Multi-Objective Categorical Deep Q-Networks

Farès Chouaki
LIP6, Sorbonne Université
Paris, France
Fares.chouaki@lip6.fr

Aurélie Beynier
LIP6, Sorbonne Université
Paris, France
aurelie.beynier@lip6.fr

Nicolas Maudet
LIP6, Sorbonne Université
Paris, France
nicolas.maudet@lip6.fr

Paolo Viappiani
LAMSADE, PSL
Paris, France
paolo.viappiani@lamsade.dauphine.fr

ABSTRACT

Motivated by recent advances in distributional reinforcement learning on the one hand and Multi-Objective Reinforcement Learning (MORL) on the other, we propose MO-CDQN, a value-based algorithm that, given a possibly non-linear scalarization function, learns the policy with maximal expected scalarized return. Leveraging the Kantorovich-Rubinstein duality, we prove the theoretical validity of our method for Lipschitz-continuous scalarization functions. We establish that the state-action return distributions learned by our algorithm converge to a fixed point whose expected scalarized return is optimal. Our approach is then extended to propose the first value-based multi-policy algorithm for solving MORL problems under the expected scalarized return criterion. The proposed algorithms are tested on several environments from the MO-gymnasium benchmark. The results are promising and show that, on the one hand, our algorithm learns policies better than those obtained by existing approaches in the literature while requiring fewer interactions with the environment. On the other hand, given a set of scalarization functions, our multi-policy algorithm takes advantage of its off-policy nature to successfully optimize several policies concurrently and efficiently provide a set of policies, each optimal for a given scalarization function.

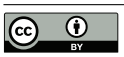
KEYWORDS

Multi-Objective Reinforcement Learning; Distributional Reinforcement Learning

ACM Reference Format:
Farès Chouaki, Aurélie Beynier, Nicolas Maudet, and Paolo Viappiani. 2026. Multi-Objective Categorical Deep Q-Networks. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 9 pages. https://doi.org/10.65109/ARIZ7102

1 INTRODUCTION

Many real-world sequential decision-making problems such as electric plant control and autonomous driving involve compromising between several possibly conflicting objectives. To solve such problems, algorithms from the Multi-Objective Reinforcement Learning (MORL) sphere can be used. On the one hand, when no information on the preferences over the objectives is available, multi-policy algorithms can be used to learn a set of policies that ensure non-dominated returns. On the other hand, when such preferences are available in the form of a scalarization function, single-policy algorithms can learn a policy maximizing the scalarized returns. A majority of single-policy algorithms assume the scalarization function to be a weighted sum. Such a simplification allows for using state-of-the-art algorithms from mono-objective reinforcement learning to solve MORL problems. However, it is often too restrictive and unrealistic (especially since it requires setting the appropriate weights, which is quite difficult in practice and limits the expressivity of the scalarization). Some single-policy algorithms address this expressivity issue through the use of more complex non-linear scalarization functions. These algorithms can be categorized according to the criterion they optimize: some use the Scalarized Expected Return (SER) criterion [18], aiming to identify a policy that maximizes a score defined as the aggregation of the obtained average return vectors, while others [8, 16, 17] choose to optimize the Expected Scalarized Return (ESR) and learn the policy that maximizes the average of the aggregated returns. Unlike SER, ESR ensures more balanced performance among the objectives over a single execution. ESR achieves this by penalizing policies that prioritize different objectives inconsistently across executions. Moreover, the distinction between ESR and SER becomes crucial when dealing with non-linear preferences such as those commonly exhibited by humans. Under such preferences, ESR is particularly well-suited for scenarios where a policy is executed only once due to the nature of the application. For example, rescue rovers or exploration robots typically operate over a single mission in a given environment and must perform effectively during that one deployment. In other cases, the cost of implementing a policy may be prohibitively high, making repeated execution impractical. Government budget allocation problems highlight a use case where repeated policy execution is not possible as funding is non-recoverable once a policy is implemented. It is also the desirable way to solve scenarios where the implementation of a policy is irreversible. Medical treatment problems illustrate this use case. Another more recent example of such scenarios is the choice of response to the current global environmental crisis, where greenhouse gas emissions have to be reduced while still ensuring that the global economy remains stable. It is clear that no matter what global



Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/ARIZ7102

policy is chosen, it will have lasting impacts on humanity. Under such circumstances, a single execution of the policy determines its value. Given the critical nature of these tasks, developing efficient algorithms to determine the optimal ESR policy is crucial.

Contributions This paper addresses two issues of the MORL literature. First, we address the lack of a value-based MORL algorithm for ESR optimization. Building on the categorical deep-Q-networks algorithm, we propose Multi-Objective Categorical Deep-Q-Networks (MO-CDQN), a value-based algorithm that finds the optimal policy for a given scalarization function. We prove that the return distribution learned by our algorithm for each state-action pair has the same expected scalarized return as the true distribution of the returns. The lack of ESR-optimizing multi-policy algorithms is then addressed. We extend MO-CDQN by leveraging off-policy experience and we provide Distributed-MO-CDQN, a first solution to multi-policy learning under ESR. Comparison of our single-policy algorithm against existing state-of-the-art algorithms proves that ours learns better policies more efficiently. Moreover, our multi-policy algorithm is even more efficient when provided with a portfolio of scalarization functions. Through the sharing of experience across learners, our algorithm is able to parallelize the policy optimization process and find the optimal policies for each scalarization function in the given portfolio.

2 BACKGROUND

2.1 Multi-Objective Markov Decision Processes

Multi-objective sequential decision-making problems can be modeled using Multi-Objective Markov Decision Processes (MO-MDPs) [12], which generalize the classical Markov Decision Process framework. An MO-MDP is defined as a tuple $\langle d, S, A, T, \mu, R, \gamma \rangle$, where d denotes the number of objectives in the decision problem, S is the state space, and A is the action space. The transition model $T : S \times A \to \mathcal{P}(S)$ maps each state-action pair (s, a) to a probability distribution over the next states, specifying the likelihood of transitioning to each state $s' \in S$. The initial state distribution given by $\mu : \mathcal{P}(S)$ represents the probability distribution over the starting states. The reward function $R : S \times A \to \mathbb{R}^d$ assigns a d-dimensional real-valued reward vector to each state-action pair, capturing the multi-objective nature of the problem. Finally, $\gamma \in [0, 1)^d$ is a vector of discount factors, one for each objective.

2.2 Optimization Criteria in Multi-Objective Reinforcement Learning

Multi-objective RL algorithms aim to either approximate the Pareto front or optimize a given aggregation of objectives. When an aggregation function is provided, single-policy methods seek the policy that maximizes the aggregated return. Under non-linear scalarization functions, two distinct optimization criteria must be distinguished:

• **Scalarized Expected Return (SER)**: The value of a policy is obtained by applying the scalarization function over the expected return vector it obtains. Thus, given a scalarization function $u : \mathbb{R}^d \to \mathbb{R}$, the value V^π of a policy π is given by $V_u^\pi = u\left(\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid \pi, s_0 \sim \mu\right]\right)$.

