Utility-Based Soft Masking for Continual Multi-Objective Reinforcement Learning

Timon Deschamps
Universite Claude Bernard Lyon 1, CNRS, INSA Lyon, LIRIS, UMR5205
Villeurbanne, France
timon.deschamps@univ-lyon1.fr

Rémy Chaput
CPE Lyon, Universite Claude Bernard Lyon 1, CNRS, INSA Lyon, LIRIS, UMR 5205
Villeurbanne, France
remy.chaput@cpe.fr

Mathieu Guillermin
UCLy, UR CONFLUENCE : Sciences et Humanités (EA 1598)
Lyon, France
mguillermin@univ-catholyon.fr

Laëtitia Matignon
Universite Claude Bernard Lyon 1, CNRS, INSA Lyon, LIRIS, UMR5205
Villeurbanne, France
laetitia.matignon@univ-lyon1.fr

ABSTRACT

Real-world decision-making usually involves balancing multiple, sometimes conflicting objectives, according to user preferences that can be complex and potentially non-linear. These preferences, or utilities, can also be subject to change over time, requiring continual adaptation. This challenge is at the center of continual multi-objective reinforcement learning (CMORL), and remains vastly understudied, with existing work limited to linear utilities. In this paper, we take a first step towards CMORL with non-linear utilities by proposing utility-based soft masking (UBSM). By generating a discretized representation of the utility and using it to soft-mask the policy's parameters, UBSM harnesses the structure of utility functions, allowing for greater knowledge transfer among them and supporting the learning of policies that adapt to dynamic preferences. We evaluate UBSM on classic multi-objective reinforcement learning environments, demonstrating its improvements over baselines and providing insights on the evaluation of CMORL algorithms.

KEYWORDS

Multi-objective reinforcement learning; continual learning

ACM Reference Format:
Timon Deschamps, Rémy Chaput, Mathieu Guillermin, and Laëtitia Matignon. 2026. Utility-Based Soft Masking for Continual Multi-Objective Reinforcement Learning. 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/FXDN3629

1 INTRODUCTION

Most real-world problems are multi-objective in essence, requiring to solve complex trade-offs between multiple criterion, e.g., lowering costs and workloads when routing vehicles [10], minimizing both electrical peak load and electricity cost in a residential smart grid [28], or developing human and ethically-aligned artificial intelligence [6, 34]. Multi-objective reinforcement learning (MORL) [8, 26] is the standard framework to handle multi-objective sequential decision making tasks. MORL agents learn to solve these problems by optimizing a policy based on a utility function, which captures the preferences of the decision maker over the objectives. Most works in MORL focus on linear utility functions, which are insufficient to model nuanced behaviors (minima [25], thresholds [31, 38], risk-averseness [9], fairness [29] etc.). For example, the user of an autonomous vehicle for daily commute is likely to have non-linear preferences, such as making sure the comfort level is high before increasing the speed.

In realistic scenarios, the preferences of users change over time, potentially rapidly. In the above example, the user could learn of an emergency and decide to temporarily reduce their comfort requirement to get to the destination as fast as possible. This relates to continual learning (CL) [35], a field of machine learning concerned with creating models able to learn on a sequence of tasks while improving continuously and without relearning from scratch. Continual learning has been applied to many domains, such as classification [5] and reinforcement learning [1, 12, 37], and it seems natural to define continual MORL as the study of agents that adapt to a stream of changing utility functions to optimize. However, continual MORL remains almost entirely unstudied, and the few papers tackling this challenge are limited to linear utility functions [2, 21]. The emergency example above, with a user rapidly switching between two non-linear preferences in a situation where urgency does not allow for a long re-learning period, highlights the need for further study.

In this work, we introduce the setting of continual MORL with non-linear utility functions and propose the utility-based soft masking scheme to tackle it. The key idea is to extract a specialized sub-network for each utility from a larger, shared policy architecture. This is achieved by using a discretized representation of the utility to mask the network's parameters. This approach encourages knowledge transfer between utilities, enables a single policy to adapt to diverse preferences, and is compatible with policy-based non-linear MORL algorithms.

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/FXDN3629

Our contributions can be summarized as follows:

- We characterize the setting of **continual multi-objective reinforcement learning (CMORL)**, discussing the appropriate optimization criterion and highlighting key constraints and differences with existing settings. Additionally, we extend existing CL and MORL metrics to the CMORL setting, and discuss their limits in capturing continual behaviors in complex environments.
- We propose **utility-based soft masking (UBSM)**, a method that leverages the structure of utility functions as a soft mask to specialize the agent's behaviors, and demonstrate its effectiveness for knowledge transfer in the continual setting on various benchmarks.

2 BACKGROUND

Multi-objective reinforcement learning extends classical RL to the more complex case where multiple objectives need to be balanced. It is modeled as a multi-objective Markov decision process [26, MOMDP] $\langle \mathcal{S}, \mathcal{A}, T, \gamma, \mu, \mathbf{R} \rangle$, with $\mathcal{S}$ the state space, $\mathcal{A}$ the action space, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ the transition function giving the probability of entering state s' given that the agent took action a in state s, γ the discount factor modulating the importance of short-term versus long-term actions, $\mu : \mathcal{S} \to [0, 1]$ a probability distribution over initial states, and $\mathbf{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^d$ a vector-valued reward function, giving a reward for each of the d objectives considered. [1] We denote $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ the policy, which tells the agent what action to take for each encountered state.

Contrary to the single objective setting, we cannot always maximize the sum of vectorial rewards, as some components may be conflicting. Thus, we use a *utility* (or scalarization) function $u \in \mathcal{U}$ which outputs a scalar score for any input reward vector. To simplify and without lack of generality, utility functions are assumed to be bounded between 0 and 1 and monotonically increasing, i.e.,

$$\mathcal{U} \coloneqq \{u : [0, 1]^d \to [0, 1] \mid \forall(\mathbf{r}, \mathbf{r}'),\ \mathbf{r} \succeq_P \mathbf{r}' \implies u(\mathbf{r}) \geq u(\mathbf{r}')\},$$

where $\succeq_P$ is the Pareto dominance relation. In this work, we refer to functions in this general class as *non-linear* utilities. If a priori information is known about the type of utility that reflects a user's preferences, we can restrict the utility space to $\mathcal{U}' \subset \mathcal{U}$. For example, assuming the user's preferences are a linear combination of the objectives, we can define $\mathcal{U}'$ as the space of linear utility functions $\{u(\mathbf{r}) = \mathbf{w}^\top \mathbf{r} \mid \mathbf{w} \in \Delta^{d-1}\}$, with $\mathbf{w}$ a weight vector and Δ^k the k-simplex. Intuitively, each weight $w_i \in \mathbf{w}$ represents the importance of the associated objective $i \in \{1, .., d\}$.

