Journal of Artificial Intelligence Research 79 (2024) 679-723 Submitted 11/2023; published 02/2024

Multi-Objective Reinforcement Learning Based on Decomposition: A Taxonomy and Framework

Florian Felten FLORIAN.FELTEN@UNI.LU
SnT, University of Luxembourg

El-Ghazali Talbi EL-GHAZALI.TALBI@UNIV-LILLE.FR
CNRS/CRIStAL, University of Lille
FSTM/DCS, University of Luxembourg

Grégoire Danoy GREGOIRE.DANOY@UNI.LU
FSTM/DCS, University of Luxembourg
SnT, University of Luxembourg

Abstract

Multi-objective reinforcement learning (MORL) extends traditional RL by seeking policies making different compromises among conflicting objectives. The recent surge of interest in MORL has led to diverse studies and solving methods, often drawing from existing knowledge in multi-objective optimization based on decomposition (MOO/D). Yet, a clear categorization based on both RL and MOO/D is lacking in the existing literature. Consequently, MORL researchers face difficulties when trying to classify contributions within a broader context due to the absence of a standardized taxonomy. To tackle such an issue, this paper introduces multi-objective reinforcement learning based on decomposition (MORL/D), a novel methodology bridging the literature of RL and MOO. A comprehensive taxonomy for MORL/D is presented, providing a structured foundation for categorizing existing and potential MORL works. The introduced taxonomy is then used to scrutinize MORL research, enhancing clarity and conciseness through well-defined categorization. Moreover, a flexible framework derived from the taxonomy is introduced. This framework accommodates diverse instantiations using tools from both RL and MOO/D. Its versatility is demonstrated by implementing it in different configurations and assessing it on contrasting benchmark problems. Results indicate MORL/D instantiations achieve comparable performance to current state-of-the-art approaches on the studied problems. By presenting the taxonomy and framework, this paper offers a comprehensive perspective and a unified vocabulary for MORL. This not only facilitates the identification of algorithmic contributions but also lays the groundwork for novel research avenues in MORL.

1. Introduction

In the complex landscape of real-life decision-making, individuals often find themselves facing the challenge of balancing conflicting objectives. Consider the case of daily commuting, where one must select a mode of transportation. Each option comes with its own set of trade-offs, including factors like time, cost, and environmental impact. These choices are influenced by personal values and preferences. Multi-objective methods are designed to enhance the process of decision-making in such scenarios. When a user's preferences are known in advance (*a priori*), the solution typically results in a single optimal choice. However, in cases where user preferences are uncertain or undefined, the search for the best

©2024 The Authors. Published by AI Access Foundation under Creative Commons Attribution License CC BY 4.0.

decision leads to a set of optimal solutions, and users are asked to make informed choices afterward (*a posteriori*).

Multi-objective optimization (MOO) represents a well-established field dedicated to solving such problems efficiently. Traditionally, evolutionary algorithms (EAs) have been the go-to tools for searching for solutions in MOO. Moreover, recent years have witnessed the emergence of decomposition-based methods, in which the multi-objective problem (MOP) is transformed into a set of single-objective problems (SOPs) through scalarization functions. Building on seminal work in decomposition such as MOEA/D (Zhang & Li, 2007), a significant body of research has emerged in this area. While MOO has found successful applications in numerous problem domains, there are scenarios where the quest for solutions involves navigating a decision space that is simply too vast to explore with traditional techniques.

Notably, reinforcement learning (RL) has recently been the subject of significant research interest due to its successes in a variety of applications, which are known to be challenging for search-based methods (Mnih et al., 2015; Silver et al., 2016; Wurman et al., 2022). However, this field of research is limited to agents aiming at maximizing a single objective. Thus, it has recently been expanded into more demanding scenarios, such as training the agent to learn to make compromises between multiple conflicting objectives in multi-objective RL (MORL). Given the conceptual proximity to RL and MOO, MORL researchers have explored the integration of ideas from these well-established fields to form new contributions in MORL. Although some survey papers have offered an overview of MORL's current state (Roijers et al., 2013; Hayes et al., 2022), these surveys have primarily delved into solution concepts and theory, without comprehensively studying recent solving methods. In particular, to the best of our knowledge, no existing work has comprehensively analyzed the interactions between RL, MORL, and MOO, or systematically identified recurring patterns and approaches employed in MORL algorithms.

Therefore, the initial portion of this work strives to clarify the similarities and distinctions between RL, MORL, and MOO, especially in scenarios where user preferences are uncertain (*a posteriori*). Specifically, we aim to show that there are ways to classify and describe MORL contributions within a broader context. Hence, we present a taxonomy that aims at classifying existing and future MORL works, drawing from RL and MOO concepts. Subsequently, this taxonomy is applied to categorize existing MORL research, exemplifying its utility in comprehending the state of the art and distilling key contributions from various papers.

Building upon the theoretical foundation laid out in the taxonomy, we introduce the multi-objective reinforcement learning based on decomposition (MORL/D) framework. This modular framework can be instantiated in diverse ways using tools from both RL and MOO. To demonstrate its flexibility, we implement different variations of the framework and evaluate its performance on well-established benchmark problems. The experiments showcase how different instantiations of the framework can yield diverse results. By presenting the taxonomy and framework, our aim is to offer perspective and a shared lexicon for experts from various fields while paving the way for fresh avenues of research in MORL.

The structure of the paper unfolds as follows: Sections 2 and 3 provide background information on RL and MOO, respectively. Section 4 introduces the taxonomy and the MORL/D framework, while Section 5 illustrates typical usages of the introduced taxonomy.

In Section 6, we demonstrate various implementations of MORL/D on benchmark problems, and the final sections encompass discussions on future directions and conclusions.

2. Reinforcement Learning

In the framework of RL, an agent interacts with an environment as follows: when faced with a particular state, the agent selects an action, which, upon execution, alters the environment, and in return, the agent receives a reward that indicates the quality of the action taken. In this context, the primary objective of the agent is to acquire a policy function, typically denoted by π, that optimally guides its action choices to maximize the cumulative rewards obtained during its interactions with the environment. After the training period, this learned policy dictates the agent's decision-making process when it encounters different states.

Formally, an RL problem is modeled by a Markov decision process (MDP), which is defined by a tuple $(\mathbb{S}, \mathbb{A}, r, p, \mu_0)$ where $\mathbb{S}$ are the states the agent can perceive, $\mathbb{A}$ are the actions the agent can undertake to interact with the environment, $r : \mathbb{S} \times \mathbb{A} \times \mathbb{S} \to \mathbb{R}$ is the reward function, $p : \mathbb{S} \times \mathbb{A} \times \mathbb{S} \to [0, 1]$ is the probability of transition function, giving the probability of the next state given the current state and action, and μ_0 is the distribution over initial states s_0 (Sutton & Barto, 2018).

Within this framework, a policy $\pi : \mathbb{S} \times \mathbb{A} \to [0, 1]$ can be assigned to a numerical value after evaluation in the MDP. This evaluation value is formally referred to as the value function and can be expressed as the discounted sum of rewards collected over an infinite time horizon (or until the end of an episode) starting from a given state s:

$$v^\pi(s) \equiv \mathbb{E}_{a_t \sim \pi(s_t)} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) |\ s_t = s \right], \tag{1}$$

where t represents the timesteps at which the agent makes a choice, and $0 \leq \gamma < 1$ is the discount factor, specifying the relative importance of long-term rewards with respect to short-term rewards. Furthermore, the overall value of a policy π, regardless of the initial state, can be expressed as $v^\pi \equiv \mathbb{E}_{s_0 \sim \mu_0} v^\pi(s_0)$. Such a value is traditionally used to define an ordering on policies, *i.e.* $\pi \succeq \pi' \iff v^\pi \geq v^{\pi'}$. The goal of an RL algorithm is then to find an optimal policy π^*, defined as one which maximizes v^π: $\pi^* = \text{argmax}_\pi\, v^\pi$.

To find such an optimal policy, many RL algorithms have been published over the last decades. A high-level skeleton of the RL process is presented in Algorithm 1. The algorithm first initializes its policy and an experience buffer (lines 1–2). Then, it samples experience tuples (s_t, a_t, r_t, s_{t+1}) from the environment by using the current policy and stores those in the buffer (lines 4–5). From the buffered experiences and its current value, the policy is improved (lines 6–7). The optimization process stops when a criterion specified by the user is met and the current policy is returned.

Each part of the algorithm can be instantiated in various ways, constituting the design choices of RL, as illustrated in Figure 1. The following sections discuss the role of each part and give a few examples of possible instantiations.

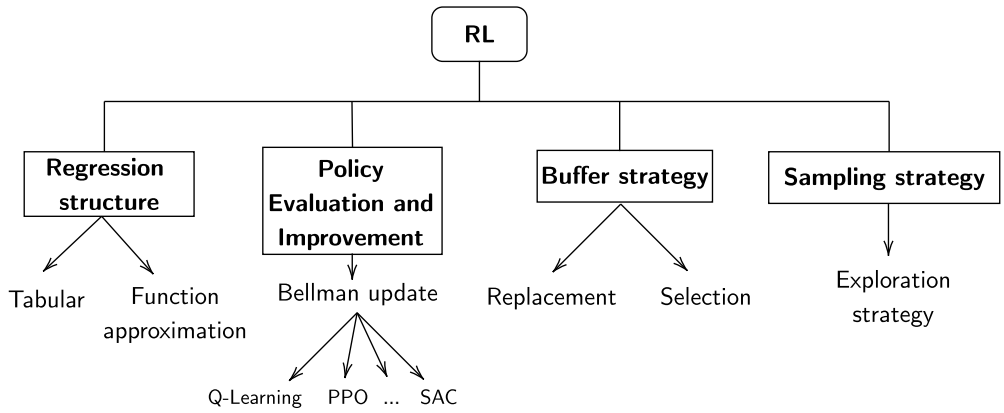


Figure 1: Reinforcement learning: design choices

Algorithm 1 Reinforcement learning process.

Input: Stopping criterion *stop*, Environment *env*.
Output: An optimized policy π.

1: $\pi = \mathit{Initialize}()$
2: $\mathbb{B} = \emptyset$
3: **while** $\neg \mathit{stop}$ **do**
4: $\quad \mathit{experiences} = \mathit{Sample}(\mathit{env}, \pi)$
5: $\quad \mathbb{B} = \mathit{UpdateBuffer}(\mathbb{B}, \mathit{experiences})$
6: $\quad \tilde{v}^{\pi} = \mathit{EvaluatePolicy}(\pi)$
7: $\quad \pi = \mathit{ImprovePolicy}(\pi, \tilde{v}^{\pi}, \mathbb{B})$
8: **end while**
9: **return** π

2.1 Regression Structure

A cornerstone design point in RL regards how to encode the policy function. Early RL algorithms often represented the policy using a tabular format. For example, Q-Learning (Watkins & Dayan, 1992) stores the action-value estimates, $\tilde{q}(s, a)$ in such a table. From this structure, a policy can be derived by selecting the action with the highest q-value from each state, *i.e.* $\pi(s, a) = \text{argmax}_{a \in \mathbb{A}} \tilde{q}(s, a)$. This is known as a greedy deterministic policy, as it always chooses the action with the largest expected reward.

However, this kind of approach does not scale to highly dimensional problems, *e.g.* with continuous states or actions. Hence, recent algorithms use function approximation based on regression, such as trees or deep neural networks (DNNs) (Mnih et al., 2015). In such settings, with the policy π_θ being parameterized by a set of parameters $\theta \in \Theta$ (Θ being the parameter space), the RL problem boils down to finding an optimal assignment of parameters $\theta^* = \text{argmax}_{\theta \in \Theta} v^{\pi_\theta}$. Finally, recent algorithms often rely on the use of multiple regression structures. For example, in actor-critic settings, one structure, the actor, aims at representing the probability of taking an action (*i.e.* the policy), while another structure, the critic, estimates the state(-action) values (Sutton & Barto, 2018).

2.2 Policy Evaluation and Improvement

RL algorithms usually rely on estimated values to bootstrap their policy improvement process (Sutton & Barto, 2018). For example, from an experience tuple (s_t, a_t, r_t, s_{t+1}) sampled from the environment and its current estimations $\tilde{q}(s, a)$, the well-known Q-Learning algorithm (Watkins & Dayan, 1992) updates its estimations using the following relation:

$$\tilde{q}(s_t, a_t) \leftarrow \tilde{q}(s_t, a_t) + \alpha \left(r_t + \gamma \max_{a' \in \mathbb{A}} \tilde{q}(s_{t+1}, a') - \tilde{q}(s_t, a_t) \right) \tag{2}$$

where $\alpha \in [0, 1]$ is the learning rate, and the term $r_t + \gamma \max_{a' \in \mathbb{A}} \tilde{q}(s_{t+1}, a') - \tilde{q}(s_t, a_t)$ is called temporal difference error (TD-error). Many other update relations have been published over the years: policy gradient methods such as REINFORCE (Sutton & Barto, 2018), or actor-critic approaches such as PPO (Schulman et al., 2017) and SAC (Haarnoja et al., 2018).

2.3 Buffer Strategy

In recent algorithms, policy updates are often batched using an experience buffer. This allows learning multiple times from an experience by replaying it, but also speeding up the policy improvement steps by performing mini-batch gradient descent in deep RL. Two choices linked to buffers arise: deciding which experiences in the buffer will be replaced by new ones, and which experiences to select to update the policy.