• **Expected Scalarized Return (ESR)**: The value of a policy is the expected value of the scalarized returns it obtains. Therefore, given a scalarization function $u : \mathbb{R}^d \to \mathbb{R}$, the value V^π of a policy π is given by $V_u^\pi = \mathbb{E}\left[u\left(\sum_{t=0}^{\infty} \gamma^t r_t\right) \mid \pi, s_0 \sim \mu\right]$.

In practice, SER-maximizing policies are suited for settings where performance is averaged over many executions, while ESR-maximizing policies are preferred when each individual execution matters or must meet specific criteria.

2.3 Categorical Reinforcement Learning

This section discusses issues related to the representation of the value function, action selection, and the update procedure in categorical RL for multi-objective problems.

State-Action Value Function Representation Given a policy π, categorical RL aims to learn, for each state-action pair (s, a), the distribution of returns $Z^\pi(s, a)$ obtained by taking action a in state s and then following π. To approximate this distribution, a categorical representation is used. It is parameterized by N, V_{MIN}, and V_{MAX}. The support of this distribution consists of N evenly spaced points $\mathcal{S} = \{z_i = V_{\text{MIN}} + i\Delta z \mid i = 0, \ldots, N-1\}$ where $\Delta z = \frac{V_{\text{MAX}} - V_{\text{MIN}}}{N-1}$, and each point z_i is assigned a probability mass p_i.

Action Selection In each state s, each action a has an associated categorical return distribution. The Q-value of each action is computed as the expected value of each return distribution, i.e., $Q(s, a) = \mathbb{E}[Z(s, a)] = \sum_{z_i \in \mathcal{S}} z_i p_i$. The action with the maximum Q-value is taken.

State-Action Value Function Update Bellemare et al. [3] extended the Bellman equation to the distributional setting as:

$$Z(s, a) \overset{D}{=} R(s, a) + \gamma Z(S', A') \tag{1}$$

This expresses the return distribution for (s, a) as the sum of the reward distribution $R(s, a)$ and the discounted return distribution at the next state-action pair $Z(S', A')$. When all three of these distributions are represented with a categorical representation with the same support, scaling $Z(S', A')$ by γ shifts its support to shrinked_$\mathcal{S} = \{\gamma z_i \mid z_i \in \mathcal{S}\}$. It must then be projected back onto the original support $\mathcal{S}$ to be combined with $R(s, a)$. Bellemare et al. [3] introduced a projection operator for this purpose, enabling learning with fixed-support categorical distributions.

3 LITERATURE REVIEW

This section introduces the existing research and algorithms on topics related to those addressed in the paper.

3.1 Distributional Reinforcement Learning

Distributional reinforcement learning can be done using either categorical or quantile-based representations. C51-DQN [3], the first distributional RL algorithm, extends DQN by modeling returns as a categorical distribution and updating them using a projected Bellman operator. However, it requires prior knowledge of the return range, limiting its flexibility. QR-DQN [7] addresses this by representing returns using fixed quantiles, which improves accuracy and performance. Still, fixing the number of quantiles is suboptimal, as shown by IQN [6], which learns the full quantile function dynamically. Recently, [29] proposed MD3QN, allowing the learning of the joint return distribution in multi-reward settings. However, their

algorithm focuses on settings where a primitive reward function can be decomposed into a sum of several components.

3.2 Multi-Objective Reinforcement Learning

Two main types of MORL algorithms can be distinguished:

Single-policy algorithms assume a known preference over objectives via a scalarization function and learn a single policy that maximizes the chosen optimization criterion.

Multi-policy algorithms assume no prior knowledge of preferences and instead learn a diverse set of potentially optimal policies, allowing later adaptation to user preferences.

When no scalarization function is provided, *inner-loop algorithms* learn multiple policies simultaneously, while *outer-loop algorithms* assume a parametrized model of the scalarization function and learn optimal policies for different parameter settings. These settings are varied systematically to generate a set of optimal policies.

Most MORL research has focused on maximizing SER [1, 2, 15, 20, 23, 28]. Recently, more attention has been given to optimizing ESR, introducing novel single-policy and multi-policy algorithms:

Single-policy algorithms for optimizing ESR: The first ESR-oriented algorithm for non-linear scalarizations, *Expected Utility Policy Gradient (EUPG)* [17], extends policy gradient methods by conditioning policies on both observations and accumulated returns, and incorporating the accumulated returns into action value estimation. To improve sample efficiency and stability, [16] proposed *Multi-Objective Categorical Actor-Critic (MOCAC)*, which uses a distributional critic to model return distributions, enabling mid-episode bootstrapping and more frequent updates. [10] introduced *Non-Linear Utility MCTS (NLU-MCTS)* and *Distributional MCTS*, which extend Monte Carlo Tree Search to optimize policies under ESR. To promote fairness across objectives, [8] proposed *NSW-Q-learning*, a Q-learning variant optimizing under the Nash Social Welfare scalarization.

Multi-policy algorithms for optimizing ESR: [11] extended value iteration to learn a set of ESR-optimal policies, while [19] built on [23] to find undominated policies under a new dominance criterion. Both are inner-loop methods restricted to discrete state and action spaces. To address this, [4] introduced a two-step outer-loop approach: it first samples N diverse non-linear scalarization functions, then solves the corresponding N scalarized MDPs using policy-gradient methods to obtain N ESR-optimal policies.

3.3 Research Gap

Despite the bulk of research in multi-objective RL being focused on SER, interest in optimizing ESR is growing. However, existing solutions still suffer from several limitations highlighted below:

(1) Policy-based algorithms like EUPG and MOCAC greedily exploit experiences, cannot learn deterministic policies or generalize to other utility functions, and are unable to leverage novel research in value-based reinforcement learning.
(2) Single-policy value-based algorithms like NSW-QL are only useful for tasks with small state and action spaces.
(3) Single-objective algorithms like [3, 14, 21] cannot treat multi-objective problems and require adaptation.
(4) Inner-loop multi-policy algorithms optimizing ESR can only be used on discrete state and action spaces.
(5) Outer-loop multi-policy algorithms optimizing ESR like [4] require retraining the agent several times with different scalarization functions to achieve a decent approximation of all possibly optimal policies.

In this paper, we first address limitations 1–3 by introducing MO-CDQN, the first value-based multi-objective distributional algorithm extending C51-DQN to optimize policies under ESR given a scalarization function. To tackle limitations 4 and 5, we then propose a multi-policy variant that leverages MO-CDQN's off-policy nature to simultaneously optimize multiple non-linear scalarization functions in large state spaces, reusing experience across functions.