In MORL, two optimality criteria are commonly used in practice [8]. *Scalarized expected returns* (SER) is used when the utility is computed over multiple executions of a policy π, and when compensation over these runs is possible. When using SER, the value function is defined as $V_u^\pi = u(\mathbb{E}[\mathbf{G}^\pi \mid \pi, \mu])$, with $\mathbf{G}^\pi$ denoting the return vector when following π, defined as $\mathbf{G}^\pi = \sum_{t=0}^{\infty} \gamma^t \mathbf{R}_t$. On the other hand, one should use *expected scalarized returns* (ESR) when each policy execution is to be independently optimal, i.e., $V_u^\pi = \mathbb{E}[u(\mathbf{G}^\pi) \mid \pi, \mu]$. When the utility function is linear, the linearity of expectation renders the ESR and SER criterion equivalent. This is not the case with non-linear utility functions, for which the policies optimized according to SER and ESR criterion can be significantly different [8]. Taking the example from Section 1, say that a user wants to keep a minimum threshold for speed and comfort. A vehicle maximizing speed for the first half of the year and comfort for the second will have a great average value for both objectives, and so the utility of the expected returns will be high. In practice, however, not a single day will actually be satisfactory to the user, as one of the two thresholds will always be unmet, and the expected utility of the returns will be low. In this case, the ESR criterion is clearly more adapted to comply with the preferences of the user.

As underlined by Hayes et al. [8], many real-world applications involve unknown or dynamic utility functions. These are often addressed with *multi-policy methods*, which learn a set of optimal policies for the user to select from based on their preferences. Without a priori knowledge of the type of utility functions the user cares about, we are interested in the complete set of Pareto non-dominated policies: the *Pareto coverage set* (PCS), a minimal *Pareto front* (PF) with redundancy removed. If the utility is known to be linear, we instead focus on the *convex coverage set* (CCS), the smallest PF subset containing all optimal policies under this assumption.

3 RELATED WORK

Our work addresses the challenge of multi-objective reinforcement learning with non-linear utilities in the continual setting. To the best of our knowledge, this intersection has not yet been explored in the literature. Therefore, this section reviews related work in both fields to help contextualize our contribution.

3.1 MORL with Non-Linear Utilities.

The vast majority of multi-objective RL works focuses on the simpler linear utility setting, as it reduces the MORL problem to a single-objective one when the utility is known, or restricts the policy search to a CCS otherwise. Although non-linear utilities can model much richer preferences, they have received considerably less attention in the literature, for several reasons: (i) non-stationary or stochastic policies must be considered as they can reach convex regions of the PCS [32, 36], greatly complexifying the space of potentially optimal solutions; (ii) the ESR and SER criteria produce significantly different policies, dividing the research between two incompatible optimization targets; (iii) non-linear utility functions break the Bellman equation, rendering many traditional methods impossible to use [8, 22]. Hence to tackle non-linear utility functions, approaches usually introduce non-stationarity in the policy. This is often done by conditioning the policies on the current timestep [23] or by splitting the return into a past (also known as accrued) and future component, as in Pareto Q-learning [20, PQL].

As noted by Hayes et al. [8], the ESR criterion remains understudied in the MORL literature. Since traditional value function-based algorithms cannot be used, works tackling ESR usually employ a policy-based approach. Expected utility policy gradient [25, EUPG] is able to optimize for ESR by learning in a Monte Carlo fashion, allowing to incorporate the utility of the full trajectory in the loss function, and by conditioning the policy on the accrued reward vector. Multi-objective categorical Actor-Critic [24, MOCAC] uses a distributional critic that estimates a categorical distribution over the returns to learn policies under ESR. Distributional Pareto-optimal

[1] Note that the $d = 1$ case is equivalent to single objective reinforcement learning.

MORL [4, DPMORL] is a multi-policy approach which learns a set of ESR-optimal policies. The algorithm first generates a diverse set of non-linear utility functions, and then optimizes one specialized policy for each. While effective, DPMORL suffers from poor scalability, as it requires storing the independent weights of a full policy for every utility learned, without possibility to share knowledge between them. Other methods [30] optimize for ESR, but are restricted to special classes of utility functions, limiting their applicability when no a priori knowledge about the user's preferences is available.

3.2 Continual Learning

As the continual learning literature is large, we focus here on approaches most similar to ours, working at the parameter level to isolate sub-networks for each task. Specifically, these methods focus on task-incremental learning, a specific setting of CL in which the model learns a sequence of tasks with access to their identity at test time. This setting is particularly relevant to our work due to the strong parallel between tasks and utility functions (see Section 4.1). SPG [14] proposes to locally mask the gradient flow, and performs better than many baseline methods while using a constant algorithmic memory overhead of the size of the network. However, it introduces a computational step at each task that grows linearly in the number of tasks seen, and adds a task-specific classification head for each task (often seen in task-incremental learning), both features which could limit its applicability to large task numbers. Context-dependent gating [19, XdG] proposes to "turn off", or gate, a fixed proportion of hidden units for each task. While it exhibits good performance when used in combination with other CL methods and requires minimal computational overhead, XdG necessitates storage of the identity of gated units for each task, resulting in a space complexity of $O(N \cdot |\boldsymbol{\theta}|)$, with N the number of tasks encountered and $|\boldsymbol{\theta}|$ the size of the learning network. Packnet [18] explores a similar idea, but freezes some of the weights after learning on each task. This allows the error to remain stable when additional tasks are introduced, but has a memory footprint of $O(\log_2(N) \cdot |\boldsymbol{\theta}|)$, and limits the total number of tasks that can be learned on. Elastic weight consolidation [13, EWC] proposes to add a quadratic penalty to the loss function to discourage large updates on weights important to previous tasks. This importance is estimated using the Fisher information matrix, allowing EWC to work with a constant memory overhead of the size of the network.

4 CONTINUAL NON-LINEAR MORL

In this section, a definition of the continual MORL setting under non-linear utilities is presented. Then our proposed utility-based soft masking approach is detailed.[2] Finally we discuss existing metrics and propose extensions thereof to better fit our context.

4.1 Problem Setting

In this work, we aim to tackle the continual MORL problem, which we define in this section. Over its lifetime, the agent interacts with a fixed MOMDP while attempting to optimize a sequence of N potentially non-linear utilities $u_1, \dots, u_N \in \mathcal{U}'$. The agent trains on each encountered utility for n timesteps (corresponding to a variable number of episodes), but does not know in advance the utilities to come or the total length of the sequence. The CMORL agent is optimized using the ESR optimization criterion to maximize the per-episode performances with respect to the current utility function (see Section 3). SER is inapplicable in this setting, as it optimizes the utility of the average return across many episodes. Indeed, compensation between episodes is undesirable and the returns are not assumed to accumulate across them [26], especially since different episodes can be optimized for different utilities. Furthermore, using SER implies aiming to optimize for the long-term utility of averages, which is not feasible when the duration of the sequence and the number of timesteps is not known. Ideally, the agent should also improve on previously encountered utilities (backward transfer), and learn to generalize on yet unseen ones (forward transfer). The complete CMORL process is depicted in Figure 1.