Replacement. While the simplest replacement criterion is based on recency, more elaborate techniques have also been studied. For example, de Bruin et al. (2016) propose using two experience buffers: one that is close to the current policy (recency criterion), while another one keeps the stored experiences close to a uniform distribution over the state-action space. This allows the computation of more robust policies and reduces the need for continuous, thorough exploration.

Selection. The most straightforward method for selecting experiences from the buffer is to choose them uniformly. However, research has shown in various articles that more intelligent experience selection strategies can significantly improve results in practice. For instance, prioritized sampling, as discussed in Schaul et al. (2016), is one such approach that has been shown to yield superior outcomes.

2.4 Sampling Strategy

The performance of an RL agent depends on which experiences were collected in the environment to learn its policy. Thus, the question of which action to choose in each state is crucial in such algorithms. To reach good performances, an agent must ideally sample (1) multiple times the same state-action pairs so as to compute good estimates of the environment dynamics (which can be stochastic), (2) regions leading to good rewards, to obtain good performances, and (3) unexplored regions, to find potentially better regions. The tension between points (2) and (3) is commonly referred to as the *exploration-exploitation dilemma*. Typically, the agent faces the dilemma of strictly adhering to its current policy (exploitation) and making exploratory moves guided by specific criteria. A widely used

approach in this scenario is known as ϵ-greedy, which introduces an element of randomness to facilitate exploration within the environment. Additionally, certain methods suggest building a model, such as a map, of the environment. This model provides the agent with a broader perspective, enabling more systematic exploration strategies. This concept is referred to as "state-based exploration" in the work of Moerland et al. (2023).

2.5 Summary

This section presented RL, which allows the automated learning of agent's behaviors. The whole learning process is based on reinforcement towards good signals, provided by the reward function. However, in MDPs, these reward functions are limited to scalar signals, which limits the applicability of such techniques. Indeed, real-world scenarios often involve making compromises between multiple objectives, as noted in Vamplew et al. (2022). To initiate the discussion towards more complex reward schemes, the following section presents the world of multi-objective optimization through the prism of decomposition techniques.

3. Multi-Objective Optimization Based on Decomposition

Multi-objective optimization aims at optimizing problems involving multiple conflicting objectives. In such settings, a solution x is evaluated using a m-dimensional vector function, where m is the number of objectives. Formally, the optimization problem can be expressed as $\max \vec{f}(x) = \max(f_1(x), ..., f_m(x))$ subject to $x \in \Omega$, where Ω is the decision space (*i.e.* search space), and $\vec{f}$ is the objective function.

Solution concepts. In cases where the decision maker (DM) has known preferences over objectives *a priori*, these problems can be simplified to single-objective problems by converting the objective values into scalars using a scalarization function $g(\vec{f}(x)) : \mathbb{R}^m \to \mathbb{R}$. However, many approaches and real-life scenarios cannot make this assumption and instead operate within the *a posteriori* setting, where the DM's preferences are not known in advance.

In the *a posteriori* setting, where the evaluation of each solution maintains a vector shape, algorithms commonly rely on the concept of Pareto dominance to establish an order among solutions. A solution x is said to Pareto dominate another solution x' if and only if it is strictly better for at least one objective, without being worse in any other objective. Formally:

$$x \succ_P x' \iff (\forall i : f_i(x) \geq f_i(x')) \wedge (\exists j : f_j(x) > f_j(x')).$$

This definition does not impose a total ordering on all solutions. For instance, it is possible that two solutions' evaluations, such as $(1, 0)$ and $(0, 1)$, may not dominate each other. These solutions are referred to as *Pareto optimal*, signifying that they are both potentially optimal solutions as long as the DM's preferences remain unknown. Consequently, the primary goal of the optimization process is to identify a collection of Pareto optimal solutions, referred to as the *Pareto set*. The evaluations associated with these solutions constitute the *Pareto front* (PF). Upon receiving a set of solutions, the DM can then make an informed choice, taking into account the trade-offs presented in the PF. Formally, a PF is defined as follows:

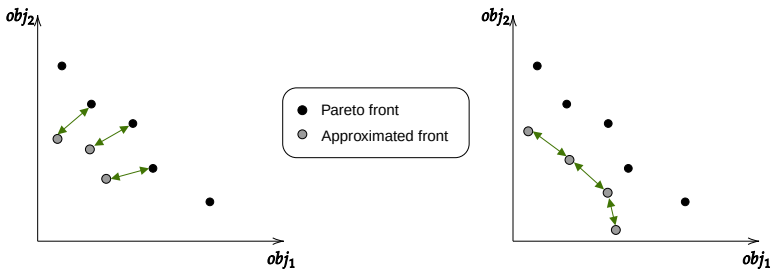


Figure 2: Illustration of Pareto front approximations. In the left part, the convergence aspect of the approximated front is represented by the arrows. In the right part, the diversity aspect is represented by the arrows.

$$\mathcal{F} \equiv \{\vec{f}(x) \mid \nexists\, x' \text{ s.t. } x' \succ_P x\}. \tag{3}$$

Approximated optimization methods. In most scenarios, real-world problems are often too hard to solve using exact methods. Hence, algorithms often provide an approximation of the Pareto set (and its corresponding front) by relying on metaheuristics. A good approximation of the PF is characterized by two criteria: (1) *convergence* with the true solution, to present solutions of good quality to the DM, and (2) *diversity*, to present a wide range of compromises to the DM. Examples of Pareto fronts showing the importance of both criteria are shown in Figure 2.

Single solution and population-based methods. The literature of *a posteriori* MOO is generally divided into two classes of algorithms: *single solution based* and *population-based* (Talbi, 2009). The first class maintains and improves a single solution at a time and loops through various preferences, trajectories, or constraints to discover multiple solutions on the PF, *e.g.* Czyżżak and Jaszkiewicz (1998), Hansen (2000). The second class maintains a set of solutions, called *population*, that are jointly improved over the search process, *e.g.* Zhang and Li (2007), Alaya, Solnon, and Ghedira (2007). Notably, this population-based approach is currently considered the state of the art for solving MOPs due to the performance of these algorithms. Consequently, this work will concentrate on population-based algorithms.

Decomposition. Multi-objective optimization based on decomposition (MOO/D) relies on the fundamental concept of dividing a MOP into several SOPs using a scalarization function. This function, represented by $g : \mathbb{R}^m \to \mathbb{R}$, employs associated weights represented by the vector $\vec{\lambda}$. This methodology, as depicted in Figure 3, facilitates the approximation of the PF by solving the SOPs corresponding to different weight vectors. Each of these weight vectors is designed to target specific regions of the PF, providing a comprehensive exploration of the objective space. Moreover, decomposition usually simplifies the problem and offers a simple way to parallelize the search process. This technique is applicable in both the context of *single-solution based* and *population-based* algorithms.

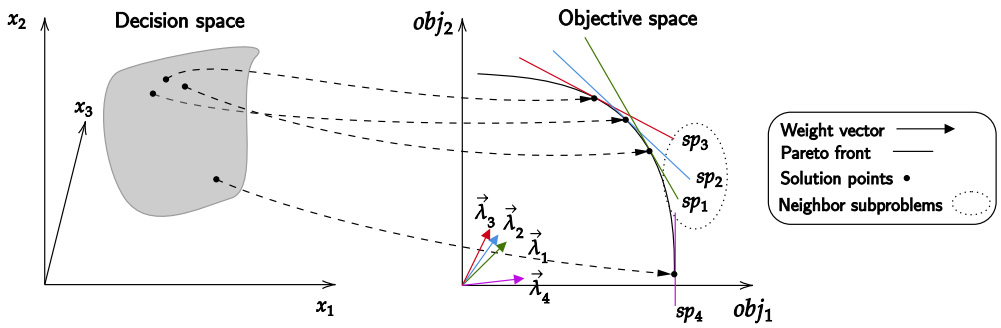


Figure 3: The decomposition in the objective space idea: split the multi-objective problem into various single-objective problems sp_n by relying on a scalarization function (weighted sum in this case). sp_1, sp_2, and sp_3 are considered to be neighbors since their associated weight vectors are close to each other while sp_4 is not considered to be in the neighborhood.

Algorithm 2 Population based MOO/D, high-level framework.

Input: Stopping criterion *stop*, Scalarization method g, Exchange trigger *exch*.
Output: The approximation of the Pareto set stored in the external archive population $\mathbb{EP}$.

$\mathbb{P}, \mathbb{W}, \mathbb{Z} = Initialize()$
$\mathbb{EP} = Prune(\mathbb{P})$
$\mathbb{N} = InitializeNeighborhood(\mathbb{P}, \mathbb{W})$
while $\neg stop$ **do**
  $individuals, \mathbb{W}', \mathbb{Z}' = Select(\mathbb{P}, \mathbb{W}, \mathbb{Z})$
  $candidates = Search(individuals, \mathbb{W}', \mathbb{Z}', g, exch)$
  $\mathbb{P} = \mathbb{P} \cup candidates$
  $evaluations = \forall c \in candidates : \vec{f}(c)$
  $\mathbb{EP} = Prune(\mathbb{EP} \cup candidates, evaluations)$
  $\mathbb{W}, \mathbb{Z} = Adapt(\mathbb{W}, \mathbb{Z}, \mathbb{EP})$
  $\mathbb{N} = UpdateNeighborhood(\mathbb{P}, \mathbb{N}, \mathbb{W})$
  $Cooperate(\mathbb{N})$
end while
return $\mathbb{EP}$

In the population-based setting, the assumption that neighboring subproblems share common solution components is often utilized, enabling these subproblems to cooperate by exchanging information. This cooperation typically improves the optimization process.

A generic framework based on such decomposition techniques can be found in Algorithm 2. MOO/D maintains a population of solutions $\mathbb{P}$, a set of weights $\mathbb{W}$, and reference points $\mathbb{Z}$ to apply in the scalarization function g. The best individuals are kept in an external archive population $\mathbb{EP}$ during the optimization process according to a criterion defined by the *Prune* function, *e.g.* Pareto dominance (Equation 3). At each iteration, a set of individuals from the population, along with weights ($\mathbb{W}' \subseteq \mathbb{W}$) and reference points

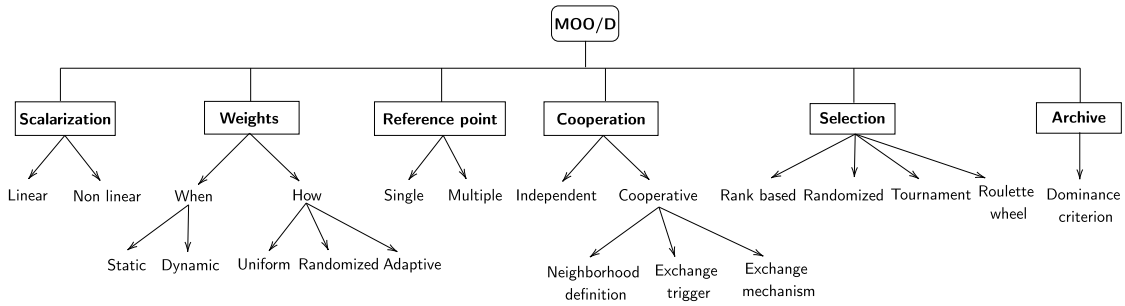


Figure 4: Design choices of multi-objective optimization based on decomposition (MOO/D).

($\mathbb{Z}' \subseteq \mathbb{Z}$) are selected as starting points to search for better solutions until an exchange criterion (*exch*) is triggered (lines 5–6). Then, the generated candidates are integrated into the population and archive based on their evaluated performance (lines 7–9). Next, the algorithm adapts the weights and reference points according to the current state of the PF (lines 10–11). Additionally, some knowledge is exchanged between subproblems in the same neighborhood (line 12). Finally, the algorithm returns the set of non-dominated solutions seen so far (line 14).

Over the years, many variations of the MOO/D framework have been proposed, and recurring patterns were identified, *e.g.* Figure 4 has been compiled from various sources (Talbi, 2009; Santiago et al., 2014; Xu et al., 2020b). The rest of this chapter explains each of the building blocks that can be instantiated in different ways in such a framework. Some illustrative examples and notable articles are also pointed out in the discussion. For further references, some surveys dedicated to decomposition techniques in MOO can be found in Santiago et al. (2014), Xu et al. (2020b).

3.1 Scalarization Functions

Various scalarization functions g have been the subject of studies over the last decades. These allow the MOP to be decomposed into multiple simpler SOPs. Moreover, scalarization functions are usually parameterized by weight vectors that target different areas of the objective space.

Linear scalarization. The most common and straightforward scalarization technique is the weighted sum: $g^{\text{ws}}(x) = \sum_{i=1}^{m} \lambda_i f_i(x) = \vec{\lambda}^{\intercal} \vec{f}(x)$. This method is easy to comprehend and enables the specification of weights as percentages to express preferences between the objectives. However, this approach has limitations. With this kind of simplification, the subproblems become linear, which means they cannot accurately capture points in the concave region of the PF (Marler & Arora, 2010).