4 OPTIMIZING A SINGLE SCALARIZATION FUNCTION UNDER ESR

We propose the Multi-Objective Categorical Deep-Q-Networks (MO-CDQN) algorithm, which extends the C51-DQN algorithm to the multi-objective case. We adapt both the action selection and update rules of the C51-DQN algorithm while also modifying the network architecture to predict multivariate categorical distributions. In this section, we first describe the representation of return distributions in our proposed method. We then outline the action-selection rule employed by our algorithm and finally detail how the policy network is updated to predict accurate estimates of the return distributions.

4.1 Multi-Objective Categorical Deep-Q-Networks

We first present the inner workings of the proposed solution and show how it derives the ESR-maximizing policy for a chosen scalarization function.

Return Distribution Representation Our MO-CDQN algorithm learns to predict the entire multivariate return distribution for each state-action pair in the environment. For each objective, we propose to represent the return distribution as a categorical representation. In the multi-objective context, since the return distribution is a multivariate distribution, our algorithm uses a multivariate categorical representation to store the return distribution. The support of this distribution is the set obtained from the Cartesian product of the supports of the categorical representations of the different objectives: $\mathcal{S} = \bigtimes_{j\in\{0..d\}} \mathcal{S}_j$, where $\mathcal{S}_j = \{V_{\text{MIN}_j} + \Delta z_j \cdot i \mid 0 \leqslant i \leqslant N_j - 1\}$. V_{MIN_j} is the minimal value of the return the agent can achieve on the j-th objective, and $\Delta z_j = \frac{V_{\text{MAX}_j} - V_{\text{MIN}_j}}{N_j - 1}$ is the size of the intervals of the categorical representation of the j-th objective. Note that each item of $\mathcal{S}$ is a tuple of size d; each tuple in $\mathcal{S}$ is called an *atom*, and we write $\mathcal{S} = \{\mathbf{s}_i \mid 0 \leqslant i \leqslant \prod_{j\in\{0..d\}} N_j\}$. To estimate these distributions, our algorithm exploits two neural networks: a *policy network* conditioned by a set of parameters θ and a *target network* conditioned by the parameters θ'. The policy network is used at decision time to estimate the return distributions $Z_\theta(s, a)$ obtained by performing action a from state s. The target network is only used when updating the policy. It allows for stabilizing learning and enables estimating the target return distribution $\widehat{\mathcal{T}}Z(s, a)$.

Action Selection [17] showed that to learn optimal policies for ESR, the agent's policy needs to be conditioned by the return it has

accumulated until the time of decision. To choose which action to perform, our algorithm takes into account the current state s_t and the accumulated return $\mathbf{r}_{\text{acc}}$ of the agent to estimate for each action a its utility $Q(s_t, a)$. Algorithm 2 in Appendix A[1] details how the action selection procedure is performed by MO-CDQN. For each action, the algorithm starts by predicting the multivariate future return $Z(s, a)$ (Line 1). The accumulated return of the agent $\mathbf{r}_{\text{acc}}$ is used to compute the new support $\mathcal{S}'$ of the predicted distributions. The new support is computed by translating the initial support $\mathcal{S}$ with the vector $\mathbf{r}_{\text{acc}}$ and is given by: $\mathcal{S}' = \bigtimes_{j\in\{0..d\}} \mathcal{S}'_j$, where $\mathcal{S}'_j = \{\mathbf{r}_{\text{acc}_j} + V_{\text{MIN}_j} + \Delta z_j \cdot i \mid 0 \leqslant i < N_j\}$ (Line 2). The atoms of the new support are then scalarized (Line 3) and multiplied by their probability to compute the Q-values of the actions (Line 4). The action performed by the agent is finally selected by sampling from the softmax distribution obtained from the predicted Q-values (Lines 5–7).

Update Rule To train the neural network to accurately predict state-action values, we use temporal difference learning [22]. At each update step, a batch of transitions is sampled from the replay buffer to compute the temporal difference loss. The temporal difference error is computed for each sample transition (s, a, r, s'). The target return distribution for the sample is first computed as $\widehat{\mathcal{T}}Z(s, a) = r + \gamma \cdot Z_{\theta'}(s', a^*)$ (Line 5). This is performed by shrinking the support of $Z_{\theta'}(s', a^*)$ by a factor of γ and then translating said support by r (cf. [3, 16]). The target distribution is then projected to the original support $\mathcal{S}$ by distributing the probability mass of each atom of the support of $\widehat{\mathcal{T}}Z(s, a)$ to its immediate neighbors in $\mathcal{S}$, giving us the projected target distribution $\Phi\widehat{\mathcal{T}}Z(s, a)$ (Line 6). The cross-entropy between these two distributions, $Z_\theta(s, a)$ and $\Phi\widehat{\mathcal{T}}Z(s, a)$, is computed and used to estimate the gradient of the parameter set θ. Gradients are accumulated through all the sample transitions in the batch; the policy and the target network are updated with the aggregated gradient (Lines 12–13). Algorithm 3 in Appendix A presents the update procedure of the policy and target network.

Validity of the Algorithm This section discusses the theoretical validity of our MO-CDQN algorithm. Consider a policy π and a return function Z, where Z assigns a distribution over returns to each state-action pair. [3] introduced the *distributional Bellman operator* $\mathcal{T}^\pi$, defined as:

$$\mathcal{T}^\pi Z(s, a) = r(s, a) + \gamma Z(s', a') \tag{2}$$

where $s' \sim P(\cdot \mid s, a)$ and $a' \sim \pi(s')$.

It has been shown by [3, 29] that, for any initial return function, the iterative application of $\mathcal{T}^\pi$ converges to a unique fixed point Z^π, which corresponds to the true return distribution induced by policy π. Formally,

$$\lim_{n\to\infty} (\mathcal{T}^\pi)^n Z = Z^\pi, \tag{3}$$

with

$$Z^\pi(s, a) \overset{d}{=} \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \qquad s_0 = s, a_0 = a$$
$$s_{t+1} \sim P(\cdot \mid s_t, a_t) \quad a_{t+1} \sim \pi(s_{t+1}) \qquad \forall t \geqslant 0,$$

[1]A version of the paper containing the appendices can be found at https://hal.science/hal-05517093v1

where $\overset{d}{=}$ denotes equality in distribution. Furthermore, [29] showed that the *distributional Bellman optimality operator* admits a fixed point corresponding to an optimal return distribution. This fixed point has the property that its expected return—when scalarized using a linear scalarization function—coincides with the optimal value function obtained through classical (scalar) Bellman optimality. While prior work (e.g., [29]) established this property, our work extends this result to a broader class of scalarization functions. We first define two operators necessary to the analysis:

Definition 4.1 (Distributional Bellman Optimality Operator). Let Z be a return function, a function assigning a probability distribution over $\mathbb{R}^d$ to each state-action pair (s, a). Define the distributional Bellman optimality operator $\mathcal{T}$ applied in Algorithm 3 as follows:

$$\mathcal{T}Z(s, a) = \mathbf{r} + \gamma Z(s', a^*),$$

where $a^* = \arg\max_{a'} \mathbb{E}_{\mathbf{z}\sim Z(s',a')} [f(\mathbf{r} + \gamma\mathbf{z})]$, $s' \sim P(\cdot|s, a)$, and $\mathbf{r} \sim R(\cdot|s, a)$.