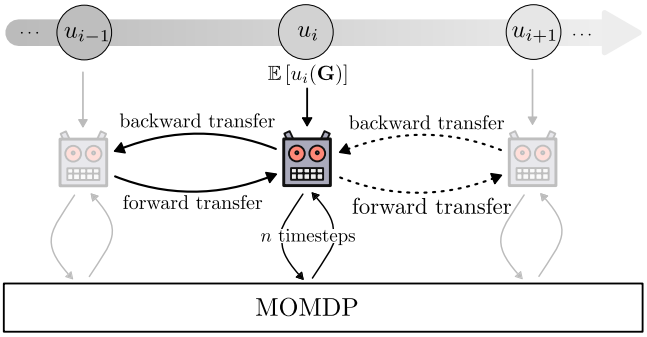

Figure 1: Illustration of the continual MORL process. The agent in the center acts according to the policy $\pi_{i,i}$, but $\pi_{i,j\neq i}$ can be tested to evaluate it on previous or future utilities. Current experience influences performances on past and future utilities, providing in the best case a positive forward and backward transfer, and inducing forgetting in the worst.

Standard multi-policy methods cannot be used directly in this CMORL setting, as they often rely on a two-phase process, where a set of policies is first generated according to some utility distribution, from which utilities are selected either automatically or by the decision maker at deployment time. In our case, however, the continual agent is deployed in the real world without an initial phase to explore the utility space, and as such is immediately evaluated at the start of learning and must learn to adapt and perform as fast as possible.

Hayes et al. [8] propose several scenarios to motivate multi-objective reinforcement learning, one of which (scenario e) is close to our formulation of CMORL, as it considers a *dynamic utility function*. In it, the agent learns a finite number of policies over time, and selects the optimal one for the current utility while continuing to improve them. However, we have in general no prior information about the length of the utility sequence. As such, we are interested in computationally and memory-efficient algorithms that do not require resources scaling with the number of utilities encountered (e.g., we forbid learning a mapping $u_i \rightarrow \pi_{\cdot,i}$ [3] that assigns an independent policy to each seen utility). Thus, the agent has access

[2] The source code is available at https://gitlab.liris.cnrs.fr/acceler-ai/UBSM.

[3] We denote $\pi_{i,j}$ the policy trained up to utility u_i, and deployed (for training or testing) on utility u_j.

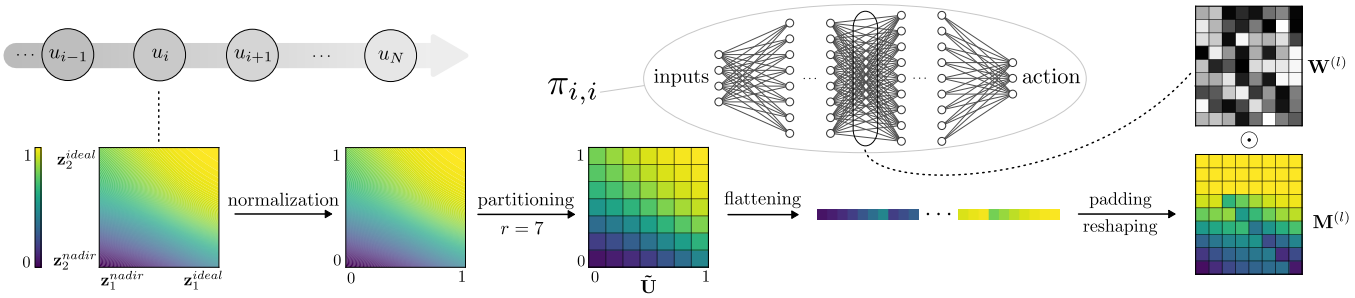

Figure 2: Illustration of our utility-based soft-masking method. The target utility function (of dimension 2) is normalized, segmented with resolution r, flattened, padded and reshaped to create the soft mask $\mathbf{M}^{(l)}$, matching the dimension of the weight matrix $\mathbf{W}^{(l)}$. Finally, the mask is applied element-wise to the weights in the forward pass.

to the current utility but does not retain information about previously seen utilities, forcing it to consolidate knowledge without relying on an external (growing) memory. This ensures that the algorithms could realistically be deployed in real-world scenarios, a key consideration in the MORL literature [8]. While the authors argue that the infinite space of policy functions can be covered with a finite number of policies, it seems that they only consider linear utilities. Indeed, while it is true that any linear utility has an optimal policy on the finite CCS, non-linear utilities can have optima in non-convex regions and hence require the entire PCS, which may contain an infinite number of solutions. Hence, in the non-linear CMORL case, we propose to restrict this scenario by constraining the agent to a single fixed-capacity policy throughout its deployment. This restriction poses the challenge of representing multiple policies with a single network, each optimized for a different non-linear utility.

While, to our knowledge, this full non-linear CMORL setting has not been studied in the literature, a few methods have explored time varying weighting of the objectives in the linear case [2, 15, 21], mainly by conditioning the policy on the current weight vector.

The following section details our approach to the complete proposed setting. We first introduce a method to represent arbitrary non-linear utilities with a finite number of parameters. We then show how this representation can be used to modulate the policy while mitigating the inevitable curse of dimensionality that arises when the number of objectives increases.

4.2 Utility-Based Soft Masking

To enable a single policy network to adapt to a wide range of non-linear utility functions, it must be able to adjust its behavior based on the utility being optimized. To do so, we propose to dynamically extract a specialized sub-network for each utility from a larger, shared architecture. We achieve this by creating a unique soft mask from the utility function, which then modulates the weights of the policy. This encourages the network to dedicate different subsets of its parameters to different preferences, supporting specialization while still allowing for knowledge transfer. We call this approach **utility-based soft masking (UBSM)**.

To generate a mask, we first need a standardized, discrete representation of the utility function. As the entire class of non-linear utility functions cannot be captured with a finite number of parameters, we must introduce assumptions to constrain this space. When domain knowledge about the problem is available (e.g., we know the class of utility function the system will encounter), it can be used as an inductive bias and incorporated into the learning agent to help it perform best (e.g., lexicographic preferences [31, 38]).

However, it is often the case that no initial knowledge about the user's preferences is available, or that the user is not initially able to formulate them precisely. In this case we assume that the complexity of the utility functions is uniformly distributed over their domain, meaning no region of the objective space is a priori more important than another. Our process therefore begins by normalizing the domain of the utility into the d-dimensional unit hypercube $[0, 1]^d$ using the ideal and nadir vectors (denoted $\mathbf{z}^{ideal}$ and $\mathbf{z}^{nadir}$), i.e., vectors containing the maximum and minimum returns for each objective.[4] We then partition this hypercube into a grid of r^d identical cells, where the resolution hyperparameter $r \in \mathbb{N}$ controls the fidelity of the representation. Finally, by averaging the utility over each cell, we obtain the **utility-representation tensor** $\tilde{\mathbf{U}} \in [0, 1]^{r^d}$, a normalized and discretized version of the original utility u. This tensor, from which we will derive the soft mask, serves as a task-descriptor [16], allowing the learning agent to use contextual information about the utility it is training on.

