalarization is not expressive enough to represent desired behaviors adequately. This explains practitioners’ labor-intensive iterative reward tuning. The challenge shifts to developing methodologies that avoid multiple design-evaluate-adjust cycles. By maintaining vector representation throughout learning, MORL enables separate objective estimation before trade-offs. However, comparing policies in multi-objective spaces introduces a partial ordering problem where one policy may excel in certain objectives while underperforming in others. This requires scalarization utility functions (typically weighted sums) to establish total ordering. Nonetheless, when conflicting objectives exist, traditional linear utility functions often drive policies toward suboptimal local minima [17].

In this paper, we introduce Fulfillment Priority Logic (FPL): a logic transforming declarative policy behavior descriptions into utility functions through principled algebraic transformations, enabling practitioners to express objectives semantically while maintaining mathematical rigor and providing formal guarantees on objective fulfillment. Our approach directly addresses the “Unspecified and Multi-Objective Reward Functions” challenge in real-world RL, where traditional methods require complex, hand-crafted rewards [18]. FPL replaces weight-based specifications with logical priorities through three key innovations: (1) generalized mean operators for flexible objective composition, generalizing linear utilities, (2) Q-value level scalarization preserving intended relationships, and (3) principled normalization for stable cross-objective learning. Implementing these advances in our novel Balanced Policy Gradient (BPG) algorithm, we demonstrate up to 500% improvement in sample efficiency over Soft Actor Critic. Section II goes through the related works that tackled the *intent-to-behavior gap*. Section III provides theoretical foundations before introducing our core contributions—the Fulfillment Priority Logic (Section IV) and BPG algorithm (Section V). Empirical results follow in Section VI. Section VII concludes by offering limitations and future directions.

<sup>\*</sup>These authors contributed equally to this work.

<sup>1</sup>Department of Computer Science, Boston University, Boston, MA, USA

<sup>2</sup>Systems Engineering Division, Boston University, Boston, MA, USA

## II. RELATED WORKS

### A. Reward Design

Reward design presents a fundamental challenge in reinforcement learning systems [1]. Traditional approaches integrate reward shaping techniques to improve learning efficiency [19], yet remain constrained by the quality of underlying reward specifications. Research has progressed along multiple trajectories: inverse reinforcement learning extracts reward functions from demonstrations [8], while recent LLM-based methods like Eureka [9] leverage natural language for behavior specification, though limited by reasoning capabilities and current reward engineering methods. Notably, complex real-world applications demonstrate an emerging pattern: practitioners inherently gravitate toward structured compositional approaches, as evidenced in tokamak reactor control [20] and various multi-objective domains [21]–[23]. This aligns with our earlier geometric reward composition work [24], where we demonstrated that explicit attention to competing objectives produces more consistent policies with superior performance, even when evaluated against original environment reward functions.

### B. Multi-Objective RL

Multi-objective reinforcement learning (MORL) addresses environments requiring simultaneous optimization of multiple (possibly competing) objectives. MORL approaches divide into *a-priori* methods (preferences specified before training, yielding a single policy) and *a-posteriori* methods (generating multiple Pareto-optimal policies for post-training selection).

Research has predominantly focused on a-posteriori approaches [25], [26], with advances including evolutionary algorithms [27], hypernet-based Pareto front approximations [28], and GPI-PD’s [29] sample-efficient Convex Coverage Sets. However, these methods typically rely on linear scalarization, which becomes limiting when objectives exhibit non-linear interactions [11]. While [12] introduced an actor-critic method that leverages a non-linear utility function, their approach assumes the utility exists and is confined to discrete action spaces. In contrast, our work explicitly designs the utility within a comprehensive framework, enabling its application to continuous control tasks. Direct comparison between a-priori and a-posteriori MORL methods is inappropriate due to different objectives: a-posteriori methods generate multiple policies optimizing Pareto front coverage, as can be measured when using benchmarks such as MO-Gym [30]. Conversely, a-priori approaches like ours produce a single policy satisfying specified preferences.

**Our work** introduces logical operators for objective composition that better captures intended trade-offs non-linearly. To our knowledge, this represents the first comprehensive a-priori MORL framework specifically designed for continuous robot control with non-linear utility functions.

### C. Formal Methods

Temporal logic frameworks [31], [32] provide rigorous approaches for specifying temporal robot behavior, with extensions to learning-based control [33] and domain-specific

languages like SPECTRL [34]. A fundamental limitation in applying formal methods to reinforcement learning is the challenge of optimizing non-differentiable logical specifications using gradient-based methods. The complementary nature of formal verification and priority-based optimization (**our work**) suggests potential integration pathways: formal methods could verify safety properties while priority-based approaches handle objective trade-offs, potentially addressing both correctness guarantees and optimization efficiency in complex robotics tasks. Recent work [35] proposed weighted STL, employing smooth min/max and arithmetic/geometric means to handle competing specifications in control systems—all special cases of our power mean operators. While the work focuses on temporal properties, **FPL** extends these compositional principles to reinforcement learning with normalized objectives and priority offsets, suggesting promising integration pathways between formal methods and multi-objective RL.

Similarly, fuzzy logic approaches [36], [37] have addressed reward design through degrees of truth, creating intermediate reward landscapes that enhance learning convergence. We analyze the mathematical connections between our power mean operators and fuzzy logic further in Definition IV-C.

### D. The Sample Efficiency Challenge

The iterative refinement of rewards depends on both algorithmic sample efficiency and reward specification methodology. Improvements to either accelerate achieving desired agent behavior, closing the *intent-to-behavior gap* with particular importance for sample-constrained robotic applications. Recent works have addressed the sample efficiency issue through various algorithmic and architectural strategies. While several studies have focused on reducing estimation bias and computational overhead via ensemble methods, target network modifications, or distributional critics [38]–[40], our approach takes a different path. By integrating logical objective composition via **FPL** into Balanced Policy Gradient (**BPG**), we directly encode intended priorities, resulting in more efficient learning dynamics. This design enables us to achieve superior sample efficiency compared to state-of-the-art methods like CrossQ and TQC, without incurring additional computational cost.