Non-linear scalarization. In response to the limitations of linear scalarization, alternative non-linear techniques, such as the Chebyshev scalarization, have been investigated. This scalarization function, also known as the norm L^{∞}, is defined as the maximum weighted distance to a utopian reference point $\vec{z}$, expressed as $g^{\text{ch}}(x) = \max_{i \in [1,m]} |\lambda_i (f_i(x) - z_i)|$. Note that in this particular case, the goal is to minimize the scalarized values instead of maximizing like in the linear case. Using such non-linear techniques enables the identifica-

tion of points within the concave regions of the PF. See for example the work of Emmerich and Deutz (2018) for some visual examples. Intriguingly, some studies, such as Ishibuchi et al. (2010), have also proposed the combination or adaptation of various scalarization techniques to leverage their respective advantages.

3.2 Weights

Scalarization functions rely on weight vectors $\vec{\lambda} \in \mathbb{W}$ to target different points in the PF. Therefore, the way these weights are generated is crucial to obtain good solutions. There are two design choices for the weights: (1) when and (2) how to (re-)generate them.

When to generate weights? The simplest approach is to fix the weights before any search is started (**static**). However, the shape of the Pareto front, being unknown at that time, can be complex and may require adapting the weights during the search to focus on sparse areas, *i.e.* where the Pareto front is not well estimated. Thus, multiple ways to adapt the weights during the search have been published, *e.g.* Qi et al. (2013). This is referred to as **dynamic** weights in this work.

How to generate weights? Weights can be assigned through different approaches, such as **uniform** distribution across the objective space (Das & Dennis, 2000; Blank et al., 2021), **randomized** processes, or **adaptation** based on evolving knowledge during the search (Qi et al., 2013; Czyżżak & Jaszkiewicz, 1998). Weight adaptation strategies can vary, including focusing on underrepresented regions in the estimated PF or predicting potential improvements in the objective space (Dubois-Lacoste et al., 2011). Pareto simulated annealing (PSA) (Czyżżak & Jaszkiewicz, 1998), for instance, exemplifies this adaptation approach, as it adjusts an individual's weights in response to its current evaluation and proximity to non-dominated solutions. Formally, for an individual x and its nearest non-dominated neighbor x', PSA modifies the weights attached to the SOP leading to x using:

$$\lambda_j^x = \begin{cases} \delta\lambda_j^x & \text{if } f_j(x) \geq f_j(x') \\ \lambda_j^x/\delta & \text{if } f_j(x) < f_j(x'), \end{cases} \tag{4}$$

with δ being a constant close to 1, typically $\delta = 1.05$. This mechanism, encapsulated within the *Adapt* function of MOO/D, allows the fine-tuning of SOPs to excel at objectives where they already exhibit proficiency. This, in turn, encourages the SOPs to move away from their neighboring solutions, ultimately enhancing the diversity within the estimated PF.

3.3 Reference Points

Certain scalarization techniques, such as Chebyshev, depend on the selection of an optimistic or pessimistic point in the objective space to serve as a reference point $\vec{z}$. Similarly to weight vectors, the choice of these reference points has a significant influence on the final performance of the MOO algorithm. The careful determination of these reference points is crucial, as it profoundly impacts the algorithm's ability to approximate the Pareto front effectively. Hence, multiple settings for reference points exist.

Single reference point. A straightforward approach to setting the reference point is to fix it before commencing the optimization process (Zhang & Li, 2007). In this manner, the

reference point remains constant throughout the optimization, allowing for a controlled and deterministic approach.

Multiple reference points. Nevertheless, setting the reference point beforehand can sometimes result in suboptimal outcomes for the algorithm. For this reason, some studies suggest dynamic adaptation of the reference point during the search, as exemplified by the work of Liu et al. (2020). This adaptive approach enables the reference point to evolve and align with the changing characteristics of the Pareto front, potentially improving the quality of the results.

3.4 Cooperation

The most straightforward approach to solving a MOP using decomposition is to **independently** address all generated SOPs. This concept underpins the first single-solution decomposition-based algorithms, such as the one introduced by Czyżżak and Jaszkiewicz (1998). Subsequently, the idea of promoting **cooperation** among subproblems in close proximity (referred to as neighbors) emerged (Zhang & Li, 2007). Numerous studies have indicated that cooperation among these subproblems can accelerate the search process and yield superior performance. Within this cooperative framework, several design choices come into play: (1) the definition of neighborhood relationships, (2) the timing of information exchange, and (3) the nature of the knowledge to be shared.

Neighborhood definition. A neighborhood $\mathbb{N}$ can be constituted of zero (no cooperation), multiple, or all other subproblems. The most common way to define neighborhoods between SOPs is to rely on the Euclidean distance of their associated weight vectors; see Figure 3. Yet, other techniques have also been published such as Murata et al. (2001), which relies on the Manhattan distance between weight vectors instead.

Exchange trigger. Information exchange between SOPs can occur at various moments throughout the search process, with the timing often determined by an exchange trigger (*exch* in Algorithm 2). There exist three primary approaches to information exchange. **Periodic exchange** involves regular and predetermined information sharing, such as at every iteration. **Adaptive exchange**, on the other hand, responds dynamically to events occurring in the search process, such as when specific improvements are achieved. Finally, **continuous exchange** is usually achieved through a shared structure that constantly disseminates information to guide the search process for all neighboring SOPs.

Exchange mechanism. The method of cooperation, denoted as *Cooperate* in MOO/D, among subproblems is closely associated with the specific search algorithm being employed. Presently, many decomposition techniques are integrated with genetic algorithms, which perform information exchange among SOPs by employing crossover operators on the solutions. This enables the sharing of solution components and relevant data. An alternative approach involves sharing part of the search memory, as seen in the case of ant colony algorithms, where elements like the pheromone matrix are shared among subproblems to guide the search process (Ke, Zhang, & Battiti, 2013).

3.5 Selection

As the weight vectors and reference points attached to individuals within the population $\mathbb{P}$ can change dynamically, an exponential number of combinations arises for each optimization step. To address this, various selection mechanisms (denoted as *Select* in MOO/D) have been developed to choose a subset of individuals, along with their associated weights and reference points, for each search iteration.

For example, existing approaches **rank** solutions according to their scalarized values and select the best ones observed thus far, using a static weight vector and reference point (Zhang & Li, 2007). In contrast, another method involves random generation of new weight vectors, followed by **tournament-style selection** to determine which solution will initiate the local search phase (Ishibuchi & Kaige, 2004). Alternative techniques propose **random selection** or systematic iteration through the available choices, akin to a **roulette wheel** selection process (Talbi, 2009).

3.6 Archive

With the dynamic parts of the algorithms explained above, a SOP search may find a worse solution after adaptation. This could lead to a degradation of the quality of the population. To solve this issue, modern algorithms often rely on the concept of external archive population ($\mathbb{EP}$), which stores the non-dominated solutions seen so far. The Pareto dominance criterion (Equation 3) can be used as a pruning function (*Prune* in MOO/D) to determine which individuals to keep in the archive. In case there are too many incomparable solutions, the size of the archive can be limited by using additional techniques, *e.g.* crowding distance (Deb et al., 2002).

3.7 Summary

Section 2 presented the RL problem, as well as its building blocks, and concluded by discussing its limited expressivity for multi-objective problems. This section presented an overview of MOO/D, which aims at solving multi-objective problems. However, there are a few notable differences between MOO/D and RL: (1) in MOO, the dynamics of the environment ($\vec{f}$, and the MOP model) are known by the algorithm and are generally deterministic, whereas RL involves solving sequential problems with unknown and potentially stochastic dynamics, (2) in MOO, the solutions are assignments of the decision variables of the problem, whereas they are reusable functions in RL.[1]

The upcoming section introduces the core contribution of this work, multi-objective reinforcement learning based on Decomposition (MORL/D). This approach establishes a taxonomy that bridges the gap between the two previously discussed fields, creating a shared vocabulary and facilitating the precise identification of research contributions within the context of MORL. Furthermore, this taxonomy enables researchers to recognize relevant concepts and findings from other related disciplines, fostering cross-disciplinary knowledge exchange. Moreover, the section reveals a unified framework that aligns with this taxonomy, demonstrating how techniques from both the realms of RL and MOO/D can be integrated.

1. Generating reusable functions to optimize problems is known as Hyper-Heuristics in optimization (Burke et al., 2013).

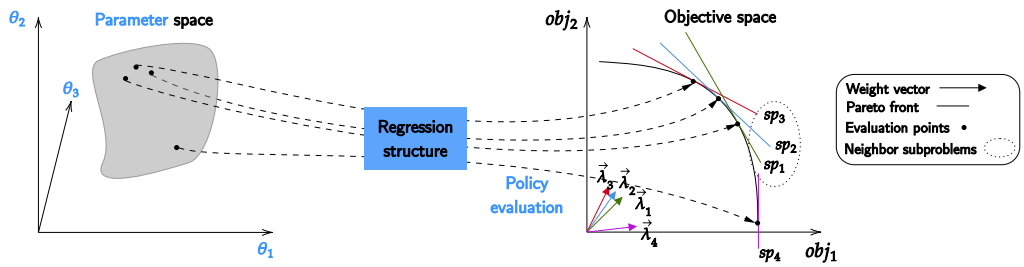


Figure 5: The decomposition idea applied to MORL. Blue parts emphasize the parts coming from RL, while black parts come from MOO. The optimization is looking for the best parameters for the regression structure to generate good policies. The idea of neighbor policies is that policies that have similar parameters should lead to close evaluations.

4. Multi-Objective Reinforcement Learning Based on Decomposition (MORL/D)

To extend RL to multi-objective problems, MORL models the problem as a multi-objective MDP (MOMDP). A MOMDP alters the MDP by replacing the reward function with a vectorial reward signal $\vec{r} : \mathbb{S} \times \mathbb{A} \times \mathbb{S} \rightarrow \mathbb{R}^m$ (Roijers & Whiteson, 2017). As mentioned earlier, one of the key differences between MOO and RL lies in the fact that RL aims at learning (optimizing) a policy, while MOO aims at optimizing a Pareto set of solutions. In the middle ground, MORL aims at learning a Pareto set of policies Π, with each policy parameterized by $\theta \in \Theta$ under function approximation.

In MORL, as in classical RL, the evaluation of policies is based on a sequence of decisions, and one usually runs a policy π for a finite amount of time to estimate its vector value function $\vec{v^\pi}$. This value, which can be considered as the equivalent of $\vec{f}(x)$ for MORL (see Figure 5), is the direct vectorial adaptation of v^π. Formally, a MORL policy has a vector value that is computed using the following:

$$\vec{v^\pi} = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \vec{r}(s_t, a_t, s_{t+1}) | \pi, \mu_0\right]. \tag{5}$$

Similar to MOO, in MORL, the DM's preferences are typically unknown during the training phase. This lack of knowledge prevents the establishment of a total ordering of actions at training time, as these actions are often Pareto incomparable. As a result, MORL algorithms are typically designed to be trained offline and executed once a specific trade-off has been chosen (Hayes et al., 2022). Our work focuses primarily on the MORL learning phase, which involves the process of training multiple policies to present a PF to the user. Notably, multiple MORL algorithms, environments, and software tools aiming at tackling such an issue have recently been published (Roijers et al., 2013; Hayes et al., 2022; Felten et al., 2023).

Pareto-based methods. In the realm of MORL, certain algorithms adopt an approach that involves directly learning a set of policies within the regression structure (Van Moffaert & Nowé, 2014; Ruiz-Montiel, Mandow, & Pérez-de-la Cruz, 2017). In these algorithms, each

Q-value is designed to store a set of Pareto optimal value vectors that can be achieved from the current state. However, it is worth noting that these algorithms are currently limited to tabular structures and face challenges when applied to problems with high dimensions. Despite efforts to develop Pareto-based MORL algorithms using function approximation techniques, learning to produce sets of varying sizes for different state-action pairs remains a challenging problem (Reymond & Nowe, 2019). Furthermore, even when the agent can learn a set of optimal policies for each action, questions persist about how to effectively order actions during the training phase as these are Pareto incomparable (Felten et al., 2022).

Decomposition-based methods. To solve such issues, multiple MORL algorithms relying on decomposition (MORL/D) techniques have been presented (Felten et al., 2022). The utilization of decomposition techniques offers several advantages for MORL. Firstly, by scalarizing rewards with different weights, these algorithms can often leverage single-objective RL techniques to learn multiple optimal policies. This allows MORL to directly benefit from advancements in RL research. Additionally, scalarization provides a method for ordering the evaluations of potentially Pareto incomparable actions during the training phase, enabling the selection of actions in a greedy manner based on certain weights. Based on these findings, it seems natural that part of the techniques from MOO/D can be transferred to MORL/D.

In this trend, in the same vein as single solution-based MOO, the most straightforward algorithm that we call vanilla outer loop, proposes to sequentially train single-objective RL with different weights applied in the scalarization to target different parts of the objective space (Roijers & Whiteson, 2017). Obviously, when compared to single-objective RL, this naive approach requires significantly more samples from the environment to compute various policies. Although MORL is a relatively new field of research, various enhancements and variations, often based on cooperation schemes, have already been proposed to enhance the sample efficiency of MORL/D algorithms (Figure 5). Nevertheless, to the best of our knowledge, no prior work has undertaken the task of systematically studying these recurring patterns and categorizing them within a comprehensive taxonomy, a gap that this present work seeks to fill.