When tackling a wide range of preferences, most linear utility MORL methods condition the policy or value network used by the agent with the weight vector $\mathbf{w}$, which greatly helps with generalization. As our tensor representation of u scales exponentially (rather than linearly) in the number of objectives d, it quickly becomes unusable as a conditioning. For high values of r and d, this input tensor would require a large increase in the network's first layer, promoting overfitting and dominating the original state input.

To avoid these issues, we propose to use $\tilde{\mathbf{U}}$ as a soft mask which modulates the weights of the learning network. In a standard feed-forward layer l, the output activation $\mathbf{a}^{(l)}$ is computed using the output of the previous layer $\mathbf{a}^{(l-1)}$ as $\mathbf{a}^{(l)} = \sigma(\mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)})$, where $\mathbf{W}^{(l)} \in \mathbb{R}^{n \times m}$ is the weight matrix of size $n \times m$, $\mathbf{b}^{(l)} \in \mathbb{R}^n$ is the bias vector, and $\sigma(\cdot)$ is a non-linear activation function.

[4] Note that this requires having access to these vectors or estimations thereof, a common assumption in the non-linear MORL literature [4, 24].

U	Te_1	Te_2	Te_3
Tr_1	U^*	$U_{1,2}$	$U_{1,3}$
Tr_2	$U_{2,1}$	U^*	$U_{2,3}$
Tr_3	$U_{3,1}$	$U_{3,2}$	U^*

FAU: Final row
RAU: +
FWT:
BWT:

Table 1: Example train-test utility matrix for a sequence of $N = 3$ utilities. The metrics are computed using the color-coded cells as indicated. Adapted from [7].

To use our utility-representation tensor as a mask, we first flatten it into a vector of length r^d, then pad it with ones up to size $n \times m$,[5] and finally reshape it into the soft-masking matrix $\mathbf{M}^{(l)} \in [0,1]^{n \times m}$. A value of 1 is used for the padding to ensure that weights not modulated by the soft-mask remain fully active, effectively serving as a shared set of utility-agnostic parameters. We modify the previously defined forward pass as follows:

$$\mathbf{a}^{(l)} = \sigma(\mathbf{M}^{(l)} \odot \mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}),$$

with $\odot$ the Hadamar (or element-wise) product. This entire process, which we call utility-based soft masking (UBSM), is presented in Figure 2. As a simple modification of the forward pass, our method is highly generic and could be applied to many architectures targeting the optimization of non-linear utilities in MORL. In our experiments, we apply UBSM on all hidden layers of a policy network, following the masking scheme of Masse et al. [19].

As opposed to using $\mathbf{M}$ as an input to condition the policy, this utility-based soft masking scheme allows keeping the resolution reasonably high even for a moderate number of objectives, as the number of weights in a neural network layer is typically large enough. It also encourages the policy network to reuse a common set of weights for similar utilities, while activating distinct weights for utilities with very different behaviors. As the agent faces new utility functions, its masked policy will focus on weights that were "pre-trained" on similar utilities, promoting transfer. The use of different sets of weights for different utilities serves a similar purpose to elastic weight consolidation [13], which uses the Fisher overlap to determine wether tasks depend on the same set of weights, but without the additional computational and memory overhead. Structurally, our approach is closer to context-dependent gating [19], but 1) we use soft rather than hard masking to enable better transfer, and 2) our mask is derived using the structure of the utility, instead of targeting a fixed random subset of units.

Note that we have not yet discussed the choice of the hyperparameter r. Ideally, if domain knowledge is available, one should either use a specific parametrization of u or select a resolution parameter allowing $\tilde{\mathbf{U}}$ to capture important features of the kind of utilities that will be seen. In the absence of such information, we propose to use the largest possible r that allows $\tilde{\mathbf{U}}$ to fit as a mask for the network layers' weight matrices, that is, $r = \lfloor \sqrt[d]{n \times m} \rfloor$.

[5]To avoid losing information about the utility, we need r^d to be less or equal to $n \times m$, which we deem to be a sensible assumption for reasonable values of r and typical neural network sizes.

4.3 Metrics

Evaluating the capabilities of continual RL agents is a highly complex task and remains an open research question to this day [12]. In this section, we discuss appropriate metrics for the CMORL setting, drawing from commonly used ones in CL and MORL. We first propose to adapt existing CL metrics to the utility-based MORL context, and then extend the expected utility metric (EUM), a standard of multi-policy linear MORL, to the continual non-linear setting.

4.3.1 Adapting continual learning metrics to CMORL. As detailed in Section 4.1, the agent encounters multiple utilities over its lifetime. After learning on an utility, the agent is tested on all utilities of the sequence. This allows us to construct a train-test utility matrix $\mathbf{U} \in \mathbb{R}^{N \times N}$ (depicted in Table 1), where $U_{i,j}$ is the average utility reached by the agent when tested on utility u_j after being trained up to utility u_i. We adapt continual learning metrics from previous works [7, 37] to evaluate the performance of the agent across the sequence of utilities.

Average utility. Using U, we can define final average utility as $FAU = \frac{1}{N}\sum_{j=1}^{N} U_{N,j}$. FAU is used as a measure of the average performance of the agent on all N utilities encountered at the end of training. Following Díaz-Rodríguez et al. [7], we also use the running average utility $RAU = \frac{2}{N(N+1)}\sum_{i=1}^{N}\sum_{j=1}^{i} U_{i,j}$, which evaluates the performance across all previously seen utilities ($u_{i,j}$ with $j \leq i$) each time a new one is introduced.

Forward transfer. Representing the impact of learning using an utility on the performance on future utilities, the forward transfer is defined as $FWT = \frac{2}{N(N-1)}\sum_{j=2}^{N}\sum_{i=1}^{j-1} U_{i,j}$. Typically, "positive" forward transfer can be achieved when utilities are highly correlated, or when the learning algorithm is able to generalize beyond the current utility using some representation.

Backward transfer. Backward transfer measures the influence that learning on a utility has on the performance on previous ones. It is defined as $BWT = \frac{2}{N(N-1)}\sum_{i=2}^{N}\sum_{j=1}^{i-1}(U_{i,j} - U_{j,j})$. A backward transfer in the interval $[-1, 0]$ characterizes forgetting: learning on a utility negatively interferes with the performances of the agent on previously encountered ones. On the other hand, when it lies in the interval $[0, 1]$, positive backward transfer happens, and the information obtained when learning on a utility improves the performance on previous ones.