## III. DEFINITIONS

### A. Multi-Objective Markov Decision Processes (MO-MDPs)

MO-MDPs extend standard MDPs with vector-valued rewards. It is defined as a tuple  $(S, A, T, \vec{R}, \gamma)$  where:

$S$ : Set	State space
$A$ : Set	Action space
$T : S \times A \rightarrow \Delta(S)$	Transition distribution
$\vec{R} : S \times A \times S \rightarrow \mathbb{R}^n$	Vector-valued reward function
$\gamma : [0,1)$	Discount factor

For MO-MDPs, given a policy  $\pi$ , the  $\vec{Q}$  vector is defined as:

$$\vec{Q}^\pi(s, a) = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t \vec{R}(s_t, a_t, s_{t+1}) \mid \begin{array}{l} s_0 = s, a_0 = a, \\ a_t \sim \pi(s_t), \\ s_{t+1} \sim T(s_t, a_t) \end{array} \right] \quad (1)$$

### B. Fulfillment

A *fulfillment* is any variable  $f \in [0, 1]$  that represents how much an objective is being fulfilled, where 0 means complete failure to fulfill and 1 is completely fulfilled. Intermediate values represent partial fulfillment.

### C. Multi-Fulfillment MDPs (MF-MDPs)

We introduce MF-MDPs, which modify MO-MDPs by expressing each objective as a *fulfillment*. We constrain the reward function such that  $\vec{R} : S \times A \times S \rightarrow [0, 1]^n$ , where each  $R_i \in \vec{R}$  expresses the fulfillment of the  $i^{th}$  objective. This formulation offers several key benefits:

**Comparable Objectives:** All objectives operate on the  $[0, 1]$  scale, making their relative fulfillment directly comparable.

**Intuitive Composition:** fulfillment values can be composed using operations that preserve their semantic meaning (e.g., AND, OR operations), producing a new fulfillment value that captures their composite meaning.

### D. Fulfillment Q-values (FQ-values)

In MF-MDPs, we can normalize Q-values to construct Fulfillment Q-values (FQ-values) due to the fact that the discounted sum in Eq. 1 is bounded from above by  $1/(1-\gamma)$ .

$$\mathbf{FQ}^\pi(s, a) = (1-\gamma)\mathbf{Q}^\pi(s, a) \in [0, 1]^n \quad \forall s \in S, a \in A \quad (2)$$

These FQ-values indicate how well a policy  $\pi$  expects to fulfill an objective over its trajectories in the MDP. Practitioners must design these FQ-values to faithfully represent their intended fulfillment—**this correspondence is fundamental**, as improper distributions will inevitably distort objective prioritization and undermine the optimization process.

## IV. FULFILLMENT PRIORITY LOGIC

In multi-objective optimization, complex relationships between objectives remain underspecified when they are composed using weighted sums because concepts such as lexicographic ordering for objective importance cannot be captured. We thus introduce Fulfillment Priority Logic (**FPL**), a domain-specific logic that uses power means to construct a larger class of these logical relationships. Power means differentially interpolate between the minimum and the maximum, allowing the choice of  $p$  to specify how much highly fulfilled values can be traded for lower fulfilled ones.

### A. Power Mean Foundations

The power mean  $\bar{\mu}_p$  unifies minimum and maximum through a continuous family of operators defined as:

$$\bar{\mu}_p(\vec{x}) = \bar{\mu}_p((x_1, \dots, x_n)) = \left( \frac{1}{n} \sum_{i=1}^n x_i^p \right)^{\frac{1}{p}}.$$

Depending on the value of the  $p$  parameter, the power mean touches many well-known operations:

$$\begin{aligned} p \rightarrow -\infty : \text{Minimum} & & p = 1 : \text{Arithmetic mean} \\ p = -1 : \text{Harmonic mean} & & p = 2 : \text{Root mean square} \\ p = 0 : \text{Geometric mean} & & p \rightarrow \infty : \text{Maximum} \end{aligned}$$

Setting  $p$  allows the specifier to transition between composition strategies smoothly, generalizing linear utilities ( $\bar{\mu}_1$ ).

For any non-negative  $\vec{x}$ , i.e. such that  $x_i \geq 0$ , the power mean has the following properties [41]:

**Range Preservation:**  $\bar{\mu}_p(\vec{x}) \in [\min(\vec{x}), \max(\vec{x})]$

**Commutativity:**  $\bar{\mu}_p(\vec{x}) = \bar{\mu}_p(\text{permutation}(\vec{x}))$

**Monotonicity:**  $y \leq z \implies \bar{\mu}_p((\vec{x}, y)) \leq \bar{\mu}_p((\vec{x}, z))$

**Monotonicity in  $p$ :**  $p_1 < p_2 \implies \bar{\mu}_{p_1}(\vec{x}) \leq \bar{\mu}_{p_2}(\vec{x})$

These properties allow for intuitive composition. First, commutativity ensures the order of objectives doesn't matter. Second, monotonicity guarantees that improving any objective only improves overall fulfillment. Crucially, monotonicity in  $p$  allows the power mean to smoothly interpolate between minimum and maximum, providing a continuous spectrum from pessimistic to optimistic composition. These properties allow deriving **FPL**'s guarantees in Definition IV-G.

### B. Logical Language

By building atop the power mean properties and formal guarantees in Definition IV-G, we introduce **FPL**. **FPL** is the first logic to formally express priority-aware objective composition specifications. When a specification is provided in **FPL**, the result is a formula  $u : \mathbf{FPL} \rightarrow [0, 1]$  that captures the desired compositional semantics as a utility function.

**(1) FPL Syntax:** We define the syntax of **FPL** formulas using the following grammar:

$$\phi ::= f \mid \phi \wedge^p \phi \mid \phi \vee^p \phi \mid \neg \phi \mid [\phi]_\delta$$

where:

- $f \in [0, 1]$  denotes a base fulfillment value;
- $p \leq 0$  in both  $\wedge^p$  and  $\vee^p$  operators;
- $\neg$  denotes logical negation;
- $[\phi]_\delta$  offsets the priority of  $\phi$  by  $\delta \in [-1, 1]$ .