4.1 Taxonomy and Framework

Algorithm 3 defines a high-level skeleton of the MORL/D framework, a middle-ground technique between MOO/D and RL aimed at finding a set of solutions to MOMDPs. Here, the population of policies (solutions) is denoted Π to clearly identify the different solution concepts with RL and MOO. The algorithm starts by initializing the policies, weights, reference points, Pareto archive, neighborhoods, and buffer (lines 1–5). At each iteration, the algorithm samples some experiences from the environment by following a chosen policy (lines 7–8). These experiences are then included in an experience buffer and used to improve the policies by sampling u batches from the buffer for each policy in the population (lines 9–10). After improvement, the policies are evaluated, and the Pareto optimal policies are included in the Pareto archive (lines 11–12). Then, the weights, reference points, and neighborhood are adapted (lines 13–14), and subproblems can cooperate with each other (line 15). Finally, the set of non-dominated policies is returned (line 17).

Algorithm 3 MORL/D high-level framework. Blue parts emphasize concepts coming from RL while black parts come from MOO/D.

Input: Stopping criterion $stop$, Population size n, Scalarization method g, Exchange trigger $exch$, Environment env, Update passes u.
Output: The approximated Pareto set stored in the external archive population $\mathbb{EP}$.

$\Pi, \mathbb{W}, \mathbb{Z} = Initialize(n)$
$evals = EvaluatePolicies(\Pi)$
$\mathbb{EP} = Prune(\Pi, evals)$
$\mathbb{N} = InitializeNeighborhood(\Pi, \mathbb{W})$
$\mathbb{B} = \emptyset$
while $\neg stop$ **do**
  $\pi, \vec{\lambda}, \vec{z} = Select(\Pi, \mathbb{W}, \mathbb{Z})$
  $experiences = Sample(env, \pi, g, \vec{\lambda}, \vec{z}, exch)$
  $\mathbb{B} = UpdateBuffer(\mathbb{B}, experiences)$
  $\Pi = ImprovePolicies(\Pi, g, \mathbb{W}, \mathbb{B}, u)$
  $evals = EvaluatePolicies(\Pi)$
  $\mathbb{EP} = Prune(\mathbb{EP} \cup \Pi, evals)$
  $\mathbb{W}, \mathbb{Z} = Adapt(\mathbb{W}, \mathbb{Z}, \mathbb{EP})$
  $\mathbb{N} = UpdateNeighborhood(\Pi, \mathbb{N}, \mathbb{W})$
  $Cooperate(\mathbb{N})$
end while
return $\mathbb{EP}$

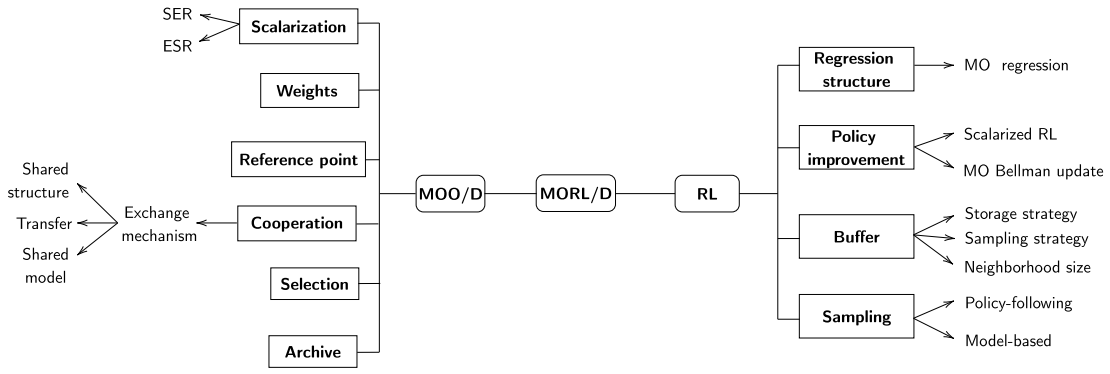


Figure 6: The Multi-Objective Reinforcement Learning based on Decomposition (MORL/D) taxonomy. Some traits from MOO/D and RL are directly applicable to this technique. Yet some alterations tailored for MORL have been published and are expanded in the figure.

Naturally, MORL/D benefits from a rich pool of techniques inherited from both MOO/D and RL. Some elements can be directly transferred and instantiated, while others require specific adaptations and novel approaches. Figure 6 visually highlights these design choices,

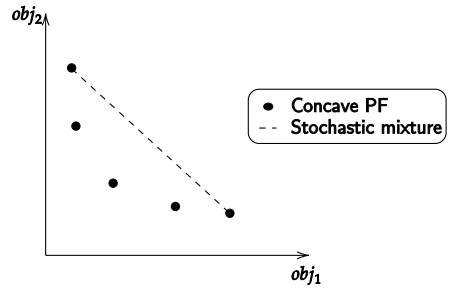


Figure 7: On average, policies resulting from a stochastic mix of deterministic policies dominate the policies in the concave part of the Pareto front.

and the following parts delve into some of these choices in more depth, accompanied by examples from existing MORL literature that employ these techniques.

4.2 Common Design Choices with MOO/D

Most of the building blocks previously identified in MOO/D can immediately be applied to MORL/D. Indeed, weight vectors or reference points generation and adaptation schemes, Pareto archive, and individual selection mechanisms can be applied straightforwardly. However, some building blocks require particular attention; these are discussed in more detail below.

4.2.1 Scalarization

In RL, the evaluation of a policy comes from multiple rewards, collected over a sequence of decisions made by the agent. To reduce these to a scalar for evaluation, RL usually relies on the expectation of the discounted sum of rewards. In MORL/D, to have a total ordering of Pareto incomparable actions or to rely on single-objective policy improvements, a scalarization function is used. To transform this sequence of reward vectors into scalars for evaluation in MORL, the scalarization function can be applied before or after the expectation operator. Hence, the algorithm can optimize for the Expected Scalarized Return (**ESR**), or the Scalarized Expected Return (**SER**).

When using a linear scalarization, both settings are equivalent. However, the ESR and SER settings lead to different optimal policies when the scalarization is not linear (Roijers, Steckelmacher, & Nowe, 2018). Formally:

$$\pi^*_{\text{SER}} = \underset{\pi}{\operatorname{argmax}}\, g\left(\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \vec{r}_t | \pi, s_0\right]\right) \neq \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}\left[g\left(\sum_{t=0}^{\infty} \gamma^t \vec{r}_t\right) | \pi, s_0\right] = \pi^*_{\text{ESR}}.$$

When the learned policies are stochastic, the concave part of the PF is dominated by a stochastic mixture of policies (Vamplew et al., 2009). This can be achieved, for instance, by employing one policy for one episode and another for the subsequent episode (see Figure 7). Hence, in such cases, one can restrict to the usage of linear scalarization.

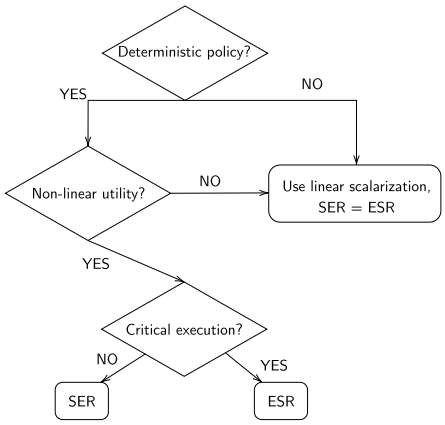


Figure 8: ESR vs. SER decision process.

Finally, the choice between ESR and SER depends on the criticality of the resulting policy decisions. ESR, where scalarization is applied before the expectation, is suitable for critical applications like cancer detection, as every episodic return is crucial. In contrast, SER applies scalarization on the average return, making it suitable for repetitive applications, such as investments. Given these observations, capturing policies in the concave parts of the PF is valuable when the user has a non-linear utility and aims to learn deterministic policies under the ESR criterion. Figure 8 summarizes the decision process to choose between ESR and SER settings in MORL. It is worth noting that in the literature most of the published algorithms optimize for the SER setting or under linear scalarization, leaving ESR understudied (Hayes et al., 2022; Röpke et al., 2023).

4.2.2 Cooperation

To improve the sample efficiency compared to the single solution approach (vanilla outer loop), some works rely on cooperation mechanisms where the information gathered by a policy is shared with other policies. It is worth noting that RL policies are often over-parameterized, and thus, policies having very different parameters may lead to close evaluation points in the objective space. However, in MORL, these parameters are often kept similar. Usually, this is enforced by the fact that policies are initialized to be very close to each other (*e.g.* via transfer learning), or their parameters are kept close to each other via the cooperation mechanisms.

While the neighborhood definitions and exchange triggers from MOO/D can be readily transferred to MORL/D, the knowledge exchange mechanism in MORL/D requires specific attention. This is because policies in MORL/D are typically encoded in regression structures or tables, which is different from the decision variable encoding in MOO/D. This distinction between MOO/D and MORL/D significantly impacts the design of information exchange techniques in the context of multi-objective reinforcement learning. Based on the surveyed articles in the MORL literature, we identify three ways to exchange information between neighbors.

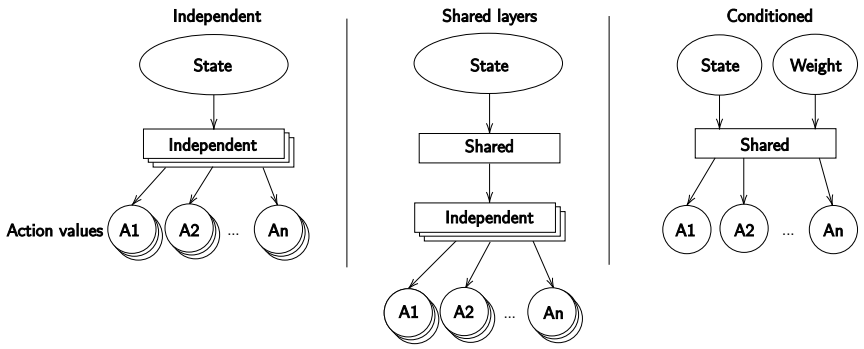


Figure 9: Shared regression structure schemes. Independent policies and conditioned regression lie on both ends of the shared regression spectrum. Independent policies do not share any parameter, whereas conditioned regression allows encoding multiple policies into a single regression structure. Shared layers lie in the middle, sharing part of the network parameters while leaving some to be independent.

Shared regression structure. While vanilla outer loop typically relies on completely independent policies (Figure 9, left), some works propose to share information between distinct policies by directly sharing part (or all) of the regression structure. This way, part of the parameters are shared, allowing to learn fewer parameters but also to share gathered experiences with multiple policies at the same time. In this fashion, Chen et al. (2020) proposes to share some base layers between multiple DNNs representing the policies (Figure 9, middle). More extreme cases, such as Abels et al. (2019), propose using only one regression structure by conditioning the input on the weights (Figure 9, right). This approach, often referred to as Conditioned Network, also works for regression trees (Castelletti et al., 2013), hence we refer to it as conditioned regression (CR).[2] While it may provide faster convergence to new points in the PF, this technique may forget previously trained policies unless tailored mechanisms are deployed (Abels et al., 2019).

Transfer. Transfer learning is a common machine learning technique that involves utilizing parameters from a previously trained regression structure to initiate the training of a new regression. In the context of MORL, transfer learning has been applied to various algorithms, allowing the training of a new policy to start from the parameters of the closest trained neighbor policy. This approach has demonstrated improved performance in some MORL algorithms (Roijers, Whiteson, & Oliehoek, 2015; Natarajan & Tadepalli, 2005).

Shared model. Another way to exchange information between policies is to share a model of the environment. The idea is that the environment dynamics, *i.e.* t and $\vec{r}$, are components that need to be estimated, but are the same for each policy that needs to be learned.

In this fashion, the simplest way to share a model is to use experiences sampled from the environment by one policy to apply Bellman updates to neighbor policies. Such a technique can drastically improve the sample efficiency of MORL methods, yet it is restricted to use an

2. This is similar to what is termed *parameter sharing* in multi-agent RL, *e.g.* Yu et al. (2022).

underlying off-policy learning algorithm. This idea is similar to sharing the search memory in MOO/D and is usually achieved by sharing the experience buffer between multiple policies in MORL (Abels et al., 2019; Chen et al., 2020).

Another technique proposes to use model-based reinforcement learning to learn the dynamics and rewards of the environment in a surrogate model, and to use the learned model to generate samples to learn policies (Wiering et al., 2014; Alegre et al., 2023). Such an approach also improves sample efficiency in MORL algorithms. This setting is very similar to what is called "same dynamics with different rewards" in model-based RL (Moerland et al., 2023).

4.3 Common Design Choices with RL

Naturally, existing techniques from RL can also be adapted and applied directly in MORL/D. However, due to the unique multi-objective setting, certain components and aspects of these techniques may need to be modified to accommodate the specific requirements of MORL. This section delves into the design choices and adaptations that originate from the field of RL but are relevant when applied in the context of MORL/D.

4.3.1 Regression Structure

As in classical RL, the choice of whether to use a function approximation or a tabular representation depends on the problem the user wants to tackle. Different regression structures allow for different sharing mechanisms. For example, transfer could be applied to any regression structure, but it may be less clear how to effectively share layers in a tabular setting. Independently, such a structure could also be adapted to incorporate multi-objective aspects.

Multi-objective regression structure. Several authors propose to slightly modify the way information is learned and encoded. Conditioned regression, as discussed above, includes the weight vectors as input to the regression structure. Moreover, it is also possible to vectorize the value function estimator. For example, multi-objective DQN (Mossalam et al., 2016) outputs $|\mathbb{A}| \times m$ components, meaning that for each action, it predicts m values corresponding to each objective. Some results suggest that these vectorized value function structures may be more efficient than the scalarized ones (Abels et al., 2019), possibly because the regression structure does not need to capture the scalarization being used when maintaining vectorized estimators.