Efficiency metrics. Díaz-Rodríguez et al. [7] propose additional metrics that evaluate the scalability of continual learning algorithms: model size efficiency, samples storage size efficiency, and computational efficiency. While these considerations are important, we omit these metrics from our experiments. Indeed, as our problem formulation (see 4.1) constrains considered methods from scaling with the number of utilities, they would invariably achieve a perfect score, making these metrics unsuitable for comparison.

Note that Lopez-Paz and Ranzato [16] mention that average accuracy (or utility) should be considered first, and the model with better BWT and FWT is to be picked in case of similar utility.

4.3.2 Extending the expected utility metric to non-linear utilities. The expected utility metric (EUM) is a widely used utility-based metric proposed by Zintgraf et al. [40] to evaluate multi-policy algorithms, which return a collection of policies called the solution set S. EUM consists in sampling utility functions from a probability

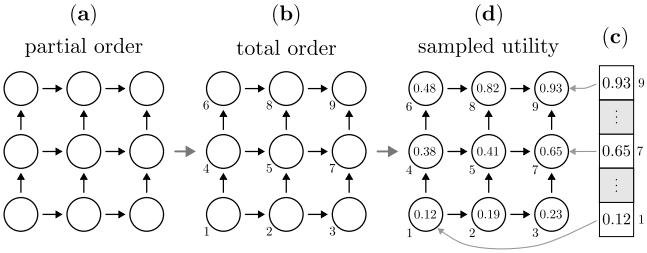

Figure 3: Illustration of the utility sampling process.

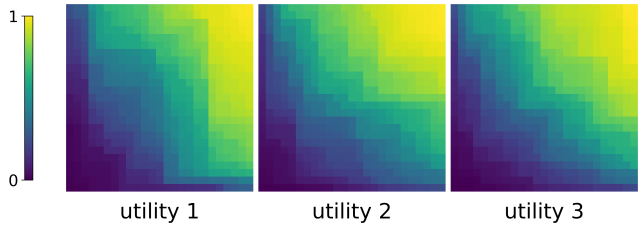

Figure 4: Utility sequence for evaluation on deep sea treasure.

distribution $P(u)$ over $\mathcal{U}'$, and measuring the expected average utility obtained with S. It is defined as

$$\text{EUM}(S, \mathcal{U}', P) = \mathbb{E}_{u\sim P(u)}[\max_{V^\pi \in S} u(V^\pi)].$$

As most work in multi-objective RL focuses on linear utilities, and since user preferences are rarely available, the EUM is usually computed using a uniform distribution over weight vectors. This makes the "default" assumption that all possible preferences are equally likely, and is used as a coverage metric similar to the hypervolume [33]. Note that, in practice, a uniform distribution over linear utilities does not necessarily produce a uniform distribution over behaviours, and it is not rare for clusters to form with large ranges of weights sharing the same optimal policy [17].

Defining such a metric for non-linear utilities is not trivial, as there is no standard basis to represent this space and as such no simple way to define a uniform distribution on it. We propose to use a similar scheme as the utility-representation tensor discussed in Section 4.2 to generate a random distribution over non-linear utilities. The full process is depicted in Figure 3 for a 2-dimensional utility using a resolution $r = 3$. First, notice that the monotonicity constraints on discretized utilities are a partial order, i.e.,

$$\forall k \in \{1, \dots, d\}, i_k \leq j_k \implies \tilde{\mathbf{U}}_{i_1,\dots,i_d} \leq \tilde{\mathbf{U}}_{j_1,\dots,j_d},$$

which can be represented as a directed acyclic graph with valued nodes (fig. 3a), i.e., an edge between two nodes implies that the target node has a value greater or equal to that of the source node. From this graph, we can obtain random total orders by sampling linear extensions (fig. 3b), using a random topological sorting algorithm (e.g., Kahn's algorithm [11]). Finally, we uniformly sample r^d values in $[0, 1]$ and sort them (fig. 3c), then assign them to the utility-representation tensor according to the sampled total order (fig. 3d). This gives us a way to sample random non-linear utilities with correctly enforced monotonicity constraints. We use this method to construct a space $\mathcal{U}'$ consisting of randomly sampled utility functions.

The algorithms we study do not generate solution sets due to the capacity restrictions discussed in Section 4.1. Therefore, this version of EUM can be used simply by evaluating the average test utility of the algorithms over the generated utility space. Remembering that $\pi_{i,j}$ represents the policy trained up to utility u_i and tested on utility u_j, we define this extension of the EUM to non-linear utilities as $\text{EUM}(\mathcal{U}', \pi_{i,\cdot}) = \mathbb{E}_{u_j\sim\mathcal{U}'}[u_j(\mathbf{G}^{\pi_{i,j}})]$. This formulation allows us to compute the EUM at any moment during training. In our experiments, we report both the average EUM across the training sequence, $\overline{\textbf{EUM}} = \frac{1}{N}\sum_{i=1}^{N} \text{EUM}(\mathcal{U}', \pi_{i,\cdot})$, and its final value, $\textbf{FEUM} = \text{EUM}(\mathcal{U}', \pi_{N,\cdot})$.

Note that Cai et al. [4] proposes a method to generate random non-linear utility functions, using multiple monotonic neural networks that learn to be as different to each other as possible, helping to augment the visual diversity of the obtained utilities. In practice, we noticed that their method was slower than our linear-extension based scheme, and both methods had similar PF coverage when tested on the deep sea treasure environment (see Section 5.2).

Although originally designed for evaluating multi-policy methods, **our adapted version of the EUM can be used in the continual non-linear MORL setting as an additional indicator of forward transfer**, given that the utilities encountered during training come from the same distribution. Indeed, while classical forward transfer (see sec. 4.3.1) only considers the utilities that will be seen in the sequence, our proposed extension of EUM theoretically takes into account the exact distribution of user preferences from which the sequence is sampled, **giving a more aligned measure of the expected performance for the user**. To that effect, we generate utility sequences for the experiments of Section 5 using this linear-extension scheme, such that the EUM is representative of the true underlying distribution.

5 EXPERIMENTS

In this section, we evaluate our proposed UBSM approach. We first describe the experimental setup, including the baselines and environments used, and then present an analysis of the results, highlighting the strengths of UBSM and noting potential points of misalignment in the formulation of classic CL metrics.

5.1 Baselines

We selected expected utility policy gradient [25, EUPG] for our approach, extending it to adapt to the continual MORL setting. EUPG is one of the few MORL methods specifically designed to optimize ESR for non-linear utility functions. It modifies the standard Monte Carlo policy gradient algorithm, optimizing directly on the utility of each trajectory, and conditioning the policy with the accrued return vector and the current timestep to introduce non-stationarity. As such, EUPG's loss function is defined as

$$\mathcal{L}(\pi) = -\sum_t u(\mathbf{R}_\tau^- + \mathbf{R}_\tau^+) \log \pi(a|s, \mathbf{R}_\tau^-, t),$$

with $\mathbf{R}_\tau^-$ and $\mathbf{R}_\tau^+$ the accumulated reward until and from timestep τ.