**(2) Semantics:** The semantics of **FPL** define how each operator transforms fulfillment values:

$$u(f) := f \quad \text{for } f \in [0, 1] \quad (3)$$

$$u(\phi_1 \wedge^p \phi_2) := \bar{\mu}_p(u(\phi_1), u(\phi_2)) \quad \text{for } \phi_1, \phi_2 : \mathbf{FPL} \quad (4)$$

$$u(\neg \phi) := 1 - u(\phi) \quad \text{for } \phi : \mathbf{FPL} \quad (5)$$

$$u(\phi_1 \vee^p \phi_2) := u(\neg(\neg \phi_1 \wedge^p \neg \phi_2)) \quad \text{for } \phi_1, \phi_2 : \mathbf{FPL} \quad (6)$$

$$u([\phi]_\delta) := \frac{u(\phi) + \max(\delta, 0)}{1 + \delta} \quad \text{for } \phi : \mathbf{FPL} \quad (7)$$

These semantics preserve logical relationships between objectives. The conjunction  $\wedge^p$  combines objectives that must be fulfilled together, with  $p \leq 0$  controlling strictness (formalized in Definition IV-G). The disjunction  $\vee^p$ , defined via De Morgan's laws, allows focusing on the objective with the highest marginal benefit. The offset  $[\phi]_\delta$  creates priority by raising the baseline fulfillment of  $\phi$ , implementing a form of lexicographic ordering where objectives with lower offsets are prioritized. By making fulfillment composition explicit, **FPL** allows practitioners to separate concerns and reason about the trade-offs between objectives in a principled way. We expect practitioners to first define fulfillment values for their

objectives, testing whether their intention for the fulfillment of objectives maps well to  $[0, 1]$  and then use **FPL** to compose them in a way that remains consistent with their priorities.

### C. Relation to Fuzzy Logic

**FPL**, similar to fuzzy logic, generalizes Boolean operations to the continuous domain  $[0, 1]$ . Thus, at the limits  $\{0, 1\}$ , it is equivalent to Boolean logic. Choosing  $p = 0$ , our conjunction operator becomes the geometric mean ( $\forall_{x,y \in [0,1]}, x \wedge^0 y = \sqrt{x \cdot y}$ ) resembling the product t-norm [42] ( $\forall_{x,y \in [0,1]}, x \wedge y = x \cdot y$ ) commonly used in fuzzy logic. Also, when  $p \rightarrow -\infty$ ,  $\wedge^p$  becomes the minimum operator, which is equivalent to the fuzzy logic's minimum t-norm.

Contrary to fuzzy logic, which provides a notion of uncertainty, we emphasize fulfillment (Definition III-B). An illustrative example of the difference occurs when we compose a fuzzy variable with itself  $x \wedge^p x$ . In a product t-norm, this would evaluate to  $x^2$ , while in our framework, it evaluates to  $x$  (idempotence). Interpreted as fulfillment,  $x^2$  would be considered less fulfilled, contradicting our intuitive expectations. Furthermore,  $\bar{\mu}_p$  is not a t-norm, as it is not associative for every  $p$ , i.e.:

$$\exists_{x,y,z,p}, \bar{\mu}_p(\bar{\mu}_p(x, y), z) \neq \bar{\mu}_p(x, \bar{\mu}_p(y, z)).$$

### D. Relation to the Hypervolume

One of the most used multi-objective optimization metrics in the literature [27] is the hypervolume indicator. It is defined as the union of the volumes of the region dominated by the set of solutions. In the special case with one combination, the hypervolume is the product of the objectives being maximized, which is equivalent to maximizing  $\bar{\mu}_0$  (geometric mean).

### E. Toy Example

To study how **FPL** operators behave, we present a system with two fulfillment values  $f_0$  and  $f_1$ . These values compete for resources—improving one may impact the other—illustrating how our operators handle fundamental trade-offs.


**1) Multi-Objective Competitiveness:** We derive competing fulfillment values  $f_i$  from a base fulfillment value  $b_i \in [0, 1]$ . Competition is modeled as a multiplicative trade-off:


$$f_0 = b_0(1 - \alpha b_1) \quad (8)$$


$$f_1 = b_1(1 - \alpha b_0), \quad (9)$$


where  $\alpha \in [0, 1]$  controls competition strength. This creates natural tension—increasing either base value reduces the other's final fulfillment. We display the evolution of this optimization next to the operators.

**2) FPL Operator Effects:** Using these competing fulfillment values, we examine how each operator resolves trade-offs:

**CONJUNCTION** ( $f_0 \wedge^p f_1$ ) : In **FPL**, the intention to equally satisfy the two objectives can be expressed with the statement  $f_0 \wedge^p f_1$ . They converge to a compromise due to  $\wedge^p$  assigning more importance to less-fulfilled objectives.

**DISJUNCTION** ( $f_0 \vee^p f_1$ ) : When either value suffices, disjunction imposes that more fulfilled values are more important. The system rapidly maximizes  $f_0$ —initialized slightly higher—at the cost of  $f_1$ .

**PRIORITY OFFSET** ( $[f_0]_\delta \wedge^p f_1$ ) : The offset gives precedence to  $f_1$  until it reaches the threshold established by  $\delta$ , at which point  $f_0$  becomes important as well. This priority-based curriculum preserves the Markov assumption, as opposed to methods that change the reward function.

**3) Linear Utility** ( $\bar{\mu}_1(f_0, f_1)$ ) : Notice that the behavior of the linear utility is very similar to the disjunction with  $p = 1$ ; this is in stark contrast with the notion of convex combination that is usually associated with linearity. Specifically, under competitive dynamics, it no longer acts as a convex combination, but rather as a disjunction.

### F. FQ-value Composition

In RL, we apply **FPL** operators to FQ-values (Definition III-D) rather than immediate rewards. This allows reasoning about long-term trade-offs: an action might temporarily reduce one objective's fulfillment to achieve better overall fulfillment later. The  $[0, 1]$  bounds of FQ-values maintain our logical interpretation: if an objective has zero fulfillment, any conjunction involving it must also have zero fulfillment. These composed FQ-values then guide policy optimization while preserving the specified priorities between objectives.

### G. Guarantees in FPL

In the context of **FPL**, the power mean enjoys another important property we term the *minimum fulfillment bound*.

#### Theorem IV.1 (Minimum Fulfillment Bound).

$$\forall_{p \in \mathbb{R}, \vec{f} \in [0, 1]^n, \min(\vec{f}) \geq \sqrt[p]{n((\bar{\mu}_p(\vec{f}))^p - 1) + 1}.$$

*This bound guarantees that when a power mean outputs value  $y$ , every input component must have at least fulfillment  $\sqrt[p]{n(y^p - 1) + 1}$ .*

*Proof.* Let  $y = \bar{\mu}_p(\vec{f})$ . By Lemma IV.3, there exists a  $v \leq \min(\vec{f})$  such that  $\bar{\mu}_p((\vec{1}_{n-1}, v)) = y$ . Then by Lemma IV.4, we know  $v = \sqrt[p]{n(y^p - 1) + 1}$ .  $\square$

This bound is crucial for practitioners providing concrete guarantees about objective fulfillment. For example, with two objectives ( $n = 2$ ) and  $p = -2$ , achieving an output of 0.9 guarantees each individual objective has at least 0.38 fulfillment. This minimum fulfillment is particularly important in safety-critical applications where we need to ensure no objective is severely underperforming. The parameter  $p$  allows practitioners to trade off between stronger guarantees (more negative  $p$ ) and easier optimization (less negative  $p$ ), as stricter bounds require more precise balancing of objectives.