4.3.2 Policy Evaluation and Improvement

In RL, a crucial aspect is the policy improvement step, which typically relies on Bellman update equations. However, standard Bellman updates are designed to handle scalar rewards, whereas MORL deals with vectorized rewards. To adapt the Bellman update to multi-objective settings, various techniques and approaches have been proposed in the MORL literature.

Scalarized update. The simplest approach, called scalarized update, is to use the scalarization function and rely on the Bellman updates from single-objective RL (Mossalam et al., 2016; Chen et al., 2020). The main advantage of this approach is that one can rely on the

vast RL literature. For example, from an experience tuple $(s_t, a_t, \vec{r}_t, s_{t+1})$ and a given weight vector $\vec{\lambda}$, scalarized Q-Learning (Van Moffaert, Drugan, & Nowe, 2013) adapts Equation 2 to the following Bellman update:

$$\tilde{q}(s_t, a_t) \leftarrow \tilde{q}(s_t, a_t) + \alpha \left(g(\vec{r}_t, \vec{\lambda}) + \gamma \max_{a' \in \mathbb{A}} \tilde{q}(s_{t+1}, a') - \tilde{q}(s_t, a_t) \right).$$

Multi-objective Bellman update. Other approaches propose to modify the regression function to include more multi-objective aspects. In some cases, such as when relying on a conditioned regression, it makes more sense to optimize the predictions towards a good representation of the full PF. In this sense, Yang, Sun, and Narasimhan (2019) proposes a Bellman update called *Envelope*, which optimizes not only across actions for each state but also for multiple weights over a space of preferences Λ. It relies on the weighted sum scalarization g^{ws} and stores the q-values as vectors. The update is defined as follows:

$$\vec{q}(s_t, a_t, \vec{\lambda}) \leftarrow \vec{q}(s_t, a_t, \vec{\lambda}) + \alpha \Bigg(\vec{r}_t + \gamma \left(\arg_{\vec{q}} \max_{a' \in \mathbb{A}, \vec{\lambda}' \in \Lambda} g^{\text{ws}}\big(\vec{q}(s_{t+1}, a', \vec{\lambda}'), \vec{\lambda}\big) \right) - \vec{q}(s_t, a_t, \vec{\lambda}) \Bigg),$$

where the $\arg_{\vec{q}} \max_{a' \in \mathbb{A}, \vec{\lambda}' \in \Lambda}$ operator returns the $\vec{q}$ vector maximizing over all actions and weights.

4.3.3 Buffer Strategy

While experience buffers are common in classical RL to enhance learning efficiency, their utilization in MORL differs somewhat. In MORL, the agent aims to learn multiple policies simultaneously, which can impact how experiences are stored and shared. Hence, the choice of how to manage experience buffers in MORL is influenced by the need to support the learning of multiple policies.

Replacement. The classic way to organize an experience buffer is to store experiences based on their recency, where old experiences are discarded for new ones, also called first-in-first-out (FIFO). However, the problem with this kind of reasoning in MORL is that the sequence of experiences sampled by policies on different ends of the PF will probably be very different. Hence, the stored information may not be very useful to improve some policies. Thus, this replacement criterion of the buffer could be changed to include multi-objective criteria. Instead of replacing all the experiences in the buffer at each iteration, some elected experiences could stay in the buffer for various iterations. This allows for keeping a diverse set of experiences in the buffer. For example, Abels et al. (2019) proposes to store the return of a sequence of experiences as a signature for the sequence in the replay buffer and a crowding distance operator (Deb et al., 2002) is used to keep the diversity of experiences in the buffer.

Selection. Independently, the way to select which experience to use from the buffer to improve a policy can also include multi-objective criteria. The simplest sampling strategy is to pick experiences from the buffer uniformly. However, this strategy might select a lot of experiences leading to poor learning. As in classical RL, one way to improve the quality of these samples is to rely on priorities (Schaul et al., 2016). Notably, this technique has been

adapted to MORL settings, where each policy keeps its own priority based on its weights (Abels et al., 2019; Alegre et al., 2023).

Neighborhood size. Instead of having either one buffer for all policies or one buffer per policy, an intermediate granularity of buffers could be shared between neighboring policies. In this way, the experiences in the buffer should be collected by policies that are close to the one being updated. The only existing work that has been found to address this idea is an ablation study presented in Chen et al. (2020).

4.3.4 Sampling Strategy

As previously mentioned, the goal of an MORL agent is to learn multiple policies and is generally applied in an offline setting. Thus, in the training stage, there is no such thing as best action for each state, since these can be Pareto incomparable. In addition, various policies will probably lead to the exploration of different areas in the environment. Thus, adapting RL sampling strategies might be beneficial in such cases.

Policy following. The straightforward adaptation of the RL sampling strategy to MORL is the one implemented in Algorithm 3. At each iteration, the agent chooses one policy to follow and samples the environment according to this policy. The choice of which policy to execute at every iteration is very similar to the individual selection problem in MOO/D. Hence, methods coming from MOO/D can be readily applied. Additionally, it is also possible to use multiple policies in parallel to fill the buffer at each iteration (Chen et al., 2020).

Model-based. Another approach consists in constructing a model of the environment to systematically sample different areas (Moerland et al., 2023). This allows for a more global sampling strategy. In this fashion, Felten et al. (2022) proposes to rely on metaheuristics to control the exploration of the agent and to use multi-objective metrics for the exploitation.

4.4 Summary

This section introduced the MORL/D taxonomy and framework, which is based on the integration of RL and MOO/D concepts. The taxonomy allows for the classification and description of existing MORL works while also serving as a guide for new research directions. The framework, with its adaptable nature, enables the direct transfer of knowledge from RL and MOO/D to MORL/D and provides a structured approach for assessing the effectiveness of new ideas.

The following sections explore the practical use of this contribution. First, we classify existing works according to the taxonomy, demonstrating its ability to comprehensively capture the core components of various MORL works. Then, we discuss how to instantiate the framework to address real-world problems. Finally, we provide evidence of the practical application of the framework by conducting experiments on established benchmark problems.

5. Using MORL/D

After presenting our taxonomy and framework in the last section, this section illustrates some of its use cases. First, we show how the taxonomy can be used to comprehend existing and future works. Then, we discuss how to make choices about the practical instantiation of the framework to solve a given problem.

5.1 Classification of Existing Work into our Taxonomy

This section categorizes three existing works within our taxonomy, demonstrating how the introduced terminology may be applied to position contributions in the broader context of MORL. These works have been chosen because they are representative of typical MORL/D instantiations. For a more exhaustive list of classified works, refer to the table in Appendix B. Note that reference points are omitted from the classification as all the methods studied in this context rely on linear scalarization, reflecting the current state-of-the-art in MORL.

5.1.1 PGMORL (Xu et al., 2020a)

Prediction-guided MORL (PGMORL) (Xu et al., 2020a) is an evolutionary algorithm that maintains a population of policies learned using a scalarized version of proximal policy optimization (PPO) (Schulman et al., 2017). Below, we categorize its components within the MORL/D taxonomy.

Scalarization. PGMORL relies on the weighted sum scalarization.

Weight vectors. The algorithm generates weight vectors uniformly in the objective space at the beginning of the process.

Cooperation. This algorithm does not enforce cooperation between policies.

Selection. PGMORL introduces a prediction model that aims to forecast improvements on the PF from previous policy evaluations and specific weight vectors. It uses this predictive model to choose pairs of policies and weight vectors that are expected to result in the most significant predicted improvements, following a tournament-style selection approach.

Archive. A Pareto archive is employed to maintain snapshots of the best policies over the course of the learning process. Pareto dominance is used as pruning criterion.

Regression structure. PGMORL maintains a population of both actors and critics, where the critics are multi-objective.

Policy improvement. The algorithm adopts a scalarized variant of PPO (Schulman et al., 2017) for policy improvement.

Buffer. The buffers in PGMORL operate independently and adhere to the classic PPO strategy, storing experiences based on recency and employing uniform selection.

Sampling. Policy following is the approach adopted in this algorithm. The experiences gathered are used exclusively to train the current policy. This restriction is attributed to the nature of PPO (Schulman et al., 2017), which is an on-policy algorithm and offers limited flexibility in terms of sampling strategies.

5.1.2 Multi-policy SAC (Chen et al., 2020)

Multi-policy soft actor-critic (Chen et al., 2020) is a two-stage algorithm that first applies an MORL/D method to learn a set of policies, then uses an evolutionary strategy to finish the search based on the policies found in the first phase. The MORL/D phase keeps a set of policies attached to predefined weight vectors for the entire process. This means that if the user specifies 5 weights, then this phase will return 5 policies. The paragraphs below describe the first phase of the algorithm.

Scalarization. Multi-policy SAC utilizes the weighted sum scalarization approach.

Weight vectors. The weight vectors are set manually before the training process starts. These are never changed during the MORL process.

Cooperation. This algorithm incorporates unique cooperation strategies. Firstly, the policies share base layers (Figure 9, middle), facilitating the exchange of information across all policies during the application of gradients in the policy improvement process. Additionally, various experience buffer sharing strategies are explored in the paper, including scenarios where each policy has its own buffer, where each policy shares with adjacent neighbors (those with the closest weight vectors), and where each policy has access to all buffers (a neighborhood that includes all policies).

Selection. At each iteration, all policies are run in a roulette wheel fashion. The weights are kept attached to their policies.

Archive. No archive is used in this algorithm.

Regression structure. This algorithm keeps track of multiple SAC policies (Haarnoja et al., 2018).

Policy improvement. The rewards are scalarized before any interaction with the learning process. Hence, the SAC policies are pure single-objective RL.

Buffer. Buffer replacement and selection strategies adhere to conventional practices, recency and uniform selection, respectively. However, this algorithm implements a shared buffer, as explained in the cooperation paragraph above.

Sampling. All policies are used at each iteration for sampling, reflecting a policy following strategy.

5.1.3 Dynamic weight-conditioned RL (Abels et al., 2019)

Dynamic weight-conditioned RL is a conditioned regression algorithm that models multiple policies into a single regression structure by adding a weight vector as input to the model. This allows such an algorithm to change weight vectors dynamically and thus be employed online but also allows for efficient parameter sharing.

Scalarization. Weighted sum is used in this algorithm.

Weight vectors. The weight vectors are generated periodically in a random manner.

Cooperation. This algorithm promotes cooperation among multiple policies by sharing parameters using conditioned regression (as illustrated in Figure 9).

Selection. Since all policies are modeled within a single structure, only the weights need to be selected. As explained above, these are randomly generated.

Archive. No Pareto archive is used in this work.

Regression structure. The algorithm keeps one DQN-based conditioned network that outputs multi-objective Q-values for each action.

Policy improvement. The algorithm scalarizes the rewards in each experience according to both the current and historical weights. This approach is employed to facilitate learning for the current weight while preventing the loss of knowledge related to previously learned policies.

Buffer. Various buffer strategies are proposed in this work. First, regarding replacement, it suggests maintaining a diverse set of experiences within a replay buffer, which accelerates learning for a wide range of weight vectors. The diversity criterion is enforced by tracking the return associated with the experience's sequence of actions. A crowding distance operation, equivalent to NSGA-II's (Deb et al., 2002), is then applied to these returns to quantify buffer diversity. Regarding selection, the authors propose the use of prioritized experience replay, where the priority is computed based on the temporal difference error for the active weight vector and a historical weight vector. For each experience selected for learning, the rewards are scalarized using both the current weight vector and a historical weight vector, akin to the concept of hindsight experience replay in RL (Andrychowicz et al., 2017). This is a crucial step to prevent forgetting past policies.

Sampling. Sampling is done by following the policy associated with the current weight.

5.2 How to Instantiate MORL/D?

As MORL/D presents a combinatorial number of potential instantiations, it may be hard to choose which technique to apply to solve a given problem. This part discusses the decisions of choosing which MORL/D variant to apply given a problem.

First, we provide a formal decision process regarding linear and non-linear scalarization choice in Section 4.2.1 and a graphical representation is illustrated in Figure 8. Additionally, some combinations of techniques do not make sense. For example, conditioned regression may not be combined with transfer learning, and shared layers are not compatible with tabular algorithms. Hence, using common sense on the presented techniques, algorithm designers may be able to restrict the number of possibilities.

However, for some parts, it is not possible to give absolute directions since a technique that performs well for a given environment might perform poorly for another, as the "no free lunch" theorem states (Wolpert & Macready, 1997). Yet, our framework may open ways to solve such an issue by allowing to automate the design of MORL/D algorithms. In such a context, an optimization algorithm is applied to search the space of possible instantiations of MORL/D and decide how to instantiate the parts of the algorithm in order to maximize its performance for a given problem. In RL, these kinds of approaches are referred to as "AutoRL" (Parker-Holder et al., 2022; Eimer et al., 2023; Felten et al., 2023). Hence, we believe our framework could be useful to extend such work to form an AutoMORL solver. Nevertheless, the next section illustrates in practice how we tackle different benchmark problems without such an automated solver at our disposal.