Definition 4.2 (Expected Scalarized Operator). Given a distribution $Z(s, a)$ over vectors in $\mathbb{R}^N$, the expected scalarized operator E_u is defined by:

$$\mathrm{E}_u Z(s, a) = \mathbb{E}_{\mathbf{z}\sim Z(s,a)} [u(\mathbf{z})]$$

where $u : \mathbb{R}^N \to \mathbb{R}$ is a possibly nonlinear scalarization function that satisfies Lipschitz continuity and has a Lipschitz constant of K.

Theorem 4.3 states that the composition of these two operators is a contraction on MO-MDPs with a discount factor of γ and K-Lipschitz continuous scalarization when $\gamma K < 1$.

Theorem 4.3. *Let Z_1 and Z_2 be two return functions. Assume the scalarization function $u : \mathbb{R}^N \to \mathbb{R}$ is K-Lipschitz with respect to the ℓ_1 norm. Then, the composed operator $\mathrm{E}_u\mathcal{T}$ satisfies:*

$$|\mathrm{E}_u\mathcal{T}Z_1 - \mathrm{E}_u\mathcal{T}Z_2| \leqslant \gamma K |\mathrm{E}_u Z_1 - \mathrm{E}_u Z_2|$$

In particular, if $\gamma K < 1$, the operator is a contraction.

Using this result, we derive Corollary 4.4, ensuring that the policy learned by our algorithm achieves the same expected scalarized return as an optimal policy.

Corollary 4.4 (Optimality of the Fixed Point). *Let Z^* be the unique fixed point of the composed operator $\mathrm{E}_u\mathcal{T}$ and suppose the scalarization function u is K-Lipschitz, such that $\gamma K < 1$. Then, the expected scalarized return of Z^* corresponds to the optimal scalarized value function:*

$$\forall (s, a), \mathrm{E}_u Z^*(s, a) = \max_\pi \mathbb{E}_\pi \left[u\left(\sum_{t=0}^{\infty} \gamma^t \mathbf{r}_t \right) \middle| s_0 = s, a_0 = a \right]$$

The proofs of Theorem 4.3 and Corollary 4.4 are provided in Appendix D.

5 OPTIMIZING SEVERAL SCALARIZATION FUNCTIONS WITH OFF-POLICY EXPERIENCE UNDER ESR

In the previous section, we assumed that the scalarization function was known and fixed throughout the learning process. However, in many applications, it is desirable or even necessary to optimize multiple scalarization functions simultaneously to capture different

trade-offs or user preferences. This is often the case in decision-support scenarios [12], where designers can optimize various utility functions to better understand user preferences. It also applies when users aim to enforce a desired behavior through a family of parameterized scalarization functions but are unsure of the best parameter values. For example, a user seeking fairness across objectives might use the Generalized Gini Index (GGI), yet still struggle to select parameters that balance fairness and efficiency. More practically, algorithms like [4] rely on this strategy to learn a diverse set of policies. In fact, the algorithm proposed by [4] is a two-step algorithm where the first step consists of generating a set of diverse scalarization functions that are later used to provide a diverse set of solutions.

This section extends the MO-CDQN algorithm to learn optimal policies for multiple scalarization functions. The Distributed MO-CDQN we propose (Algorithm 1) addresses the gap in multi-policy optimization under Expected Scalarized Return (ESR) by reusing experience across several learning instances, each targeting a different scalarization function.

The intuition behind Distributed MO-CDQN resides in the off-policy nature of our algorithm. Suppose we are given two scalarization functions $u, u' : \mathbb{R}^d \longrightarrow \mathbb{R}$ and let $\pi_u^{t_i}$ and $\pi_{u'}^{t_i}$ be the policies of two agents optimizing u and u', respectively. It is possible to use experience collected under π_u to update $\pi_{u'}$ and vice versa. With this in mind, we assume that we are given a finite set of possible scalarization functions that can be considered online by the decision-making agent. Our approach aims at learning policies that can fit different scalarization functions without requiring retraining. We thus extend our single-policy algorithm MO-CDQN to account for a set of M scalarization functions $U = \{u_1, \ldots, u_M\}$ by introducing M artificial agents (referred to as learners), where each agent l_i is assigned the scalarization function u_i and searches for the policy $\pi_{u_i}^* = \arg\max_\pi \mathbb{E}\left[u_i\left(\sum_{t=0}^{\infty} \gamma^t r^t\right) \mid \pi, s_0 \sim \mu\right]$. However, these learners are not trained separately. Instead, we propose to share environment experiences between the learners. More precisely, we explore two distinct action selection strategies to balance exploration and exploitation across the set of M learners. In the first strategy, referred to as *per-step action sampling*, each learner l_i proposes an action $a_i^t \sim \pi_{u_i}(s_t)$ based on its policy at every environment timestep t[2]. A single action is then randomly sampled from the set of proposed actions $\{a_1^t, \ldots, a_M^t\}$ and executed in the environment, yielding the experience tuple $(s_t, a_t, \mathbf{r}_t, s_{t+1})$, which is shared across all learners for policy updates. In the second strategy, termed *per-episode learner selection*, a single learner is randomly selected at the beginning of each episode, and its policy π_{u_i} is followed exclusively for the entire episode. This means all actions within that episode are sampled from the chosen learner's policy, generating a sequence of experience tuples that are again shared across all learners. The policy of each learner is then updated using the rule described in Algorithm 3.

Validity of the Algorithm We briefly discuss the theoretical validity of the multi-policy algorithm. Specifically, we address the following question: how can the policy of learner l_i in charge of optimizing the scalarization function u_i converge to its corresponding optimal policy while exploiting experience collected from other learners?

The answer to this question resides in the off-policy nature of the algorithm, meaning an agent's policy can be updated from experience collected from any other policy. A popular example of this is the use of ϵ-greedy policies to improve exploration in Q-learning [27]. With this in mind, when updating the policy of the i-th learner using experience tuple $(s_t, \text{acc\_r}_t, a_t, \mathbf{r}, s'_t, t)$, we can distinguish two cases. If the algorithm picked the action proposed by l_i (or by another learner with the same greedy action), then $a_t = \pi_{u_i}$ is a greedy action from l_i's perspective. When the algorithm picks an action different from the one proposed by l_i, i.e., $a_t \neq \pi_{u_i}$, then the action a_t can be considered as an exploratory action from l_i's point of view. For learners that had proposed the action that was actually executed, the selected action represents exploitation, whereas for other learners it represents exploration. Thus, the action selection procedure of the algorithm does not hinder its convergence properties but also improves each learner's exploration. The algorithm also allows for concurrent policy updates, making it more advantageous than applying a single-policy algorithm sequentially to learn multiple policies one by one. Algorithm 1 uses this property to reduce the number of environment interactions needed to learn multiple policies, resulting in more frequent updates and faster learning than learning policies sequentially.