#### Theorem IV.2 (Power Mean Conservation).

$$\forall_{\vec{x}, \delta, i, j \exists \delta', \bar{\mu}_p(\vec{x}) = \bar{\mu}_p(\vec{x} + \delta \mathbb{1}_i - \delta' \mathbb{1}_j)$$

*where  $\mathbb{1}_i$  denotes a vector of zeros with a 1 in position  $i$ . This states that for any change  $\delta$  to component  $i$ , there exists a change  $\delta'$  to component  $j$  that maintains the same power mean.*

*Proof.* By commutativity of  $\bar{\mu}$ , for convenience and without loss of generality, we choose  $i = 0$  and  $j = 1$ :

$$\begin{aligned}\bar{\mu}_p(\vec{x}) &= \bar{\mu}_p(\vec{x} + \delta \mathbb{1}_0 - \delta' \mathbb{1}_1) \\ x_0^p + x_1^p + \sum_{k \geq 2} x_k^p &= (x_0 + \delta)^p + (x_1 - \delta')^p + \sum_{k \geq 2} x_k^p \\ (x_1 - \delta')^p &= x_0^p + x_1^p - (x_0 + \delta)^p \\ \delta' &= x_1 - (x_0^p + x_1^p - (x_0 + \delta)^p)^{\frac{1}{p}}\end{aligned}$$

This proves the existential by constructing  $\delta'$  explicitly.  $\square$

**Lemma IV.3** (Worst Case Configuration).

$$\forall \vec{f} \in [0, 1]^n \exists v \in \mathbb{R} : \bar{\mu}_p((\vec{1}_{n-1}, v)) = \bar{\mu}_p(\vec{f}) \text{ and } v \leq \min(\vec{f})$$

For any vector  $\vec{f}$ , there exists a vector with  $n - 1$  ones and a value  $v$  that has the same power mean but with  $v$  being bounded from above by the minimum of  $\vec{f}$ .

*Proof.* Let  $m = \text{argmin}(\vec{f})$  be the index of the minimum component. By repeatedly invoking Theorem IV.2, for each  $i \in \{1, \dots, n\} \setminus \{m\}$ , we can increase the  $i$ -th component to 1 while decreasing the  $m$ -th component to maintain the same power mean. Since we always decrease the minimum component or keep it the same, the final value  $v$  of component  $m$  must be less than  $\min(\vec{f})$ .  $\square$

**Lemma IV.4** (Explicit Minimum Solution). For a vector with  $n - 1$  ones and one value  $f$ , if  $\bar{\mu}_p((\vec{1}_{n-1}, f)) = y$  then:

$$f = \sqrt[p]{n(y^p - 1) + 1}$$

*Proof.* Let  $\vec{x}$  be such a vector. Then:

$$\begin{aligned}y &= \left( \frac{1}{n} \sum_{i=1}^n x_i^p \right)^{\frac{1}{p}} \\ y^p &= \frac{1}{n} ((n-1) \cdot 1^p + f^p) \\ ny^p &= (n-1) + f^p \\ f^p &= n(y^p - 1) + 1 \\ f &= \sqrt[p]{n(y^p - 1) + 1} \quad \square\end{aligned}$$

## V. BALANCED POLICY GRADIENT (BPG)

Balanced Policy Gradient (**BPG**) extends Deep Deterministic Policy Gradient (DDPG) to efficiently optimize policies for MF-MDPs (Definition III-C) using **FPL** specifications. *We highlight in blue our additions to DDPG.* The key innovation in **BPG** is its ability to directly accept and optimize for specifications written in **FPL**, bridging the gap between human-intuitive objective descriptions and reinforcement learning optimization. Unlike standard DDPG, which operates on scalar Q-values, BPG works with **Fulfillment Q-values (FQ-values)** (Definition III-D) that represent the degree to which each objective is fulfilled across time. These **FQ-values** are then composed using the **power mean operators as specified by the FPL formula**, preserving the logical relationships between objectives during policy updates. This approach enables the algorithm to make decisions that respect the intended priority and composition of objectives while maintaining the sample efficiency benefits of actor-critic methods.

### Algorithm 1 Balanced Policy Gradient (BPG)

---

```

Initialize networks and targets  $\pi, \pi^{\text{targ}}$  and  $\mathbf{FQ}, \mathbf{FQ}^{\text{targ}}$ 
Initialize replay buffer  $B$ 
repeat
  Receive initial state  $s_1$ 
  for each timestep  $t$  in episode do
     $\theta^\pi \leftarrow \mathcal{N}(\theta^\pi, \sigma J^{\text{pre}})$  // performance-based noise
     $a_t \leftarrow \pi(s_t)$ 
    Execute  $a_t$  and Store  $(\vec{r}_t, s_t, a_t, s_{t+1})$  in  $B$ 
  end for
  Compute and Store  $\mathbf{FV}^{\text{obs}}$  for each step in  $B$ 
  for each training iteration do
     $(s, a, \vec{r}, s_{\text{next}}, \mathbf{FV}^{\text{obs}}) \sim B$  // sample from the buffer
     $\vec{y}^{\text{TD}} \leftarrow (1 - \gamma)\vec{r} + \gamma \mathbf{FQ}^{\text{targ}}(s_{\text{next}}, \pi^{\text{targ}}(s_{\text{next}}))$ 
     $L^{\text{TD}} \leftarrow \bar{\mu}_2(\vec{y}^{\text{TD}} - \mathbf{FQ}(s, a))$ 
     $L^{\text{FV}} \leftarrow \bar{\mu}_2(\mathbf{FV}^{\text{obs}} - \mathbf{FQ}(s, a))$ 
    Update critic using  $-\nabla_{\theta^{\text{FQ}}}(L^{\text{TD}} + \alpha_{\text{FV}} L^{\text{FV}})$ 
     $J \leftarrow \bar{\mu}_2(u_{\text{FPL}}(\mathbf{FQ}(s_i, \pi(s_i))))$ 
    Update actor using policy gradient  $\nabla_{\theta^\pi} J$ 
  end for
  Update target networks
until convergence