6. Experiments

The last sections introduced the MORL/D framework and taxonomy, and existing works have been classified using the latter. This section shows that the MORL/D framework is not limited to theoretical analysis but is also fit for application. MORL/D has been instantiated to tackle two contrasting multi-objective benchmarks (Alegre et al., 2022), showing how versatile the framework can be. Indeed, the two studied environments contain concave and convex PF, and involve continuous and discrete observations and actions. Each set of experiments has been run 10 times over various seeds for statistical robustness. Additionally, to give a reference, we compare MORL/D results against state-of-the-art methods. In practice, we reuse the data hosted by OpenRLBenchmark (Huang et al., 2024) to plot metrics from such state-of-the-art methods. MORL/D has been implemented using utilities from the MORL-Baselines (Felten et al., 2023) and Pymoo projects (Blank & Deb, 2020). Our experiments have been run on the high-performance computer of the University of Luxembourg (Varrette et al., 2014).[3]

6.1 Assessing Performance

MORL differs from RL in the way performance results are assessed. Indeed, since multi-policy MORL aims at returning a set of policies and their linked PF, different metrics than episodic return over training time must be used. In general, these metrics aim to turn PFs into scalars to ease comparison. There are two categories of metrics: utility-based metrics, which rely on specific assumptions about the utility function (such as linearity), and axiomatic metrics, which do not make assumptions, but may yield less informative performance information for users (Hayes et al., 2022; Felten et al., 2023).

6.1.1 Axiomatic Approaches

As discussed in Section 3, MOO/D methods involve finding a good approximation of the PF with a focus on two critical aspects: convergence and diversity. To assess convergence, various performance indicators are employed, with the inverted generational distance (IGD) being one such metric (Coello Coello & Reyes Sierra, 2004). The IGD quantifies the distance between a reference Pareto front, denoted as $\mathcal{Z}$, and the current PF approximation $\mathcal{F}$. Formally, it is computed as follows:

$$\text{IGD}(\mathcal{F}, \mathcal{Z}) = \frac{1}{|\mathcal{Z}|}\sqrt{\sum_{\vec{z}\in\mathcal{Z}} \min_{\vec{v^\pi}\in\mathcal{F}} \|\vec{z} - \vec{v^\pi}\|^2}.$$

On the other hand, to evaluate diversity, metrics such as sparsity (Xu et al., 2020a) come into play. Sparsity measures the average squared distance between consecutive points in the PF approximation:

$$S(\mathcal{F}) = \frac{1}{|\mathcal{F}|-1}\sum_{j=1}^{m}\sum_{i=1}^{|\mathcal{F}|-1}(\tilde{P}_j(i) - \tilde{P}_j(i+1))^2,$$

3. The code used for experiments is available in MORL-Baselines https://github.com/LucasAlegre/morl-baselines.

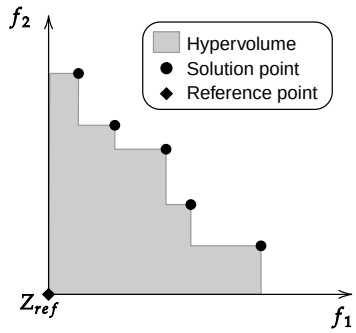


Figure 10: The hypervolume metric in a two objective problem for a given PF.

where $\tilde{P}_j$ is the sorted list for the jth objective values in $\mathcal{F}$, and $\tilde{P}_j(i)$ is the ith value in this sorted list.

Additionally, there are hybrid methods designed to provide insight into both convergence and diversity, such as the hypervolume metric (Zitzler, 1999; Talbi, 2009). Hypervolume quantifies the volume of the region formed between each point in the approximated PF and a reference point in the objective space. This reference point, $\vec{z}_{ref}$, should be carefully chosen as a lower bound for each objective, as illustrated in Figure 10.

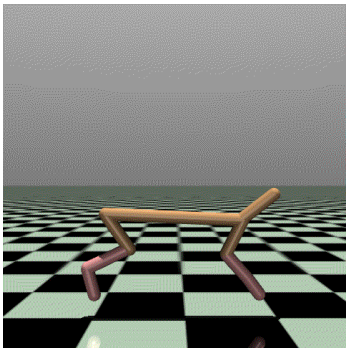
Because the Hypervolume metric offers a combined evaluation of both criteria, it is often plotted against the number of timesteps to offer insights into the learning curve within the MORL literature. However, we argue that relying solely on one metric may be insufficient to determine which algorithm outperforms others on a given problem, as illustrated by the experimental results presented later.

6.1.2 Utility-Based Approaches

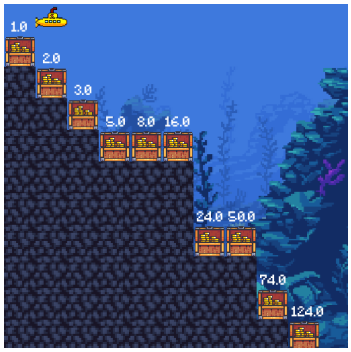
Alternatively, MORL algorithms often seek to maximize the utility of the end user. While the above-mentioned metrics give insights into which methods perform better, they give little information to the end user. To solve such an issue, performance metrics making use of the utility function of the user have also emerged in the fields of MOO and MORL (Talbi, 2009; Hayes et al., 2022). For example, Zintgraf et al. (2015) proposes the expected utility metric (EUM). This unary metric, close to the family of R-metrics in MOO (Talbi, 2009), is defined as follows:

$$\text{EUM}(\mathcal{F}) = \mathop{\mathbb{E}}_{\vec{\lambda}\in\Lambda}\left[\max_{\vec{v^\pi}\in\mathcal{F}} u(\vec{v^\pi}, \vec{\lambda})\right]$$

where u represents the utility function of the user (generally weighted sum), that it uses to choose a solution point on the PF, and Λ is a set of uniformly distributed weight vectors in the objective space.



(a) *mo-halfcheetah-v4*.



(b) *deep-sea-treasure-concave-v0*.

Figure 11: Studied environments from MO-Gymnasium (Alegre et al., 2022).

6.2 Benchmark Problems

The first problem, *mo-halfcheetah-v4* (Figure 11a), presents a multi-objective adaptation of the well-known Mujoco problems (Todorov, Erez, & Tassa, 2012). In this scenario, the agent takes control of a bipedal robot and aims to mimic the running behavior of a cheetah. The agent's objectives in this environment involve simultaneously maximizing its speed and minimizing its energy consumption. Unlike the original Mujoco implementation, which relies on a weighted sum with hard-coded weights to transform the problem into a single-objective MDP, the multi-objective version treats both objectives independently. Thus, it allows learning various trade-offs for each of these two objectives.

The second problem of interest is referred to as *deep-sea-treasure-concave-v0* (Figure 11b), wherein the agent assumes control of a submarine navigating through a grid-world environment (Vamplew et al., 2011). In this task, the agent's goal is to collect one of the treasures located on the seabed while striving to minimize the time spent traveling. The rewards for collecting treasures are directly proportional to their distance from the starting point, resulting in conflicting objectives. This benchmark problem holds particular significance due to the presence of a known PF that exhibits a concave shape. This concavity poses a challenge for MORL methods that rely on linear scalarization when the user's objective is to derive a deterministic policy, as discussed in Section 4.2.1.

6.3 Solving *mo-halfcheetah-v4*

This section explains how we tackled the *mo-halfcheetah-v4* problem using MORL/D and compares results against state-of-the-art methods implemented in MORL-Baselines (Felten et al., 2023).

6.3.1 MORL/D Variants

To tackle this task, we initially perform an ablation study for some components of MORL/D (Figure 6). Subsequently, we compare the best version found against various existing MORL algorithms.

The first MORL/D instantiation, which we refer to as MORL/D vanilla, neither performs any cooperation nor weight vector adaptation. We then tried to add cooperation

and weight vector adaptation to this vanilla algorithm and examine the results. In practice, we tried adding PSA's method to adapt weights (Equation 4), and a shared buffer as cooperation mechanism. When relying on one technique, we suffix the technique acronym to MORL/D, *e.g.* MORL/D SB PSA refers to a variant of the algorithm that implements a shared buffer and PSA's weight vector adaptation. It is worth noting that more sophisticated schemes coming from MOO or RL literature can easily be integrated too. Our instantiation is discussed in more detail below.

For this continuous problem, each policy in the population relies on a scalarized multi-objective version of the SAC algorithm (Haarnoja et al., 2018). Practically, the SAC implementation from Huang et al. (2022) was modified by including multi-objective critics and adding a scalarization function.

Scalarization. Because the learned policies need not be deterministic, its simplicity to implement, and to avoid making a choice between ESR and SER setting, the weighted sum has been chosen to instantiate MORL/D on this environment.

Weight vectors. For initialization, weight vectors are uniformly generated on a unit simplex using the Riesz s-Energy method from Blank et al. (2021). Moreover, some MORL/D variants perform PSA's weight vector adaptation (Equation 4) every 50,000 steps.[4]

Cooperation. MORL/D vanilla does not implement any cooperation mechanism, whereas MORL/D SB implements shared buffer across the entire population. In the latter, the neighborhood is all other policies in the population and the exchange happens continuously since the buffer is shared by everyone.

Selection. In all variants, each policy is attached to given weights (which can be adapted) and trained using those. To uniformly train all policies, candidates are chosen in a roulette-wheel fashion to sample the experiences.

Archive. To ensure no performance loss after weight adaptation, a Pareto archive has been implemented using the Pareto dominance criterion as sole pruning function. The archive stores snapshots of the SAC policies leading to Pareto optimal points.

Regression structure. The studied MORL/D variants are actor-critic methods based on neural networks. The critics have been modified to implement a multi-objective regression, *i.e.* each critic outputs m values.

Policy improvement. The scalarization function is applied on the muti-objective estimates from the critics to transform them into scalars. This allows falling back to the original implementation of the Bellman update with scalarized RL.

Buffer. Both MORL/D variants use experience buffers with a recency criterion for storage and sample uniformly from the buffers. In the case of MORL/D SB variants, a single buffer is shared among all the policies.

Sampling strategy. In all variations, samples are collected by following the selected candidate policy, *i.e.* policy following. Note that policy updates are also performed on the currently followed policy while following the policy, as in the original implementation of SAC.

4. In our experience, it is also beneficial to normalize the reward components to facilitate the finding of weight vectors which lead to a diverse PF when the scale between objective values is different.

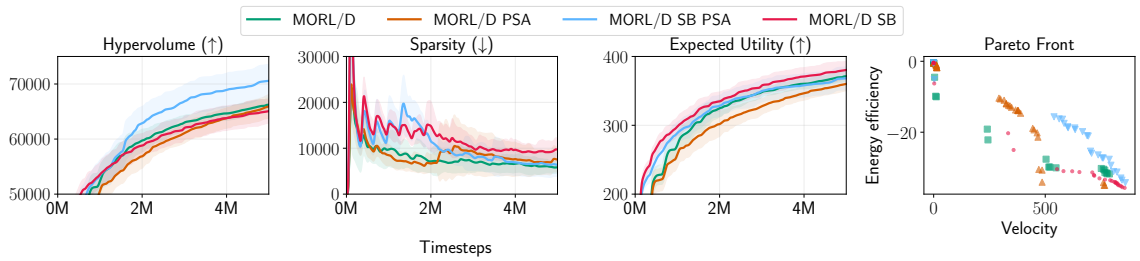


Figure 12: Average and 95% confidence interval of various metrics over timesteps on *mo-halfcheetah-v4* for variants of MORL/D. The rightmost plot is the resulting PF of each method's best run (best hypervolume).

A set of hyperparameters in both MORL/D and the underlying SAC implementations has been set to conduct our experiments. Table 1 in Appendix A lists those.

6.3.2 Experimental Results

Ablation study. Figure 12 shows the results of an ablation study of various versions of MORL/D on the *mo-halfcheetah* problem. A salient observation is that all MORL/D variants consistently improve their Pareto sets and PFs due to the utilization of the Pareto archive. When it comes to hypervolume (reference point $(-100, -100)$), it is evident that MORL/D with a shared buffer and PSA weight adaptation (MORL/D SB PSA) appears to outperform other variants in general. However, in terms of metrics such as sparsity and expected utility, no particular variant emerges as superior to the others.

The right plot of the PF illustrates how policies are distributed across the objective space for their best run.[5] Notably, this graph reveals an unequal scaling of objectives, which implies that scalarized values, based on uniformly spaced weights and metrics like expected utility, may exhibit bias in favor of objectives with larger scales. This phenomenon is corroborated by the fact that most of the Pareto optimal points are located on the right side of the plot, primarily because the velocity objective exhibits a larger scale compared to energy efficiency. Even though we normalize the rewards using a wrapper for this problem, it appears that this normalization is insufficient to learn a continuous PF. Nevertheless, the plot underscores that adding cooperation and weight adaptation to the vanilla MORL/D improves its performance.

MORL/D vs. state of the art. We now compare MORL/D SB PSA to the state-of-the-art methods in Figure 13 to gauge its performance against established baselines. Specifically, we evaluate our algorithm against PGMORL (Xu et al., 2020a) and CAPQL (Lu, Herman, & Yu, 2023). We have previously discussed the first algorithm in Section 5. On the other hand, it is interesting to note that CAPQL closely resembles the instantiation of MORL/D we have employed to address this problem. CAPQL, indeed, relies on scalarized SAC using a weighted sum. Nevertheless, it distinguishes itself by relying on conditioned regression and randomly sampling new weight vectors for each environment step.