Algorithm 1 Distributed MO-CDQN

Require: Number of objectives d, Set of scalarization functions to optimize $U = \{u_1, \ldots, u_M\}$, Training budget B

for $i = 1$ to M **do**
  Create learner l_i with replay buffer D_i, value network Z_θ^i, and target $Z_{\theta'}^i$
end for
for $t = 1$ to B **do**
  $\text{acc\_r} \leftarrow \vec{0}$; $c \leftarrow 0$
  Initialize environment and receive initial state s_0
  $s_t \leftarrow s_0$
  while $\neg$ done **do**
    proposed_actions $\leftarrow \{\}$
    for $i = 1$ to M **do**
      $a_i \leftarrow \text{Dist\_Action\_Selection}(s_t, \text{acc\_r}, Z_\theta^i, u_i)$
      Add a_i to proposed_actions
    end for
    $a_t \leftarrow \text{RandomChoice}(\text{proposed\_actions})$
    Perform action a_t in the environment and observe reward $\mathbf{r}_t$, next state s_{t+1}, and done flag
    for $i = 1$ to M **do**
      Add $(s_t, \text{acc\_r}, a_t, \mathbf{r}_t, s_{t+1}, c)$ to D_i
      Update Z_θ^i using distributional Bellman update described in Algorithm 3
    end for
    $\text{acc\_r} \leftarrow \text{acc\_r} + \gamma^c r_t$; $s_t \leftarrow s_{t+1}$; $c \leftarrow c + 1$
  end while
end for

[2]The action of learner l_i is sampled from the distribution $\pi_{u_i}(a|s_t)$ as described in Algorithm 2.

6 EXPERIMENTS

In this section, we compare the performance of the proposed algorithms to state-of-the-art algorithms proposed in [16, 17] on several environments of the MO-Gymnasium suite of benchmarks[3]. This section briefly describes those environments and the baselines and then dives into analyzing the obtained results. The experimental protocol followed to train and evaluate the different algorithms is described in the following section, while Appendix C gives an empirical argument for the validity of our algorithm.

6.1 Evaluation Scenarios and Baselines

We evaluate the algorithms on the Fruit-Tree-Navigation (FTN), Deep-Sea-Treasure (DST), Fishwood, Four-Room, and Minecart environments from the MO-Gymnasium benchmark suite [9], commonly used to assess ESR optimization in multi-objective RL. Full environment details are provided in Appendix C.

MO-CDQN and its multi-policy extension, Distributed MO-CDQN, are compared to state-of-the-art multi-objective RL algorithms that learn optimal policies for ESR. Specifically, the EUPG [17] and MOCAC [16] algorithms are used as baselines. Additionally, we compare our algorithm to the Distributionally Pareto-Optimal Multi-Objective Reinforcement Learning approach proposed in [4] while using the state-of-the-art single-objective RL algorithm PPO. For each scalarization function, this approach transforms the MO-MDP into a scalarized MDP with the same optimal policy, then applies PPO to this MDP. To demonstrate the limitations of the approach proposed by [8], their algorithm is tested on a small toy MO-MDP, with results presented in Appendix B.

6.2 Experimental Protocol

6.2.1 Single-Policy Algorithms. We describe the experimental protocol used to evaluate single-policy algorithms, including training budgets, scalarization functions, and key hyperparameters (learning rate, batch size, number of atoms per objective).[4] All policy-based methods use a neural network with one hidden layer of 64 units to represent the policy (and value function for MOCAC), while MO-CDQN relies on a dueling architecture [26] with the same backbone. Algorithms are evaluated periodically by freezing the current policy, executing it multiple times in the environment, and reporting the expected scalarized return. To alleviate the impact of random initialization of the policies, experiments are repeated over 11 seeds for short tasks (Fruit-Tree-Navigation, Deep-Sea-Treasure) and 5 seeds for longer tasks (Fishwood, Minecart, Four-Room, MountainCar), and the median score is reported.

Fruit-Tree-Navigation: The objective is to maximize the product of three objectives. Our method is trained for 500k environment steps, while baselines use 1M steps. Each objective's value is normalized between $[0, 1]$. Policies are evaluated every 5k steps over 100 episodes. Policy-based methods use a learning rate of $5 \cdot 10^{-5}$, MO-CDQN uses 10^{-4}, and both MOCAC and MO-CDQN discretize objectives using 21 atoms in $[0, 1]$.

Deep-Sea-Treasure: We follow the motivating scenario from [16], using the same piecewise scalarization function. Our method is trained for 100k steps, while baselines are trained for 200k steps, with evaluations every 1k steps. All algorithms use a learning rate of 10^{-4}, and MOCAC and MO-CDQN discretize objectives using 21 atoms with ranges $r_0 \in [0, 125]$ and $r_1 \in [-25, 0]$.

Fishwood: We adopt the setting of [17] with scalarization $u(r_1, r_2) = \min(r_1, \lfloor r_2/2 \rfloor)$. Policy-based methods are trained for 1M steps, while MO-CDQN uses 500k steps. Policies are evaluated every 5k steps over 100 episodes. EUPG and MOCAC use a learning rate of 10^{-3}, while MO-CDQN uses 10^{-4}. MOCAC and MO-CDQN use 11 atoms with objective ranges $r_1 \in [0, 20]$ and $r_2 \in [0, 100]$.

Four-Room: Agents aim to maximize the product of three objectives. Policy-based methods are trained for 1M steps, MO-CDQN for 500k steps. All methods use a learning rate of 10^{-4}, with MOCAC and MO-CDQN discretizing objectives using 11 atoms in $[0, 4]$. Policies are evaluated every 5k steps over 100 episodes.

Minecart: We focus on the two ore-collection objectives, normalized to $[0, 1.5]$, and use their product as scalarization. MO-CDQN is trained for 2.5M steps, while policy-based baselines are trained for 5M steps. Policies are evaluated every 10k steps over 100 episodes. EUPG uses a learning rate of 10^{-3}, others use 10^{-4}, and MOCAC and MO-CDQN use 21 atoms.

MountainCar: This continuous-state task uses three objectives: total time t, forward actions f, and backward actions b, with scalarization $u(t, f, b) = -(t^2 + f \cdot b)$. The evaluation protocol and hyperparameters match those of Minecart, except that MOCAC and MO-CDQN use 5 atoms per objective.