```

---

#### A. Mitigating Overestimation Bias

Overestimation bias is a significant challenge in Q-learning-based algorithms. Existing works tackling sample efficiency such as REDQ [38] and TQC [39] address it with multi-critic approaches. However, CrossQ [40] shows that overestimation bias does not affect sample efficiency, and indeed, our experiments support the claims of CrossQ being more sample efficient. In **BPG**, however, this issue is particularly critical—inaccurate **FQ-value** estimates present **FPL** with incorrect fulfillment values, leading to incorrect prioritization between objectives. Since **FPL** makes decisions based on the relative fulfillment levels of different objectives, even minor estimation errors can significantly alter the learned behavior.

Our approach addresses this challenge through the addition of an **observed discounted returns regularization**. For each rollout of length  $n$ , we compute observed fulfillment values:

$$\mathbf{FV}^{\text{obs}} = (1 - \gamma) \sum_{t=0}^{n-1} \gamma^t \vec{r}_t + \text{TRUNCATED} \cdot (1 - \gamma) \vec{r}_n \frac{\gamma^n}{1 - \gamma}$$

where  $\mathbf{FV}^{\text{obs}}$  is the observed fulfillment values,  $\vec{r}_t$  are the normalized rewards at timestep  $t$ ,  $\gamma$  is the discount factor, and TRUNCATED indicates episode truncation. This provides a conservative estimate of fulfillment value, which we store alongside transition tuples in the replay buffer. Since  $\mathbf{FV}^{\text{obs}}$  represents returns from a previous policy with exploration noise, it serves as an effective underestimate that helps counteract overestimation without requiring additional critics.

Algorithm 1 illustrates how these components work together. After collecting experience, we compute and store **observed fulfillment values ( $\mathbf{FV}^{\text{obs}}$ )** for each state-action pair. During training, we combine a standard temporal difference loss ( $L^{\text{TD}}$ ) with a **supervised loss against these observed values ( $L^{\text{FV}}$ )**. The **power mean operators** are applied



during policy updates, where  $u_{\mathbf{FPL}}(\mathbf{FQ}(s_i, a))$  scalarizes the vector-valued FQ-value according to the priority relationships specified in **FPL**. By working directly with fulfillment values in the  $[0,1]$  range, BPG ensures that the logical semantics of **FPL** operators are preserved throughout the learning process.

## VI. EXPERIMENTS

We conducted a comprehensive empirical evaluation of **BPG** across multiple continuous control environments from the Gymnasium benchmark suite [43]. Our experimental framework assesses two primary aspects: (1) sample efficiency, measured by the number of environment interactions required to reach predefined reward thresholds, and (2) the algorithm’s robustness to overestimation bias through our normalization of value functions into **FQ**-values. We compared **BPG** against its baseline (DDPG [44]) and state-of-the-art reinforcement learning algorithms designed for sample efficiency, including SAC [45], TQC [39], and CrossQ [40], to establish its relative performance characteristics. Importantly, while **BPG** is trained using our **FPL** framework—which structures rewards into prioritized objectives—we evaluate its performance using the original scalar rewards of the benchmark environments. This choice ensures our evaluation directly compares **BPG** against baselines on standard metrics while demonstrating that our objective decomposition approach generalizes effectively to conventional performance measures.

### A. Performance on Benchmarks

1) *Sample Efficiency*: Our results on several benchmarks demonstrate significant improvements in sample efficiency compared to baselines and the state-of-the-art. The top row in Fig. 1 summarizes these findings, showing substantial reductions in the samples required to reach target performance thresholds. Notice that the benefits of **FPL** are greater in environments with more complex objectives, explained by greater loss of expressivity induced by linear composition.

Our findings reveal substantial sample efficiency improvements across environments:

**LunarLanderContinuous-v2**: BPG reaches 200 rewards in 20,000 timesteps—84% faster than DDPG (128,000) and 44% faster than the state-of-the-art CrossQ (36,000).

**Hopper-v4**: BPG requires only 27,400 timesteps to reach the 2000 reward threshold, compared to 66,600 for CrossQ (59% reduction) and 154,400 for DDPG (82% reduction).

**Pendulum-v1** and **Reacher-v4**: BPG similarly outperforms other algorithms, with improvements of 51% over CrossQ in Pendulum-v1 and outperforms all algorithms in Reacher-v4.

2) *Progress Plots*: Fig. 1 presents learning trajectories across environments, revealing two key advantages of BPG. First, BPG demonstrates significantly steeper learning curves, particularly in Pendulum-v1 and LunarLanderContinuous-v2, enabling rapid policy acquisition with minimal environment interactions. Second, BPG’s learning curves show remarkable consistency, achieving near-monotonic improvement with rapidly increasing fulfillment. This suggests that **FPL** enables more coherent credit assignment during critical early learning stages, contributing to both accelerated initial learning and optimization stability throughout the training process.

### B. Overestimation Bias

In multi-objective settings, accurate Q-value estimation is crucial for proper objective prioritization. To evaluate BPG’s resilience to overestimation bias, we conducted controlled experiments on the Hopper-v4 environment with deliberately reduced Polyak averaging.

Without our Q-value normalization mechanism  $\mathbf{FV}^{\text{obs}}$ , the average Q-value error for Fulfillment Q-values (Definition III-D) reached 0.627 after 38k steps. Adding underestimating loss with learning rate  $\alpha_{\text{FV}}=0.75$  reduced error by 77% to 0.146, while  $\alpha_{\text{FV}}=2.0$  achieved similar results (0.138). This confirms that  $L^{\text{FV}}$  mitigates overestimation bias without requiring additional critics or complex ensemble methods.