Our first baseline therefore consists in using **EUPG** in its original form applied to our setting with dynamic utilities, testing its

ability to evolve when new utility functions are met. Since EUPG was not designed for the continual setting, we explore using it in combination with elastic weight consolidation [19] and context-dependent gating [19] (baselines respectively denoted **EUPG+EWC** and **EUPG+XdG**), existing continual methods described in Section 2. Finally, our utility-based soft masking approach is combined with EUPG (**EUPG+UBSM**), modulating the weights of the policy network using our utility-representation. In the experiments, all methods are initialized at the start of the sequence, and learn continually throughout it.

5.2 Environments

Deep sea treasure (DST) is a classic MORL environment, presented in its original form by Vamplew et al. [33]. The agent controls a submarine in a grid-world representing a sea scattered with treasures, and learns to balance between 2 objectives ($d = 2$): the value of the gathered treasure with the time spent for reaching it.

Four room (4R) [3] is a grid-world environment consisting of four rooms with three types of shapes (representing the objectives thus $d = 3$) distributed within them. The goal of the agent is to navigate from a start state to a goal state, while collecting shapes along the way to maximize the utility.

Minecart (MC) is a complex environment proposed by Abels et al. [2], with discrete actions and continuous observations. We use the deterministic variant of the environment, in which the agent travels through a square world, collecting resources in mines before returning to the base to sell them. The 3-dimensional objective space is defined by the sales of the two ores collected alongside the minimization of fuel consumption.

Fruit tree navigation (FTN) [39] is an environment modeled by a binary tree with fruits on its leaves. Each fruit gives the agent a 6-dimensional reward vector ($d = 6$) representing its nutritional qualities (proteins, carbs...). The agent chooses between left and right at each non-terminal node, and receives its reward vector when reaching a fruit. In our experiments, we use the most complex version of this environment available with a depth of 7.

5.3 Results

For each experiment, we use the utility generation technique described in Section 4.3.2 to construct a sequence of utilities for the agent to learn on. We use a sequence of 3 utilities for DST (depicted in Figure 4, with each objective plotted on its own axis), in order to validate the method and display legible train-test utility matrices. We use sequences of 20 utilities for both 4R and FTN in order to simulate an extended learning period. To compute the EUM, we use the same method to generate a large set of utilities (30 for DST, 100 for 4R, MC, and FTN). Note that the utilities generated both for the learning sequence and for EUM are the same for all evaluated methods, ensuring fair comparison.

5.3.1 Deep sea treasure. We first validate our method using the simple and well-established DST environment. Due to the sparsity of its treasure reward component, we have found that adding a small entropy regularization term to our policies allowed the agents to consistently achieve better performances, regardless of the method. As such, results presented below integrate it in the tested agents.

Table 2: Aggregated results on all environments, with best results highlighted in bold (higher is better).

Env.	Model	$\overline{\text{EUM}}$	FEUM	FAU	RAU	FWT	BWT
(a) DST	EUPG	0.353	0.420	0.340	0.393	0.325	0.026
	+UBSM	**0.382**	**0.460**	**0.356**	**0.411**	**0.330**	**0.041**
(b) 4R	EUPG	0.534	0.514	0.519	0.539	0.532	0.020
	+EWC	0.601	0.584	0.598	0.593	0.608	-0.002
	+XdG	0.611	0.638	0.584	0.641	0.578	-0.034
	+UBSM	**0.642**	**0.768**	**0.635**	**0.677**	**0.612**	**0.030**
(c) MC	EUPG	0.831	0.832	0.757	0.830	0.832	0.000
	+EWC	0.770	0.781	0.667	0.772	0.768	0.008
	+XdG	0.828	0.839	0.754	0.831	0.824	0.006
	+UBSM	**0.844**	**0.871**	**0.775**	**0.854**	**0.833**	**0.030**
(d) FTN	EUPG	0.269	0.269	0.140	0.277	0.260	**0.000**
	+EWC	**0.378**	**0.380**	**0.351**	**0.385**	**0.368**	**0.000**
	+XdG	0.269	0.265	0.152	0.274	0.263	-0.003
	+UBSM	0.270	0.270	0.129	0.280	0.260	**0.000**

Table 3: Train-test utility matrices on deep sea treasure.

(a) EUPG

U	Te_1	Te_2	Te_3
Tr_1	0.41	0.33	0.34
Tr_2	0.40	0.33	0.30
Tr_3	0.46	0.38	0.38

(b) EUPG+UBSM

U	Te_1	Te_2	Te_3
Tr_1	0.42	0.34	0.33
Tr_2	0.40	0.34	0.32
Tr_3	0.49	0.40	0.41

As shown in Table 2 (a), EUPG+UBSM holds a small advantage over EUPG across all metrics. We can interpret this difference more finely by looking at the values used to compute these metrics in the train-test utility matrices for both methods (cf. Table 3). Looking row by row, we see that EUPG+UBSM starts without any specific advantage over EUPG, and is even slightly worse when testing u_3 after training on u_1. As the training continues, the improvement caused by adding UBSM to EUPG increases, showing its capacity to specialize for each utility.

5.3.2 Four room. The results in Table 2 (b) lead to several observations. First, all methods have similar performances on backward transfer, with values close to zero, characterizing low forgetting. Adding EWC or XdG to EUPG improves all metrics by a substantial margin except for BWT. EUPG+UBSM performs best on all metrics, and it notably achieves the highest scores on both average and final EUM, meaning it would perform best across the entire unseen preference range of the simulated user. Additionally, UBSM attains a large relative improvement between final and average EUM of nearly 20%, while XdG's relative improvement sits at around 4.5% and the EUM performance of EUPG and EWC actually decrease with improvements of around -4% and -3%, respectively.

This can be visually confirmed by looking at the plot of U^* throughout the learning sequence of 20 utilities in Figure 5, i.e., the performances on the current utility during training. While both EUPG and EUPG+EWC seem to slightly improve or stagnate, EUPG+XdG and EUPG+UBSM show an improvement trend over

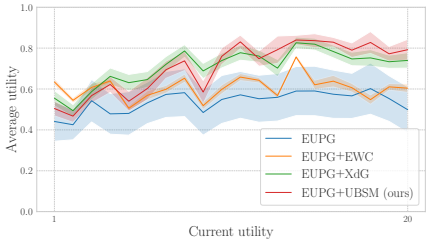

Figure 5: Evolution of U^* on four room.