5. It is worth emphasizing that such a plot reflects the performance of only one run, while the metric plots reflect the general performance over multiple runs.

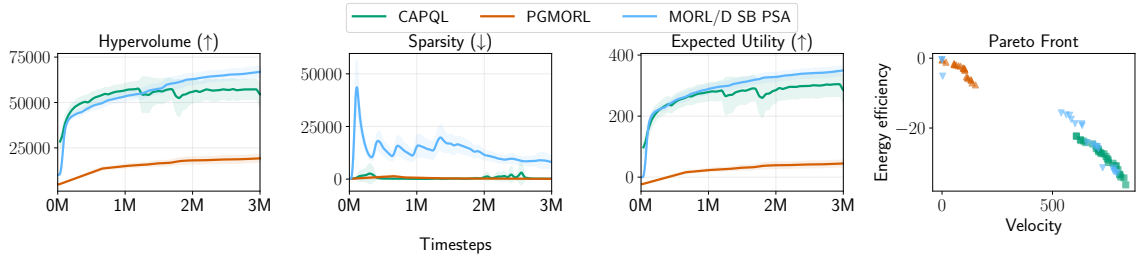


Figure 13: Average and 95% confidence interval of various metrics over timesteps on *mo-halfcheetah-v4*, MORL/D compared against state-of-the-art methods. The rightmost plot is the PF of each method's best run (best hypervolume).

Based on the data presented in Figure 13, our implementation achieves performance that is comparable to or even superior to the state of the art, particularly in terms of hypervolume and expected utility. This, in particular, contradicts the current belief that conditioned regression-based methods are more efficient than relying on multiple networks (Abels et al., 2019). It is worth noting that the performance of CAPQL exhibits occasional drops during its training phase, which we suspect may be attributed to the conditioned neural network employed in the algorithm that forgets previously learned policies. This issue does not arise when using multiple neural networks in conjunction with a Pareto archive. We also provide performance in terms of runtime in Appendix C.

Additionally, the PF plot reveals a noteworthy observation: while other algorithms tend to produce nearly continuous PFs on one side of the objective space, our algorithm discovers policies on both ends, contributing to improved diversity. However, sparsity appears to be more favorable in the case of the other algorithms. We believe that this highlights a limitation of this state-of-the-art metric: it assesses distance based on the points found by the algorithm rather than considering the entire objective space. Consequently, an algorithm that locates only a few closely clustered points may exhibit a low sparsity score despite providing smaller diversity.

6.4 Solving *deep-sea-treasure-concave-v0*

This section illustrates how MORL/D can be used to solve problems involving PFs with concave parts.

6.4.1 MORL/D Variant

In this section, our MORL/D algorithm depends on the expected utility policy gradient algorithm (EUPG) (Roijers et al., 2018). EUPG is a single-policy ESR algorithm able to learn policies with non-linear scalarization. Employing such an algorithm with different weights enables us to capture policies within the concave part of the PF, which is uncommon in the current MORL literature. This section demonstrates how MORL/D can serve as a framework for converting pre-existing single-policy MORL algorithms into multi-policy ones.

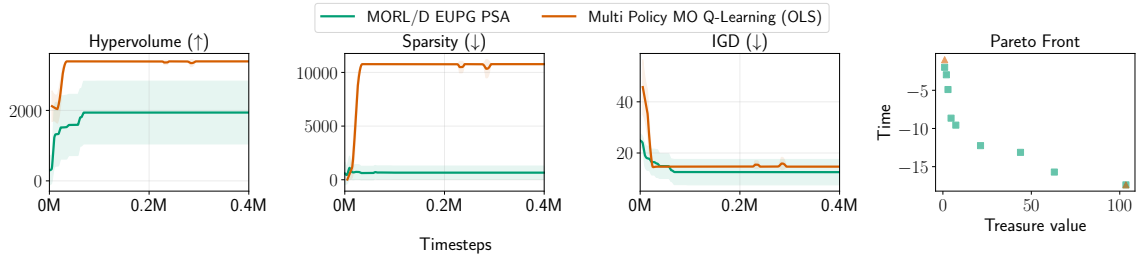


Figure 14: Average and 95% confidence interval of various metrics over timesteps on *deep-sea-treasure-concave-v0*. The rightmost plot is the resulting PF of each method's best run (best hypervolume).

Scalarization. In this case, we are interested in finding the concave points in the PF. Hence, we rely on the Chebyshev function (Section 3), which is a non-linear scalarization.

Reference points. The Chebyshev scalarization necessitates using a utopian reference point $\vec{z}$. In many cases, this reference point is hard to set in advance. Hence, we propose to automatically adapt it over the course of the learning process by setting $\vec{z}$ to be the maximum value observed for each objective, plus a factor $\tau = 0.5$.

Weight vectors. As for *mo-halfcheetah*, the Riesz s-Energy method is used to uniformly generate weight vectors. Moreover, PSA's weight vectors adaptation (Equation 4) is used every 1,000 steps.

Cooperation. No cooperation has been implemented on this problem.

Selection. Each policy is attached to given weights (which can be adapted) and trained using those. To uniformly train all policies, candidates are chosen in a roulette-wheel fashion to sample the next experiences.

Archive. A Pareto archive has been implemented using the Pareto dominance criterion as sole pruning function. The archive stores snapshots of the EUPG policies leading to Pareto optimal points.

Regression structure. EUPG relies on a single NN to model the policy. Interestingly, it proposes to condition the NN on the accrued reward to allow learning ESR policies with non-linear scalarization.

Policy improvement. This MORL/D variant relies directly on EUPG's policy improvement.

Buffer. This algorithm uses experience buffers with a recency criterion for storage and samples uniformly from the buffers.

Sampling strategy. Similar to the previous algorithm, samples are collected following the selected candidate policy, *i.e.* policy following. Note that policy updates are also performed on the currently followed policy.

The list of hyperparameter values used for these experiments is available in Table 2 in Appendix A.

6.4.2 Experimental Results

Figure 14 displays the training outcomes of our MORL/D variant, which relies on EUPG as its underlying algorithm. Additionally, we present results for a comparative analysis involving multi-policy multi-objective Q-learning (MPMOQL) (Van Moffaert et al., 2013), a tabular algorithm that depends on multi-objective regression and linear scalarization. Similarly to single-population algorithms in MOO, MPMOQL is run multiple times with various weights generated using optimistic linear support (OLS) (Roijers et al., 2015) to learn various trade-offs. It is worth emphasizing again that the use of linear scalarization restricts this algorithm from effectively learning points within the concave part of the Pareto front. We used $(0, -50)$ as a reference point for the hypervolume computation. Note that we do not report expected utility in this case, since the metric supposes linear utility, which would not capture any contribution from the concave points in the PF.

Upon examining the plots, it appears that MORL/D performs less favorably in terms of hypervolume compared to MPMOQL. However, when assessing diversity and convergence metrics, a different picture emerges: MORL/D outperforms MPMOQL in terms of sparsity and IGD, respectively. Furthermore, the PF revealed by the best-performing runs clearly demonstrates that MORL/D can capture points within the concave portion of the PF, whereas MPMOQL with linear scalarization can only capture extreme points along the convex hull.

Upon closer examination, we saw that MPMOQL consistently captures the two extreme points, while MORL/D occasionally struggles to capture points on the right side of the Pareto front, primarily due to limited exploration in EUPG. Notably, the two extreme points captured by MPMOQL result in a very high hypervolume, overshadowing the contributions from the points in the concave region. Consequently, even if MORL/D manages to capture a few points in the concave region, its hypervolume remains lower than that of MPMOQL, which captures only the two extreme points. This underscores the limitation of relying solely on a single performance metric, such as hypervolume or sparsity in the previous example, for algorithm comparison. Indeed, attempting to condense such a wealth of information into a single scalar value comes with inherent trade-offs. Therefore, we advocate for the utilization of multiple metrics and the visualization of PFs whenever possible as a more comprehensive approach.

7. Future Directions

This work gave insights on how to transfer existing knowledge from the fields of MOO and RL into MORL. The presented framework allows us to atomically test variations of the algorithm using existing or novel techniques, resulting in novel variations of MORL/D. From our MORL/D taxonomy (Figure 6) and the surveyed articles, we identify key points that have been less studied than others in the next paragraphs.

Non-linear scalarization. From the surveyed articles and as previously stated in Hayes et al. (2022), a significant portion of MORL algorithms predominantly rely on linear scalarization, thus leaving **non-linear scalarization** schemes comparatively less explored. This phenomenon might be attributed to the intricate challenges that non-linear scalarization introduces into MORL, as demonstrated by the complexities in the comparison of ESR and

SER, see Section 4.2.1. Moreover, the employment of linear scalarization in many existing works negates the need to **set or adapt reference points**, leading to this aspect receiving minimal attention in the MORL context. It is also worth noting that, to the best of our knowledge, no existing MORL algorithm has successfully ventured into combining **multiple scalarization functions** as has been practiced in the MOO/D domain (Ishibuchi et al., 2010).

Weight vector adaptation. In MORL, the generation of weight vectors is often carried out through manual pre-definition or uniform distribution within the objective space, leaving **weight vector adaptation** schemes being relatively underdeveloped. Notable exceptions can be found in the works of Roijers et al. (2015) and Alegre et al. (2023), which employ intelligent weight vector generation based on the current state of the PF. However, once again, it is important to highlight that these methods primarily assume the utilization of linear scalarization. There exist MOO/D methods that are ready to use in MORL/D, as exemplified by PSA (Equation 4) that we used in this work.

Cooperation schemes. The MORL/D landscape opens up possibilities for novel cooperation schemes. For instance, when employing multiple regression structures, there exists the potential to infer new policies by combining two existing ones, akin to the utilization of **crossover operators** in neuroevolution (Stanley et al., 2019) or soups of models (Wortsman et al., 2022). This approach enables the MORL/D algorithm to blend policy improvements with crossovers, facilitating the efficient generation of a Pareto set of policies.

Parallelizing MORL/D. The central focus of this article primarily revolves around enhancing sample efficiency, as often seen in RL. Yet, little attention has been given to the enhancement of sample throughput, which holds the potential to substantially reduce the training time required for MORL/D algorithms, even for less sample-efficient (but faster) algorithms.

In this context, maintaining the independence of policies, meaning the enforcement of no cooperation schemes, naturally lends itself to a straightforward approach of breaking down the problem and simultaneously addressing all subproblems in parallel, akin to embarrassingly parallel search (Régin, Rezgui, & Malapert, 2013). Though, to the best of our knowledge, the comprehensive exploration of fully parallelized MORL algorithms remains uncharted territory, with no known study having undertaken this multifaceted investigation.

Furthermore, there exists a middle ground where cooperation schemes and parallelized training of multiple policies may offer promising results. However, it is important to note that the sharing of information (cooperation) between subproblems in such scenarios can introduce bottlenecks in parallel programs due to the need for thread synchronization. This represents an exciting avenue for future research and innovation within the field of MORL.

Automated MORL. Having a modular framework that is instantiable with many techniques from both RL and MOO/D leads to a combinatorial number of choices of instantiation. As discussed in Section 5.2, our framework could be combined with automated design techniques to automatically choose well-performing algorithm components for a given problem.

8. Conclusion

This work first presented both fields of RL and MOO by breaking up existing solutions into atomic design choices. The differences and common points between these fields of research have been discussed and notable literature is surveyed.

Subsequently, the paper introduced multi-objective reinforcement learning based on decomposition (MORL/D), a methodology that draws inspiration from both RL and MOO. MORL/D's primary objective is to identify a collection of Pareto optimal policies for solving multi-objective sequential challenges. It employs a scalarization function to break down the multi-objective problem into individual single-objective problems. Building upon the foundations of MOO and RL, the paper introduces a taxonomy focused on solving methods that facilitates the categorization of existing and prospective MORL works. To showcase its utility, a portion of the existing MORL literature has been examined through the taxonomy.

Furthermore, the paper presented a unified framework based on the established taxonomy and adapted it in various ways to conduct experiments on diverse benchmark problems. These experiments demonstrate MORL/D's capacity to address a wide range of challenges through easily identifiable adjustments in the framework's instantiation. Notably, the experiments illustrated how one could port existing knowledge from MOO to MORL, *e.g.* weight vector adaptation while achieving competitive performance compared to current state-of-the-art methods. Moreover, the experimental results and the discussion unveiled important concerns when relying solely on performance metrics to evaluate algorithms.

Lastly, the taxonomy introduced in this work has been utilized to provide insights into potential directions for future MORL research, to leverage knowledge from MOO and RL, or to design entirely novel approaches tailored to the field of MORL.

Acknowledgments

We would like to express our gratitude to the reviewers and editors for their invaluable feedback during the review process. Additionally, we would like to thank Pierre Talbot and Maria Hartmann for proofreading the text.

This work has been funded by the Fonds National de la Recherche Luxembourg (FNR), CORE program under the ADARS Project, ref. C20/IS/14762457.

Appendix A. Reproducibility

This section presents the hyperparameter values used in our experiments. Our code can be found in the MORL-Baselines repository (Felten et al., 2023). Moreover, the results of all runs and hyperparameters are hosted in OpenRLBenchmark (Huang et al., 2024). Environment implementations are available in MO-Gymnasium (Alegre et al., 2022).