### C. Reward Engineering Comparison

We compare our approach to traditional reward engineering methods, demonstrating how **FPL** simplifies the reward specification process while maintaining or improving performance. For each environment, we show the original reward function and our **FPL** specification, highlighting how the latter more clearly expresses the intended behavioral priorities. *Note that we use  $\bigwedge^p \vec{f}$  to denote a vector of fulfillment values  $\vec{f}$  being composed with the  $\bigwedge^p$  operator.*

1) *Pendulum-v1*: On the left of the following table, we show the reward of Pendulum-v1, which is a weighted sum of angle and actuation terms with fine-tuned coefficients

Pendulum-v1 Reward Function	FPL Specification: $\phi_{\text{pendulum}}$
$-\theta^2 - 0.1 \dot{\theta}^2 - 0.001 \text{torque}^2$	$F_{\text{angle}}^2 \bigwedge^p F_{\text{actuation}}$

On the right, we show our **FPL** specification. Here  $F_{\text{angle}}$  is the fulfillment value for angle alignment, and  $F_{\text{actuation}}$  represents minimizing actuation fulfillment. The squared angle term emphasizes the primary task of angle alignment.

2) *Reacher-v4*: The reward is described by a fine-tuned weighted sum of distance and the norm of torque terms

Reacher-v4 Reward Function	FPL Specification: $\phi_{\text{reacher}}$
$-\text{distance} - 0.1   \text{torque}  ^2$	$F_{\text{distance}}^2 \bigwedge^p \left( \bigwedge^p \vec{F}_{\text{torque}} \right)$

Our **FPL** specification represents reaching the target with  $F_{\text{distance}}$ , squared for emphasis, and minimizing the torque fulfillments with  $\bigwedge^p \vec{F}_{\text{torque}}$ .

3) *Hopper-v4*: The reward is described by a fine-tuned weighted sum of velocity and the norm of action terms

Hopper-v4 Reward Function	FPL Specification: $\phi_{\text{hopper}}$
$1 + \frac{dx}{dt} - 0.001 \cdot   \text{action}  _2^2$	$\left( \bigwedge^p \vec{F}_{\text{speed}} \right) \bigwedge^p \left( \bigwedge^p \vec{F}_{\text{action}} \right)$

Here  $\bigwedge^p \vec{F}_{\text{speed}}$  represents the fulfillments for the velocity of each limb in the Hopper, and  $\bigwedge^p \vec{F}_{\text{action}}$  represents the fulfillments of the minimizing the three joint torques.

4) *LunarLanderContinuous-v2*: The original reward in LunarLander is particularly complex, defined as:

$$\begin{aligned} & \text{distance\_reward} + \text{velocity\_reward} + \text{angle\_reward} \\ & + 10 \cdot \text{legs\_contact} - 0.3 \cdot |\text{main\_engine}| \\ & - 0.03 \cdot |\text{side\_engines}| + \text{terminal\_reward} \end{aligned}$$

where the distance, velocity, and angle rewards increase as

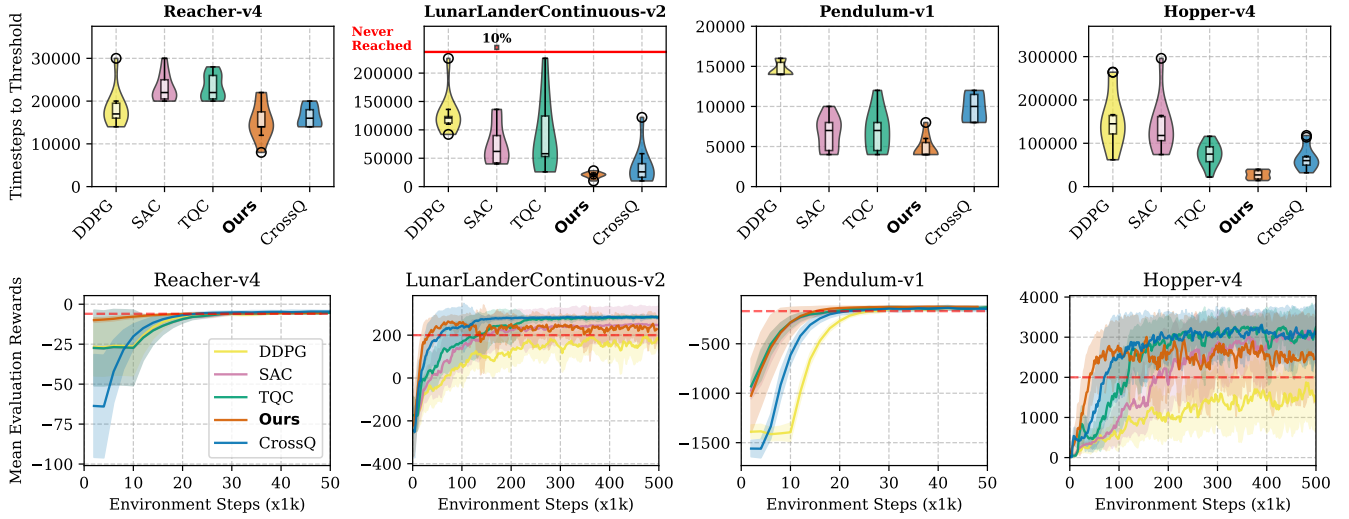


Fig. 1: The top figures show violin plots indicating the distribution of timesteps required to reach performance thresholds across 10 random seeds. The red horizontal line separates seeds failing to reach the threshold. In the bottom figures, we show a smoothed training progress of rewards versus environment steps for each algorithm. Shaded regions represent standard deviation across seeds, and the dashed lines indicate the placement of reward thresholds for each environment.

the lander gets closer to the landing pad, moves slower, and stays more horizontal. The terminal reward is +100 for safe landing or -100 for crashing.

Our **FPL** specification  $\phi_{\text{lander}}$  uses a hierarchical structure:

$$\bigwedge^p (F_{\text{near}}, [F_{\text{very\_near}}]_{0.1}, [F_{\text{legs}}]_{0.1}, [F_{\text{landed}}]_{0.1}, [F_{\text{fuel}}]_{0.5})$$

Here the offsets create a natural curriculum during training: the agent first focuses on basic proximity  $F_{\text{near}}$ , then simultaneously addresses precise positioning, leg contact, and landing ( $[F_{\text{very\_near}}]_{0.1}$ ,  $[F_{\text{legs}}]_{0.1}$ ,  $[F_{\text{landed}}]_{0.1}$ ), and finally optimizes fuel efficiency  $[F_{\text{fuel}}]_{0.5}$  once the primary landing objectives are reasonably satisfied. The conjunction ensures all objectives must ultimately be satisfied for successful landing.