6.2.2 Multi-Policy Algorithm. Our multi-policy algorithm is evaluated on three of the previously presented tasks, namely the Fruit-Tree-Navigation task, the Four-Room task, and the Minecart task. The FTN and Minecart tasks are limited to 2 objectives with 5 hand-picked non-linear scalarization functions (u_1 to u_5). To showcase the scalability of the distributed algorithm with respect to the number of objectives, we consider the Four-Room task with 5 other hand-picked scalarization functions (u_6 to u_{10}). The considered scalarization functions for these tasks are the following. Note that $r^{\uparrow}$ is the vector obtained by reordering the components of the reward vector in increasing order, and I is the ideal point for each task and takes the values $(9.50, 8.78)$ for the FTN task, $(1.5, 1.5)$ for the Minecart task, and $(4, 4, 4)$ for the Four-Room task.

$$
\begin{aligned}
u_1(r_0, r_1) &= \max(r_0, r_1) & u_6(r_0, r_1, r_2) &= r_0 r_1 r_2 \\
u_2(r_0, r_1) &= \min(r_0, r_1) & u_7(r_0, r_1, r_2) &= -\tfrac{(r_0 - I_0)^2 + (r_1 - I_1)^2}{2} \\
u_3(r_0, r_1) &= r_0 r_1 & u_8(r_0, r_1, r_2) &= -\tfrac{(r_0 - I_0)^2 + (r_2 - I_2)^2}{2} \\
u_4(r_0, r_1) &= w_0 r_0^{\uparrow} + w_1 r_1^{\uparrow} & u_9(r_0, r_1, r_2) &= -\tfrac{(r_1 - I_1)^2 + (r_2 - I_2)^2}{2} \\
u_5(r_0, r_1) &= -\tfrac{(r_0 - I_0)^2 + 4(r_1 - I_1)^2}{5}
\end{aligned}
$$

$$
u_{10}(r_0, r_1, r_2) = -\frac{(r_0 - I_0)^2 + (r_1 - I_1)^2 + (r_2 - I_2)^2}{3}
$$

The performance of our multi-policy algorithm is compared to that obtained by single-policy algorithms, namely MO-CDQN, EUPG, and MOCAC, and to the one obtained by DPMORL [4], the only existing multi-policy algorithm. The single-policy algorithms and the multi-policy baseline (DPMORL) are allowed a budget of $B =$ 100,000 steps to optimize each of $m = 5$ scalarization functions independently, while the proposed multi-policy algorithm is allowed a budget of $B' = B \times m =$ 500,000 to optimize all 5 scalarization functions in parallel. Each experiment is repeated 5 times. We then

[3] The code used to perform the experiments can be found in the following GitLab repository: https://gitlab.com/chouakifares/multi-objective-categorical-deep-q-networks

[4] Additional hyperparameters and implementation details are provided in Appendix C.

compare the best scores obtained by the median policy found by each algorithm.

6.3 Experimental Results

This section presents the experimental results obtained by our MO-CDQN algorithm and the considered baselines.

Comparison to Baselines We now focus on showing how our single-policy algorithm yields better results than existing state-of-the-art algorithms optimizing ESR. In all the experiments, the baselines were allowed twice as many environment steps as the proposed algorithm to showcase the superiority of our algorithm even when the baselines are given a higher budget. Figures 1b, 1a, 1c, 1d, and 1e show the evolution of the median discounted scalarized return obtained by each algorithm. The shaded regions show the fourth and sixth decile of the returns obtained by the agents. The figures clearly illustrate the superiority of our algorithm compared to the baselines. For instance, from Figure 1a, it can be seen that our algorithm requires fewer timesteps to converge to a better policy than the one found by the baselines. Figure 1b shows that our algorithm learns policies with higher returns even on tasks with a higher number of objectives. Figure 1c outlines the superiority of our algorithm in heavily stochastic environments. From the figure, we can clearly see that our algorithm converges to a better policy after less than 100,000 training timesteps while MOCAC requires more than 200,000 timesteps to achieve similar returns. Figures 1d and 1e illustrate the superiority of our algorithm on tasks with large and continuous state spaces. Figure 1d clearly shows the superiority of our algorithm, which learns a satisfying policy after only 150,000 steps, whereas EUPG and MOCAC require many more environment interactions and are unable to solve this task even when their training budget is double that given to MO-CDQN. Figure 1e shows the same tendency; although the baselines perform better on this task, our algorithm is still the only one able to converge to the optimal policy. Indeed, both baselines fail to optimally solve the task even after 5,000,000 steps. Finally, Figure 1f shows the total domination of MO-CDQN over the baselines on continuous state-space tasks with a denser reward function, where our agent learns to solve the task after just about 50,000 steps and then starts focusing on finding the optimal policy that balances the use of forward and backward actions. The remaining baselines are not even able to efficiently solve the task. Indeed, the policies learned by the baselines need more than 160 environment steps to reach the top of the mountain compared to less than 110 environment steps for the policy learned by MO-CDQN.

Results of the Multi-Policy Algorithm Tables 1 and 2 show the best expected scalarized return obtained by the policy found using each algorithm on the FTN and Four-Room tasks, respectively[5]. From the tables, we can observe that for each scalarization function, the proposed single-policy algorithm (MO-CDQN) matches or outperforms the baselines, confirming the results of the previous section. It can be observed that the multi-policy variant of our algorithm is not only competitive with the rest of the baseline. It also manages to outperform them on 9 out of 10 considered scalarization functions without requiring retraining. A very interesting observation is that the multi-policy variant of MO-CDQN obtains better results than the single-policy counterpart; this is due to the higher exploration ratio and experience sharing across the learners. This increased diversity of experience can help the algorithm escape local optima faster, whereas the single-policy algorithm might get trapped for longer, ultimately leading to better overall performance. Finally, we can also observe that both action-selection strategies result in similar returns even in the Four-Room environment. We argue that further research is needed to explore better action selection strategies or policy reusability across scalarization functions under ESR.