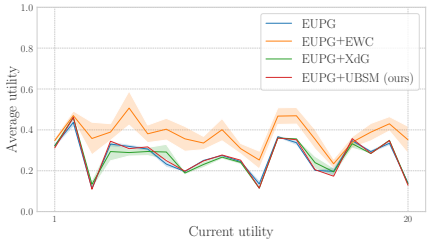

Figure 6: Evolution of U^* on FTN.

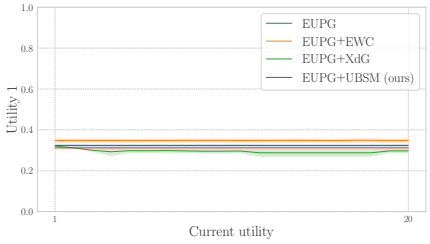

Figure 7: Evolution of $U_{\cdot,1}$ on FTN.

time, suggesting that positive forward transfer is occurring, with past utilities being used to kickstart performance on new ones.

This is a key result, highlighting the capacity of UBSM to significantly increase transfer when added to a baseline, resulting in a method able to continuously improve for the end user.

5.3.3 Minecart. EUPG+UBSM outperforms other methods in the Minecart environment, and, while smaller than on 4R, still reaches the highest relative EUM improvement. Adding other continual baselines to EUPG has little effect (XdG) and can even be detrimental (EWC). We attribute this to the higher complexity of Minecart's state and reward spaces. This complexity likely results in a smaller intersection of optimal parameters accross utilities, making the regularization of EWC overly restrictive, and leads XdG's subnetworks to struggle to re-learn important features for each utility.

5.4 Perspectives on the Evaluation of CMORL

In this section, we use the results from the fruit tree navigation environment as a case study to question the state of the evaluation of continual learning, and by extension that of CMORL.

Table 2 (d) seems to show that adding EWC to the EUPG baseline is the best way among those tested to induce a better continual behavior for the agent. While EUPG+EWC does outperform other methods for all metrics on FTN, all methods generate agents with similar behaviors. Figure 6 shows that EUPG, XdG and UBSM are almost indistinguishable when looking at average current utility, and that EUPG+EWC seems to follow a similar pattern with higher values. In Figure 7, we display the evolution of the first encountered utility throughout training. It highlights that all agents get stuck in an optimum for utility u_1 and do not deviate from it, with EUPG+EWC reaching a slightly better one than others, explaining the slight improvement in performance across all utilities.

We attribute these results to two main reasons. First, FTN is a complex, non-ergodic environment, with little correlation between actions taken and final reward vector. Coupled with coarse utility functions, this makes for a challenging domain and further complexifies exploration. Second, the better leaf reached by EWC seems not to be caused as much by the method but by the addition of a regularization term that helps exploration. In fact, we noticed similar improvements when adding entropy or L2 regularization, which could help delaying early collapse. Additionally, adding EWC to EUPG+UBSM also improved results.

Importantly, these degenerate behavior were not obviously reflected on the reported metrics. A backward transfer of 0.0 could hint at such stuck policies, yet it was very close to 0 in previous experiments with agents that still exhibited learning. Another signal is the difference between final and average EUM, which is close to 0 for all methods on FTN. When this difference is high, the agent tends to showcase an ability to transfer all experience to unseen utilities. When it is low, the agent may perform well on the encountered utilities but could overfit them, losing plasticity and worsening performances on unseen utilities. When close to zero, it means that the performance on unseen utilities is not affected by the learning process, strongly hinting at collapsed policy. Additionally, the forward transfer is positive for all methods despite them being stuck. This is a sign of misalignment between the behavior a continual agent capable of transferring knowledge is expected to have, and the result of this metric, highlighting the need for better, specialized way of evaluating continual learning. Finally, note the relative importance that each utility has on metrics computation. For RAU, the weight associated with utility u_i is $\frac{2(n-i+1)}{n(n+1)}$, meaning that that the earlier a utility is in the learning sequence (lower index i), the larger its contribution on the final values will be. As such, agents stuck on an optimum for the first few utilities may end up with artificially inflated RAU results.

We emphasize again the importance of evaluation in continual learning, and the need for the introduction of varied *auxiliary metrics* [27] to improve both the robustness of benchmarking and the understanding of the studied agent's behaviors.

6 CONCLUSION

In this paper, we explore the setting of continual multi-objective reinforcement learning (CMORL) with non-linear utility functions, which is still absent from the literature despite its relevance to real world problems. We introduce utility-based soft masking (UBSM), a framework designed to extend MORL methods to this dynamic setting. Our experiments demonstrate that UBSM performs better than compared baselines on classic MORL benchmarks, and highlight shortcomings of existing continual learning metrics in specific cases. While this work provides a first step into CMORL, there is ample room for future research in the area. Immediate next steps include the incorporation of domain knowledge – such as specific classes of parameterized utility functions – to improve learning efficiency and extending UBSM to a multi-policy setting to evaluate its zero-shot capabilities when an initial learning phase is permitted.

ACKNOWLEDGMENTS

Work funded by project ACCELER-AI (ANR-22-CE23-0028-01).

REFERENCES

[1] David Abel, André Barreto, Benjamin Van Roy, Doina Precup, Hado P van Hasselt, and Satinder Singh. 2023. A definition of continual reinforcement learning. *Advances in Neural Information Processing Systems* 36 (2023), 50377–50407.

[2] Axel Abels, Diederik Roijers, Tom Lenaerts, Ann Nowé, and Denis Steckelmacher. 2019. Dynamic Weights in Multi-Objective Deep Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 11–20.

[3] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. 2017. Successor features for transfer in reinforcement learning. *Advances in neural information processing systems* 30 (2017).

[4] Xin-Qiang Cai, Pushi Zhang, Li Zhao, Jiang Bian, Masashi Sugiyama, and Ashley Llorens. 2023. Distributional Pareto-Optimal Multi-Objective Reinforcement Learning. *Advances in Neural Information Processing Systems* 36 (Dec. 2023), 15593–15613.

[5] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. 2021. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence* 44, 7 (2021), 3366–3385.

[6] Timon Deschamps, Rémy Chaput, and Laëtitia Matignon. 2024. Multi-Objective Reinforcement Learning: An Ethical Perspective. (2024).

[7] Natalia Díaz-Rodríguez, Vincenzo Lomonaco, David Filliat, and Davide Maltoni. 2018. Don't forget, there is more than forgetting: new metrics for continual learning. *arXiv preprint arXiv:1810.13166* (2018).

[8] 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), 26. https://doi.org/10.1007/s10458-022-09552-y

[9] 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 (2023), 26.

[10] Nicolas Jozefowiez, Frédéric Semet, and El-Ghazali Talbi. 2008. Multi-objective vehicle routing problems. *European journal of operational research* 189, 2 (2008), 293–309.

[11] Arthur B Kahn. 1962. Topological sorting of large networks. *Commun. ACM* 5, 11 (1962), 558–562.