#### D. Behavioral Analysis

Standard reward functions often embody fundamental limitations that **FPL** effectively addresses. In LunarLander, the non-Markovian reward aggregates multiple state-history components, complicating Q-value estimation and impeding learning efficiency. Hopper-v4 exemplifies semantic ambiguity, where identical reward values ( $\sim 1000$ ) can represent qualitatively distinct behaviors—either sustained upright posture without progression or significant forward motion lacking stability—conflating disparate policy qualities. In contrast, the learned behavior specified by **FPL** distinguishes between these behaviors, as numerically evaluated in Definition VI-E.

*A note on parameter selection in **FPL**:* **FPL** is robust to reasonable variations in power mean parameters and offsets, we choose  $p$  as either 0 or  $-1$ , which primarily serve to optimize sample efficiency rather than fundamentally changing the desired behavior. For example, not squaring the angle term in Pendulum would still result in an upright pendulum, but with slower convergence due to more conservative actions. This behavioral consistency persists across training runs, unlike linear weighted reward functions that often converge to different local optima depending on initialization.

#### E. Ablation Study: Impact of **FPL** on Behavior

Metric	With <b>FPL</b>	Without <b>FPL</b>
<b>FPL</b> : $\phi_{\text{hopper}}$	0.625	0.194
Hopper-v4 Reward	2288.80	750.35

In the table above, we describe **BPG**'s performance in Hopper-v4 on 10 seeds after 48k steps of training. Beyond performance gains with **FPL**, we observed a critical qualitative difference: without **FPL**, agents frequently achieved rewards of approximately 1000 by simply standing still—a reward hacking scenario where linear rewards were fulfilled in the original reward function but failed to achieve the intended behavior. Our **FPL** specification assigned near-zero fulfillments ( $3.8 \times 10^{-5}$ ) to such behaviors, correctly classifying them as failing to satisfy the intended objectives as the agent must move all ( $\bigwedge^p \vec{F}_{\text{speed}}$ ) the limbs forward to be considered fulfilled. Our **FPL** specification generalized effectively across multiple MuJoCo locomotion environments (HalfCheetah-v4, Walker2d-v4, Ant-v4), consistently producing forward progression in preliminary tests.

## VII. LIMITATIONS AND FUTURE WORK

This paper introduced Fulfillment Priority Logic (**FPL**), bridging the intent-to-behavior gap in multi-objective reinforcement learning through power-mean operators over normalized objectives. Our Balanced Policy Gradient algorithm achieves state-of-the-art sample efficiency while preserving intended behavioral priorities. Despite these advances, **BPG** shows lower asymptotic performance, and **FPL** exhibits sensitivity to overestimation bias that our FQ-value estimation addresses but requires an additional hyperparameter.

Future work should analyze gradient propagation through **FPL** operators, address replay buffer implementation complexities, and integrate **FPL** with more sophisticated algorithms to combine sample efficiency with improved convergence properties. Extensions to handle dynamic objectives would

further enhance **FPL**'s applicability to long-horizon problems with changing constraints.

## VIII. ACKNOWLEDGMENTS

We thank Hanna Krasowski, and Ahmad G. Ahmad for reviewing this paper, as well as Patrick Carpanedo for the help on brainstorming and the use of fulfillment as our naming.