[5]Results obtained on the Minecart task are given in Appendix D

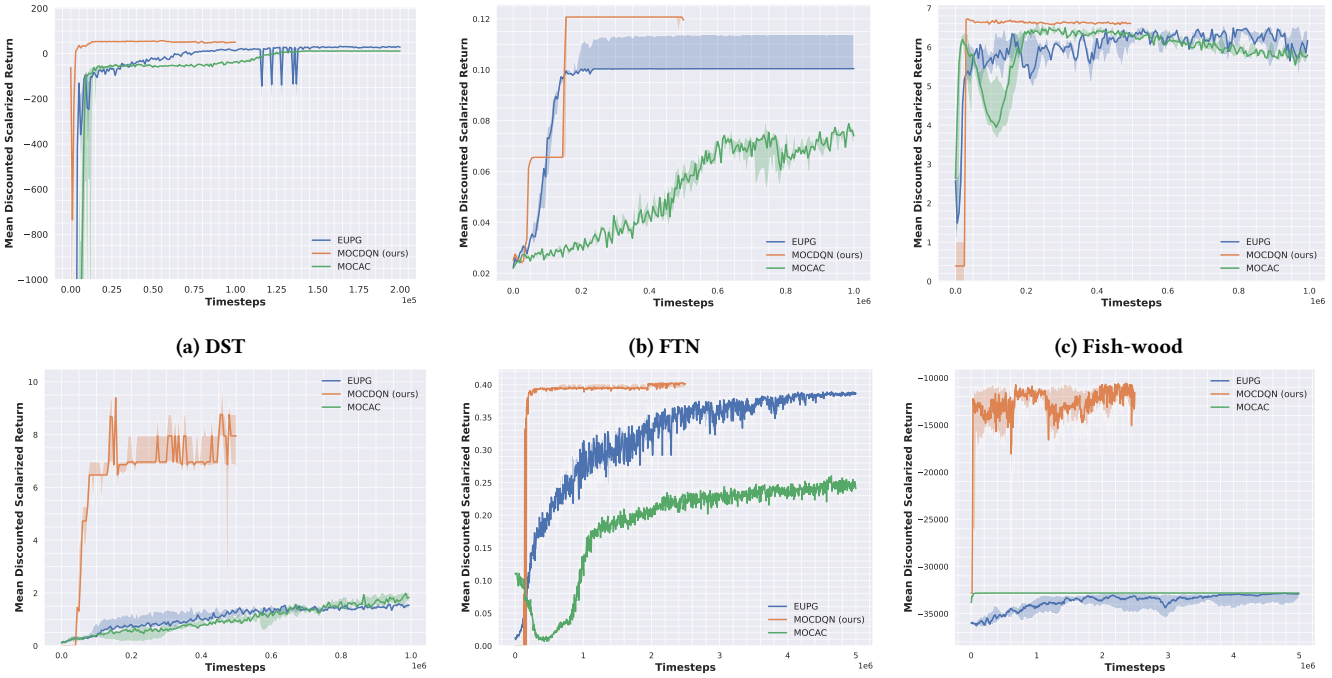
7 DISCUSSION

In this section, we discuss the performance of our algorithm and the limitations that our approach still suffers from.

Our results demonstrate that our algorithm achieves superior sample efficiency compared to state-of-the-art policy-based methods such as EUPG and MOCAC. As shown in Figure 1b, our approach converges to better policies than those learned by MOCAC and EUPG while requiring fewer timesteps. Moreover, our algorithm is the only one allowing agents to learn both deterministic and stochastic policies, unlike existing approaches that can only learn stochastic policies. Furthermore, MO-CDQN is able to leverage recent research in value-based RL such as [13] and [24]. Moreover, our multi-policy algorithm illustrates how experience can be reused to train policies optimizing different scalarization functions, further improving the sample efficiency of our approach. Both algorithms highlight the advantages of our method. However, the categorical return representation can become computationally expensive as the number of objectives increases, since the size of the support of the representation increases exponentially with the number of objectives in the environment. Another limitation of our algorithm is its inability to effectively handle continuous action spaces. Since our approach relies on a discrete set of actions, applying it to continuous control problems would require discretization, which may lead to suboptimal performance due to the loss of fine-grained control. In contrast, policy-based methods like EUPG and MOCAC inherently support continuous actions through parametrized policies.

8 CONCLUSION

This paper tackles multi-objective sequential decision-making problems where the goal is to optimize non-linear scalarization functions under ESR. A novel value-based algorithm based on learning a categorical representation of the multivariate return distribution is proposed. We provide a theoretical argument for the validity of our algorithm. Comparison results between our algorithm and other existing solutions on several benchmark MORL problems prove that the proposed solution learns better policies faster. The algorithm is then extended to propose the first value-based multi-policy algorithm able to optimize several non-linear scalarization functions under ESR. The contributed algorithms still suffer from scaling issues that will be addressed in future work. Moreover, the multi-policy algorithm suffers from a limitation in that it assumes the scalarization functions are known and explicitly specified by the users. To alleviate this issue, we plan to extend our algorithm to a two-step approach like [4], where the first step consists of generating a set of diverse non-linear scalarization functions that



(a) DST (b) FTN (c) Fish-wood

(d) Four-room (e) Minecart (f) MountainCar

Figure 1: Comparison of the performance of the proposed algorithms to the baselines on the validation and evaluation tasks

	MO-CDQN	MOCAC	EUPG	Dist MO-CDQN		DPMORL
				per-step	per-episode	
u_1	8.55 ± 0.18	8.05 ± 2.51	8.06 ± 2.29	8.52 ± 0.05	8.43 ± 0.14	$\mathbf{8.94 \pm 0.00}$
u_2	4.11 ± 0.92	2.04 ± 0.37	3.07 ± 0.42	$\mathbf{4.81 \pm 0.22}$	4.77 ± 0.16	2.99 ± 0.29
u_3	24.78 ± 5.06	10.64 ± 1.88	23.98 ± 6.46	$\mathbf{36.04 \pm 1.37}$	$\mathbf{36.04 \pm 2.65}$	24.80 ± 2.42
u_4	4.76 ± 0.19	4.47 ± 0.87	4.43 ± 1.00	$\mathbf{4.89 \pm 0.17}$	4.87 ± 0.12	4.78 ± 0.08
u_5	-6.84 ± 0.09	-23.04 ± 4.40	-22.23 ± 3.87	-4.51 ± 1.46	$\mathbf{-4.39 \pm 1.91}$	-16.88 ± 1.20

Table 1: Performance of each algorithm on each scalarization function for the FTN task (median $\pm$ standard deviation over 5 seeds).

	MO-CDQN	MOCAC	EUPG	Dist MO-CDQN		DPMORL
				per-step	per-episode	
u_6	10.12 ± 4.88	0.57 ± 0.35	0.61 ± 0.26	$\mathbf{24.64 \pm 8.46}$	20.19 ± 10.52	1.85 ± 0.31
u_7	-1.69 ± 0.87	-5.67 ± 0.53	-7.37 ± 0.74	$\mathbf{-0.64 \pm 2.58}$	-0.82 ± 0.46	-3.22 ± 0.21
u_8	-1.03 ± 0.77	-5.01 ± 0.46	-7.02 ± 0.56	-0.99 ± 3.06	$\mathbf{-0.71 \pm 1.53}$	-2.89 ± 0.08
u_9	-1.96 ± 0.25	-5.27 ± 0.28	-7.00 ± 0.57	-0.87 ± 1.29	$\mathbf{-0.80 \pm 1.78}$	-3.27 ± 0.32
u_{10}	-1.24 ± 1.87	-3.63 ± 0.30	-4.72 ± 0.40	$\mathbf{-0.40 \pm 0.68}$	-0.56 ± 2.30	-2.46 ± 0.16

Table 2: Performance of each algorithm on each scalarization function for the Four-Room task (median $\pm$ standard deviation over 5 seeds).

are then fed to the multi-policy algorithm to optimize each function and generate a diverse front of possibly optimal policies under ESR.

REFERENCES

[1] Mridul Agarwal, Vaneet Aggarwal, and Tian Lan. 2022. Multi-Objective Reinforcement Learning with Non-Linear Scalarization. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems* (Virtual Event, New Zealand) *(AAMAS '22)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 9–17.