[12] Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. 2022. Towards continual reinforcement learning: A review and perspectives. *Journal of Artificial Intelligence Research* 75 (2022), 1401–1476.

[13] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2016. Overcoming Catastrophic Forgetting in Neural Networks. In *PNAS*. https://doi.org/10.1073/pnas.1611835114

[14] Tatsuya Konishi, Mori Kurokawa, Chihiro Ono, Zixuan Ke, Gyuhak Kim, and Bing Liu. 2023. Parameter-Level Soft-Masking for Continual Learning. In *Proceedings of the 40th International Conference on Machine Learning*. PMLR, 17492–17505.

[15] Lihe Li, Ruotong Chen, Ziqian Zhang, Zhichao Wu, Yi-Chen Li, Cong Guan, Yang Yu, and Lei Yuan. 2024. Continual Multi-Objective Reinforcement Learning via Reward Model Rehearsal. *IJCAI* (2024).

[16] David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. *Advances in neural information processing systems* 30 (2017).

[17] Haoye Lu, Daniel Herman, and Yaoliang Yu. 2022. Multi-Objective Reinforcement Learning: Convexity, Stationarity and Pareto Optimality. In *The Eleventh International Conference on Learning Representations*.

[18] Arun Mallya and Svetlana Lazebnik. 2018. PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7765–7773.

[19] Nicolas Y. Masse, Gregory D. Grant, and David J. Freedman. 2018. Alleviating Catastrophic Forgetting Using Context-Dependent Gating and Synaptic Stabilization. *Proceedings of the National Academy of Sciences* 115, 44 (Oct. 2018), E10467–E10475. https://doi.org/10.1073/pnas.1803839115

[20] Kristof Van Moffaert and Ann Nowé. 2014. Multi-Objective Reinforcement Learning Using Sets of Pareto Dominating Policies. *Journal of Machine Learning Research* 15, 107 (2014), 3663–3692.

[21] Sriraam Natarajan and Prasad Tadepalli. 2005. Dynamic preferences in multi-criteria reinforcement learning. In *Proceedings of the 22nd international conference on Machine learning*. 601–608.

[22] Patrice Perny and Paul Weng. 2010. On Finding Compromise Solutions in Multi-objective Markov Decision Processes.. In *ECAI*, Vol. 215. 969–970.

[23] Mathieu Reymond, Eugenio Bargiacchi, and Ann Nowé. 2022. Pareto Conditioned Networks. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. 1110–1118.

[24] Mathieu Reymond, Conor F. Hayes, Denis Steckelmacher, Diederik M. Roijers, and Ann Nowé. 2023. Actor-Critic Multi-Objective Reinforcement Learning for Non-Linear Utility Functions. *Autonomous Agents and Multi-Agent Systems* 37, 2 (Oct. 2023), 23. https://doi.org/10.1007/s10458-023-09604-x

[25] Diederik Roijers, Denis Steckelmacher, and Ann Nowe. 2018. Multi-Objective Reinforcement Learning for the Expected Utility of the Return.

[26] Diederik Marijn Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. 2013. A Survey of Multi-Objective Sequential Decision-Making. *Journal of Artificial Intelligence Research* 48 (Oct. 2013).

[27] Tom Schaul, Hado van Hasselt, Joseph Modayil, Martha White, Adam White, Pierre-Luc Bacon, Jean Harb, Shibl Mourad, Marc Bellemare, and Doina Precup. 2018. The barbados 2018 list of open issues in continual learning. *arXiv preprint arXiv:1811.07004* (2018).

[28] Hamed Shakouri and Aliyeh Kazemi. 2017. Multi-objective cost-load optimization for demand side management of a residential area in smart grids. *Sustainable cities and society* 32 (2017), 171–180.

[29] Umer Siddique, Paul Weng, and Matthieu Zimmer. 2020. Learning fair policies in multi-objective (deep) reinforcement learning with average and discounted rewards. In *International Conference on Machine Learning*. PMLR, 8905–8915.

[30] Umer Siddique, Paul Weng, and Matthieu Zimmer. 2020. Learning Fair Policies in Multi-Objective (Deep) Reinforcement Learning with Average and Discounted Rewards. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 8905–8915.

[31] Joar Skalse, Lewis Hammond, Charlie Griffin, and Alessandro Abate. [n.d.]. Lexicographic Multi-Objective Reinforcement Learning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*.

[32] Peter Vamplew, Richard Dazeley, Ewan Barker, and Andrei Kelarev. 2009. Constructing Stochastic Mixture Policies for Episodic Multiobjective Reinforcement Learning Tasks. In *AI 2009: Advances in Artificial Intelligence*, David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Ann Nicholson, and Xiaodong Li (Eds.). Vol. 5866. Springer Berlin Heidelberg, Berlin, Heidelberg, 340–349. https://doi.org/10.1007/978-3-642-10439-8_35

[33] Peter Vamplew, Richard Dazeley, Adam Berry, Rustam Issabekov, and Evan Dekker. 2011. Empirical Evaluation Methods for Multiobjective Reinforcement Learning Algorithms. *Machine Learning* 84, 1 (July 2011), 51–80. https://doi.org/10.1007/s10994-010-5232-5

[34] Peter Vamplew, Richard Dazeley, Cameron Foale, Sally Firmin, and Jane Mummery. 2018. Human-Aligned Artificial Intelligence Is a Multiobjective Problem. *Ethics and Information Technology* 20, 1 (March 2018), 27–40. https://doi.org/10.1007/s10676-017-9440-6

[35] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. 2024. A comprehensive survey of continual learning: Theory, method and application. *IEEE transactions on pattern analysis and machine intelligence* 46, 8 (2024), 5362–5383.

[36] D. J White. 1982. Multi-Objective Infinite-Horizon Discounted Markov Decision Processes. *J. Math. Anal. Appl.* 89, 2 (Oct. 1982), 639–647. https://doi.org/10.1016/0022-247X(82)90122-6

[37] Maciej Wołczyk, Michał Zając, Razvan Pascanu, Łukasz Kuciński, and Piotr Miłoś. 2021. Continual world: A robotic benchmark for continual reinforcement learning. *Advances in Neural Information Processing Systems* 34 (2021), 28496–28510.

[38] Kyle Wray, Shlomo Zilberstein, and Abdel-Illah Mouaddib. 2015. Multi-objective MDPs with conditional lexicographic reward preferences. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 29.

[39] Runzhe Yang, Xingyuan Sun, and Karthik Narasimhan. 2019. A Generalized Algorithm for Multi-Objective Reinforcement Learning and Policy Adaptation. In *Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc.

[40] Luisa M Zintgraf, Timon V Kanters, Diederik M Roijers, Frans Oliehoek, and Philipp Beau. 2015. Quality assessment of MORL algorithms: A utility-based approach. In *Benelearn 2015: proceedings of the 24th annual machine learning conference of Belgium and the Netherlands*.