## REFERENCES

- [1] S. Ibrahim, M. Mostafa, A. Jnadi, H. Salloum, and P. Osinenko, "Comprehensive overview of reward engineering and shaping in advancing reinforcement learning applications," *IEEE Access*, vol. 12, pp. 175473–175500, 2024.
- [2] S. Booth, W. B. Knox, J. Shah, S. Niekum, P. Stone, and A. Allievi, "The perils of trial-and-error reward design: misdesign through overfitting and invalid task specifications," in *Proc. AAAI*, vol. 37, pp. 5920–5929, 2023.
- [3] W. B. Knox, A. Allievi, H. Banzhaf, F. Schmitt, and P. Stone, "Reward (mis)design for autonomous driving," *Artif. Intell.*, vol. 316, p. 103829, 2023.
- [4] C. F. Hayes, R. Rădulescu, E. Bargiacchi, J. Kallstrom, M. Macfarlane, M. Reymond, T. Verstraeten, L. M. Zintgraf, R. Dazeley, F. Heintz, et al., "A brief guide to multi-objective reinforcement learning and planning," in *Proc. AAMAS*, pp. 1988–1990, 2023.
- [5] P. Vamplew, J. Yearwood, R. Dazeley, and A. Berry, "On the limitations of scalarisation for multi-objective reinforcement learning of pareto fronts," in *AI 2008: Advances in Artificial Intelligence* (W. Wobcke and M. Zhang, eds.), pp. 372–378, 2008.
- [6] X. Li, C.-I. Vasile, and C. Belta, "Reinforcement learning with temporal logic rewards," in *IEEE/RSJ IROS*, pp. 3834–3839, 2017.
- [7] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Sci. Robot.*, vol. 5, no. 47, p. eabc5986, 2020.
- [8] S. Adams, T. Cody, and P. A. Beling, "A survey of inverse reinforcement learning," *Artif. Intell. Rev.*, vol. 55, pp. 4307–4346, 2022.
- [9] Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar, "Eureka: Human-level reward design via coding large language models," in *Proc. ICLR*, 2024.
- [10] W. Yu, N. Gileadi, C. Fu, S. Kirmani, K.-H. Lee, M. G. Arenas, H.-T. L. Chiang, T. Erez, L. Hasenclever, J. Humprik, et al., "Language to rewards for robotic skill synthesis," *arXiv:2306.08647*, 2023.
- [11] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, "A survey of multi-objective sequential decision-making," *J. Artif. Intell. Res.*, vol. 48, pp. 67–113, 2013.
- [12] M. Reymond, C. F. Hayes, D. Steckelmacher, D. M. Roijers, and A. Nowé, "Actor-critic multi-objective reinforcement learning for non-linear utility functions," *Auton. Agents Multi-Agent Syst.*, vol. 37, no. 2, p. 23, 2023.
- [13] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
- [14] D. Silver, S. Singh, D. Precup, and R. S. Sutton, "Reward is enough," *Artif. Intell.*, vol. 299, p. 103535, 2021.
- [15] J. M. V. Skalse and A. Abate, "The reward hypothesis is false," in *NeurIPS ML Safety Workshop*, 2022.
- [16] M. Bowling, J. D. Martin, D. Abel, and W. Dabney, "Settling the reward hypothesis," in *Proc. ICML*, 2023.
- [17] K. Van Moffaert, T. Brys, A. Chandra, L. Esterle, P. R. Lewis, and A. Nowé, "A novel adaptive weight selection algorithm for multi-objective multi-agent reinforcement learning," in *IEEE IJCNN*, pp. 2306–2314, 2014.
- [18] D. J. Mankowitz, G. Dulac-Arnold, and T. Hester, "Challenges of real-world reinforcement learning," 2019.
- [19] Y. Hu, W. Wang, H. Jia, Y. Wang, Y. Chen, J. Hao, F. Wu, and C. Fan, "Learning to utilize shaping rewards: a new approach of reward shaping," in *Adv. NeurIPS*, 2020.
- [20] J. Degraeve, F. Felici, J. Buchli, M. Neunert, B. Tracey, F. Carpanese, T. Ewalds, R. Hafner, A. Abdolmaleki, D. Casas, et al., "Magnetic control of tokamak plasmas through deep reinforcement learning," *Nature*, vol. 602, pp. 414–419, 2022.
- [21] A. Jalalimanesh, H. S. Haghighi, A. Ahmadi, H. Hejazian, and M. Soltani, "Multi-objective optimization of radiotherapy: distributed q-learning and agent-based simulation," *J. Exp. Theor. Artif. Intell.*, vol. 29, no. 5, pp. 1071–1086, 2017.
- [22] F. Pianosi, A. Castelletti, and M. Restelli, "Tree-based fitted q-iteration for multi-objective markov decision processes in water resource management," *J. Hydroinform.*, vol. 15, no. 2, pp. 258–270, 2013.
- [23] T. Verstraeten, A. Nowé, J. Keller, Y. Guo, S. Sheng, and J. Helsen, "Fleetwide data-enabled reliability improvement of wind turbines," *Renew. Sustain. Energy Rev.*, vol. 109, pp. 428–437, 2019.
- [24] B. Mabsout, S. Mysore, K. Saenko, and R. Mancuso, "How to train your quadrotor: A framework for consistently smooth and responsive flight control via reinforcement learning," *ACM Trans. Cyber-Phys. Syst.*, vol. 5, no. 4, 2021.
- [25] M. Sakawa and K. Kato, "Interactive decision-making for multiobjective linear fractional programming problems with block angular structure involving fuzzy numbers," *Fuzzy Sets Syst.*, vol. 97, no. 1, pp. 19–31, 1998.
- [26] T. T. Nguyen, N. D. Nguyen, P. Vamplew, S. Nahavandi, R. Dazeley, and C. P. Lim, "A multi-objective deep reinforcement learning framework," *Eng. Appl. Artif. Intell.*, vol. 96, p. 103915, 2020.
- [27] J. Xu, Y. Tian, P. Ma, D. Rus, S. Sueda, and W. Matusik, "Prediction-guided multi-objective reinforcement learning for continuous robot control," in *Proc. ICML*, 2020.
- [28] T. Shu, K. Shang, C. Gong, Y. Nan, and H. Ishibuchi, "Learning pareto set for multi-objective continuous robot control," in *Proc. IJCAI*, 2024.
- [29] L. N. Alegre, A. L. C. Bazzan, D. M. Roijers, A. Nowé, and B. C. da Silva, "Sample-efficient multi-objective learning via generalized policy improvement prioritization," in *Proc. AAMAS*, p. 2003–2012, 2023.
- [30] F. Felten, L. N. Alegre, A. Nowé, A. L. C. Bazzan, E. G. Talbi, G. Danoy, and B. C. da Silva, "A toolkit for reliable benchmarking and research in multi-objective reinforcement learning," in *Proceedings of the 37th (NeurIPS 2023)*, 2023.
- [31] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE Trans. Robot.*, vol. 25, no. 6, pp. 1370–1381, 2009.
- [32] M. Lahijanian, S. B. Andersson, and C. Belta, "Temporal logic motion planning and control with probabilistic satisfaction guarantees," *IEEE Trans. Robot.*, vol. 28, no. 2, pp. 396–409, 2011.
- [33] D. Aksaray, A. Jones, Z. Kong, M. Schwager, and C. Belta, "Q-learning for robust satisfaction of signal temporal logic specifications," in *IEEE CDC*, pp. 6565–6570, 2016.
- [34] K. Jothimurugan, R. Alur, and O. Bastani, "A composable specification language for reinforcement learning tasks," *Adv. NeurIPS*, vol. 32, 2019.
- [35] N. Mehdipour, C.-I. Vasile, and C. Belta, "Specifying user preferences using weighted signal temporal logic," *IEEE Control Systems Letters*, vol. 5, no. 6, pp. 2006–2011, 2020.
- [36] Z. Zhai, W. Chen, X. Li, and J. Guo, "A modified average reward reinforcement learning based on fuzzy reward function," in *Proc. Int. MultiConf. Eng. Comput. Sci.*, vol. 1, pp. 113–117, 2009.
- [37] Y. Singh and N. Pal, "Reinforcement learning with fuzzified reward approach for mppt control of pv systems," *Sustain. Energy Technol. Assess.*, vol. 48, p. 101665, 2021.
- [38] X. Chen, C. Wang, Z. Zhou, and K. Ross, "Randomized ensembled double q-learning: Learning fast without a model," *arXiv:2101.05982*, 2021.
- [39] A. Kuznetsov, P. Shvechikov, A. Grishin, and D. Vetrov, "Controlling overestimation bias with truncated mixture of continuous distributional quantile critics," in *Proc. ICML*, pp. 5556–5566, 2020.
- [40] A. Bhatt, D. Palenicek, B. Belousov, M. Argus, A. Amiranashvili, T. Brox, and J. Peters, "Crossq: Batch normalization in deep reinforcement learning for greater sample efficiency and simplicity," *arXiv:1902.05605*, 2019.
- [41] P. S. Bullen, "Chapter iii - the power means," in *Handbook of Means and Their Inequalities*, pp. 175–265, Kluwer, 2003.
- [42] I. Beg and S. Ashraf, "Similarity measures for fuzzy sets," *Appl. Comput. Math.*, vol. 8, no. 2, pp. 192–202, 2009.
- [43] M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. De Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG, et al., "Gymnasium: A standard interface for reinforcement learning environments," *arXiv:2407.17032*, 2024.
- [44] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. M. O. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv:1509.02971*, 2015.
- [45] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. ICML*, pp. 1861–1870, 2018.