[2] Toygun Basaklar, Suat Gumussoy, and Umit Y. Ogras. 2023. PD-MORL: Preference-Driven Multi-Objective Reinforcement Learning Algorithm. arXiv:2208.07914 [cs.LG] https://arxiv.org/abs/2208.07914

[3] Marc G. Bellemare, Will Dabney, and Rémi Munos. 2017. A Distributional Perspective on Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70* (Sydney, NSW, Australia) *(ICML'17)*. JMLR.org, 449–458.

[4] Xin-Qiang Cai, Pushi Zhang, Li Zhao, Jiang Bian, Masashi Sugiyama, and Ashley J. Llorens. 2023. Distributional pareto-optimal multi-objective reinforcement learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '23)*. Curran Associates Inc., Red Hook, NY, USA, Article 686, 21 pages.

[5] Krzysztof Ciesielski. 2007. On Stefan Banach and some of his results. *Banach Journal of Mathematical Analysis [electronic only]* 1, 1 (2007), 1–10. http://eudml.org/doc/53684

[6] Will Dabney, Georg Ostrovski, David Silver, and Remi Munos. 2018. Implicit Quantile Networks for Distributional Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 1096–1105. https://proceedings.mlr.press/v80/dabney18a.html

[7] Will Dabney, Mark Rowland, Marc G. Bellemare, and Rémi Munos. 2018. Distributional reinforcement learning with quantile regression. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence* (New Orleans, Louisiana, USA) *(AAAI'18/IAAI'18/EAAI'18)*. AAAI Press, Article 353, 10 pages.

[8] Ziming Fan, Nianli Peng, Muhang Tian, and Brandon Fain. 2023. Welfare and Fairness in Multi-objective Reinforcement Learning. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems* (London, United Kingdom) *(AAMAS '23)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1991–1999.

[9] Florian Felten, Lucas N. Alegre, Ann Nowé, Ana L. C. Bazzan, El Ghazali Talbi, Grégoire Danoy, and Bruno C. da Silva. 2023. A Toolkit for Reliable Benchmarking and Research in Multi-Objective Reinforcement Learning. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS 2023)*.

[10] Conor F. Hayes, Mathieu Reymond, Diederik M. Roijers, Enda Howley, and Patrick Mannion. 2023. Monte Carlo tree search algorithms for risk-aware and multi-objective reinforcement learning. *Autonomous Agents and Multi-Agent Systems* 37, 2 (April 2023), 37. https://doi.org/10.1007/s10458-022-09596-0

[11] Conor F. Hayes, Diederik M. Roijers, Enda Howley, and Patrick Mannion. 2022. Decision-Theoretic Planning for the Expected Scalarised Returns. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems* (Virtual Event, New Zealand) *(AAMAS '22)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1621–1623.

[12] Conor F. Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M. Zintgraf, Richard Dazeley, Fredrik Heintz, Enda Howley, Athirai A. Irissappane, Patrick Mannion, Ann Nowé, Gabriel Ramos, Marcello Restelli, Peter Vamplew, and Diederik M. Roijers. 2022. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems* 36, 1 (April 2022). https://doi.org/10.1007/s10458-022-09552-y

[13] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. 2018. Rainbow: combining improvements in deep reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence* (New Orleans, Louisiana, USA) *(AAAI'18/IAAI'18/EAAI'18)*. AAAI Press, Article 393, 8 pages.

[14] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Kirkeby Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518 (2015), 529–533. https://api.semanticscholar.org/CorpusID:205242740

[15] Mathieu Reymond, Eugenio Bargiacchi, and Ann Nowé. 2022. Pareto Conditioned Networks. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems* (Virtual Event, New Zealand) *(AAMAS '22)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1110–1118.

[16] Mathieu Reymond, Conor Hayes, Denis Steckelmacher, Diederik Roijers, and Ann Nowe. 2023. Actor-critic multi-objective reinforcement learning for non-linear utility functions. *Autonomous Agents and Multi-Agent Systems* 37 (04 2023). https://doi.org/10.1007/s10458-023-09604-x

[17] Diederik M. Roijers, Denis Steckelmacher, and Ann Nowé. 2020. Multi-objective reinforcement learning for the expected utility of the return. 2018 Adaptive Learning Agents, ALA 2018 - Co-located Workshop at the Federated AI Meeting, FAIM 2018 ; Conference date: 14-07-2018 Through 15-07-2018.

[18] 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

[19] Willem Röpke, Conor F. Hayes, Patrick Mannion, Enda Howley, Ann Nowé, and Diederik M. Roijers. 2023. Distributional multi-objective decision making. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence* (Macao, P.R.China) *(IJCAI '23)*. Article 634, 9 pages. https://doi.org/10.24963/ijcai.2023/634

[20] Manuela Ruiz-Montiel, Lawrence Mandow, and José-Luis Pérez de-la Cruz. 2017. A temporal difference method for multi-objective reinforcement learning. *Neurocomputing* 263 (2017), 15–25. https://doi.org/10.1016/j.neucom.2016.10.100 Multiobjective Reinforcement Learning: Theory and Applications.

[21] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2015. Prioritized Experience Replay. *CoRR* abs/1511.05952 (2015). https://api.semanticscholar.org/CorpusID:13022595

[22] Gerald Tesauro. 1995. Temporal difference learning and TD-Gammon. *Commun. ACM* 38, 3 (March 1995), 58–68. https://doi.org/10.1145/203330.203343

[23] Kristof Van Moffaert and Ann Nowé. 2014. Multi-objective reinforcement learning using sets of pareto dominating policies. *J. Mach. Learn. Res.* 15, 1 (Jan. 2014), 3483–3512.

[24] Nino Vieillard, Olivier Pietquin, and Matthieu Geist. 2020. Munchausen reinforcement learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '20)*. Curran Associates Inc., Red Hook, NY, USA, Article 356, 12 pages.

[25] Cédric Villani. 2008. *Optimal Transport: Old and New*. Grundlehren der mathematischen Wissenschaften, Vol. 338. Springer-Verlag, Berlin Heidelberg.

[26] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. 2016. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48* (New York, NY, USA) *(ICML'16)*. JMLR.org, 1995–2003.

[27] Christopher J. C. H. Watkins and Peter Dayan. 1992. Q-learning. *Machine Learning* 8, 3 (1992), 279–292. https://doi.org/10.1007/BF00992698

[28] Runzhe Yang, Xingyuan Sun, and Karthik Narasimhan. 2019. *A generalized algorithm for multi-objective reinforcement learning and policy adaptation*. Curran Associates Inc., Red Hook, NY, USA.

[29] Pushi Zhang, Xiaoyu Chen, Li Zhao, Wei Xiong, Tao Qin, and Tie-Yan Liu. 2021. Distributional reinforcement learning for multi-dimensional reward functions. In *Proceedings of the 35th International Conference on Neural Information Processing Systems (NIPS '21)*. Curran Associates Inc., Red Hook, NY, USA, Article 117, 11 pages.