Appendix B. Additional Works Classified in our Taxonomy

Table 3 presents existing MORL work classified according to the proposed MORL/D taxonomy (Figure 6). This demonstrates the modularity of the proposed framework and offers a high-level view of the current state of the art in MORL. The table is separated by a double

	Hyperparameter	**Value**
MORL/D	Population size	$n = 6$
	Episodes for policy evaluation	5
	Update passes	$u = 10$
	Total environments steps	$5e6$
	Exchange trigger	Every 50,000 steps
	Scalarization	Weighted sum
	Weight adaptation	PSA, $\delta = 1.1$
SAC	Buffer size	10^6
	Gamma	0.99
	Target smoothing coefficient	0.005
	Batch size	256
	Steps before learning	15,000
	Hidden neurons in Neural Networks	[256;256]
	Activation function in Neural Networks	ReLU
	Actor learning rate	$3e{-}4$
	Critic learning rate	10^{-3}
	Actor training frequency	2
	Target training frequency	1
	Entropy regularization coefficient	automatic

Table 1: Hyperparameters for MORL/D on *mo-halfcheetah-v4*.

	Hyperparameter	**Value**
MORL/D	Population size	$n = 10$
	Episodes for policy evaluation	5
	Update passes	$u = 10$
	Total environments steps	$4e5$
	Exchange trigger	Every 1,000 steps
	Scalarization	Chebyshev
	Optimization criterion	ESR
	Weight adaptation	PSA: $\delta = 1.1$
EUPG	Buffer size	10^5
	Gamma	0.99
	Hidden neurons in NN	[32;32]
	Activation function in NN	Tanh
	Learning rate	$1e{-}4$

Table 2: Hyperparameters for MORL/D on *deep-sea-treasure-concave-v0*.

Reference	MOO					RL				
	Weight vectors		Cooperation					Buffer		
	When?	How?	Neighb.	Mechanism	Trigger	Regression structure	Policy improv.	Neighb.	Storage & Sampling Strategy	Sampling strategy
Roijers et al. (2015)	Dynamic	Adaptive - OLS	Single - Closest weight	Transfer	Periodic	$n\times$ Tabular	Scalarized POMDP solver	/	/	Policy following
Mossalam et al. (2016)	Dynamic	Adaptive - OLS	Single - Closest weight	Transfer	Periodic	$n\times$ DNN + MO reg.	Scalarized DQN	Indep.	Recency + Uniform	Policy following
Chen et al. (2020)	Static	Manual	All	Shared buffer Shared layers	Continuous	$n\times$ DNN	Scalarized SAC	All	Recency + Uniform	Parallel policy following
Yang et al. (2019)	Dynamic	Random	All	CR	Continuous	$1\times$ DNN	Envelope DQN	All	HER + Recency + Uniform	Policy following
Xu et al. (2020a)	Dynamic	Uniform	None	None	None	$n\times$ DNN + MO reg.	Scalarized PPO	Indep.	Recency + Uniform	Policy following
Abels et al. (2019)	Dynamic	Random	All	CR	Continuous	$1\times$ DNN + MO reg.	Scalarized, Multi-weights DQN	All	HER + PER (Diversity)	Policy following
Alegre et al. (2023)	Dynamic	Adaptive - GPI-LS	All	CR Shared model	Continuous	$1\times$ DNN + MO reg.	Scalarized, Multi-weights DQN or TD3	All	HER + PER (GPI)	Policy following
Castelletti et al. (2013)	Dynamic	Random	All	CR	Continuous	$1\times$ Trees	Scalarized FQI	/	/	Historical dataset

Table 3: A non-exhaustive list of MORL works classified according to the MORL/D taxonomy. OLS = Optimistic Linear Support, DNN = Deep Neural Network, POMDP = Partially Observable MDP, MO reg. = Multi-objective regression, CR = Conditioned Regression, HER = Hindsight Experience Replay, PER = prioritized experience replay, GPI = Generalized Policy Improvement.

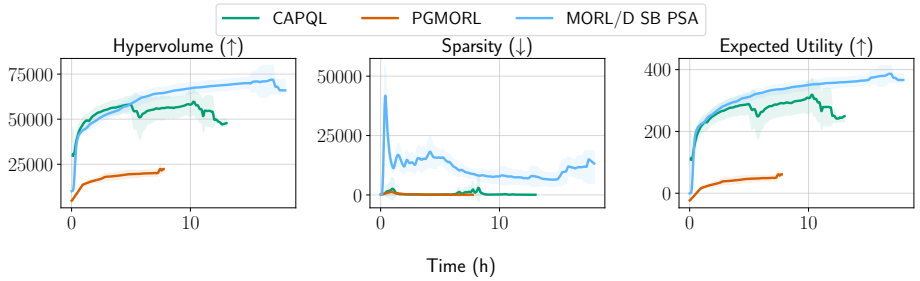


Figure 15: Comparisons in terms of runtime for the *mo-halfcheetah-v4*.

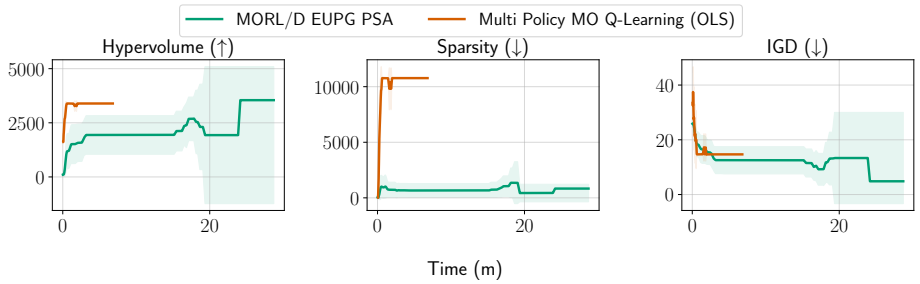


Figure 16: Comparisons in terms of runtime for the *deep-sea-treasure-concave-v0*.

vertical line showing again the boundaries between traits inherited from MOO and RL. For each trait, a column presents the instantiation choice made in each paper.

For instance, in the first line, the work of Roijers and Whiteson dynamically assigns weight vectors based on Optimistic Linear Support (OLS). As a cooperation mechanism, the algorithm reuses (transfers) the knowledge from the closest already trained policy to hot-start the training of a new policy. The algorithm relies on n tabular representations, where n is the number of desired policies. The Bellman update used is based on a Partially Observable MDP solver (POMDP). Finally, the sampling strategy proposes to follow the policy currently being trained (with its internal exploration technique).

All the works referenced in the table make use of the weighted sum scalarization. Thus, scalarization and reference point columns have been omitted from the table since they would bring little information. In general, non-linear scalarization schemes are understudied when compared to the weighted sum. Additionally, for space reasons, population selection and archive have not been represented either. In both traits, the work of Xu et al. (2020a) is particularly interesting, as explained in Section 5.

Appendix C. Additional Results

Figures 15 and 16 give results in terms of runtime of algorithms. Similar conclusions to what has been discussed in the main paper can be drawn here too.

References

Abels, A., Roijers, D., Lenaerts, T., Nowé, A., & Steckelmacher, D. (2019). Dynamic Weights in Multi-Objective Deep Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 11–20. PMLR. ISSN: 2640-3498.

Alaya, I., Solnon, C., & Ghedira, K. (2007). Ant Colony Optimization for Multi-objective Optimization Problems. In *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 450–457, Patras, Greece. IEEE Computer Society.

Alegre, L. N., Bazzan, A. L. C., Roijers, D. M., & Nowé, A. (2023). Sample-Efficient Multi-Objective Learning via Generalized Policy Improvement Prioritization. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems*.

Alegre, L. N., Felten, F., Talbi, E.-G., Danoy, G., Nowé, A., Bazzan, A. L., & da Silva, B. C. (2022). MO-Gym: A Library of Multi-Objective Reinforcement Learning Environments. In *Proceedings of the 34th Benelux Conference on Artificial Intelligence BNAIC/Benelearn*.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., & Zaremba, W. (2017). Hindsight experience replay. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 5055–5065, Red Hook, NY, USA. Curran Associates Inc.

Blank, J., & Deb, K. (2020). pymoo: Multi-Objective Optimization in Python. *IEEE Access*, *8*, 89497–89509.

Blank, J., Deb, K., Dhebar, Y., Bandaru, S., & Seada, H. (2021). Generating Well-Spaced Points on a Unit Simplex for Evolutionary Many-Objective Optimization. *IEEE Transactions on Evolutionary Computation*, *25*(1), 48–60. Conference Name: IEEE Transactions on Evolutionary Computation.

Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., & Qu, R. (2013). Hyper-heuristics: a survey of the state of the art. *Journal of the Operational Research Society*, *64*(12), 1695–1724.

Castelletti, A., Pianosi, F., & Restelli, M. (2013). A multiobjective reinforcement learning approach to water resources systems operation: Pareto frontier approximation in a single run. *Water Resources Research*, *49*(6), 3476–3486. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/wrcr.20295.

Chen, D., Wang, Y., & Gao, W. (2020). Combining a gradient-based method and an evolution strategy for multi-objective reinforcement learning. *Applied Intelligence*, *50*.

Coello Coello, C. A., & Reyes Sierra, M. (2004). A Study of the Parallelization of a Coevolutionary Multi-objective Evolutionary Algorithm. In Monroy, R., Arroyo-Figueroa, G., Sucar, L. E., & Sossa, H. (Eds.), *MICAI 2004: Advances in Artificial Intelligence*, Lecture Notes in Computer Science, pp. 688–697, Berlin, Heidelberg. Springer.

Czyżżak, P., & Jaszkiewicz, A. (1998). Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, *7*(1), 34–47.

Das, I., & Dennis, J. (2000). Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems. *SIAM Journal on Optimization*, *8*.

de Bruin, T., Kober, J., Tuyls, K., & Babuska, R. (2016). Improved deep reinforcement learning for robotics through distribution-based experience retention. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3947–3952, Daejeon, South Korea. IEEE.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197. Conference Name: IEEE Transactions on Evolutionary Computation.

Dubois-Lacoste, J., López-Ibáñez, M., & Stützle, T. (2011). Improving the anytime behavior of two-phase local search. *Annals of Mathematics and Artificial Intelligence*, *61*(2), 125–154.

Eimer, T., Lindauer, M., & Raileanu, R. (2023). Hyperparameters in Reinforcement Learning and How To Tune Them. In *Proceedings of the 40th International Conference on Machine Learning (ICML 2023)*.

Emmerich, M. T. M., & Deutz, A. H. (2018). A tutorial on multiobjective optimization: fundamentals and evolutionary methods. *Natural Computing*, *17*(3), 585–609.

Felten, F., Alegre, L. N., Nowe, A., Bazzan, A. L. C., Talbi, E. G., Danoy, G., & Silva, B. C. d. (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)*.

Felten, F., Danoy, G., Talbi, E.-G., & Bouvry, P. (2022). Metaheuristics-based Exploration Strategies for Multi-Objective Reinforcement Learning:. In *Proceedings of the 14th International Conference on Agents and Artificial Intelligence*, pp. 662–673. SCITEPRESS - Science and Technology Publications.

Felten, F., Gareev, D., Talbi, E.-G., & Danoy, G. (2023). Hyperparameter Optimization for Multi-Objective Reinforcement Learning.. arXiv:2310.16487 [cs].

Felten, F., Talbi, E.-G., & Danoy, G. (2022). MORL/D: Multi-Objective Reinforcement Learning based on Decomposition. In *International Conference in Optimization and Learning (OLA2022)*.

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1861–1870. PMLR. ISSN: 2640-3498.

Hansen, M. (2000). Tabu search for multiobjective combinatorial optimization: TAMOCO. *Control and Cybernetics*, *29*.

Hayes, C., Rădulescu, R., Bargiacchi, E., Källström, J., Macfarlane, M., Reymond, M., Verstraeten, T., Zintgraf, L., Dazeley, R., Heintz, F., Howley, E., Irissappane, A., Mannion, P., Nowe, A., Ramos, G., Restelli, M., Vamplew, P., & Roijers, D. (2022). A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, *36*.

Huang, S., Dossa, R. F. J., Ye, C., Braga, J., Chakraborty, D., Mehta, K., & Araújo, J. G. M. (2022). CleanRL: High-quality Single-file Implementations of Deep Reinforcement Learning Algorithms. *Journal of Machine Learning Research*, *23*(274), 1–18.

Huang, S., Gallouédec, Q., Felten, F., Raffin, A., Dossa, R. F. J., Zhao, Y., Sullivan, R., Makoviychuk, V., Makoviichuk, D., Danesh, M. H., Roumégous, C., Weng, J., Chen, C., Rahman, M. M., Araújo, J. G. M., Quan, G., Tan, D., Klein, T., Charakorn, R., Towers, M., Berthelot, Y., Mehta, K., Chakraborty, D., KG, A., Charraut, V., Ye, C., Liu, Z., Alegre, L. N., Nikulin, A., Hu, X., Liu, T., Choi, J., & Yi, B. (2024). Open RL Benchmark: Comprehensive Tracked Experiments for Reinforcement Learning.. arXiv:2402.03046 [cs].

Ishibuchi, H., & Kaige, S. (2004). Implementation of Simple Multiobjective Memetic Algorithms and Its Application to Knapsack Problems. *International Journal of Hybrid Intelligent Systems - IJHIS*, *1*.

Ishibuchi, H., Sakane, Y., Tsukamoto, N., & Nojima, Y. (2010). Simultaneous Use of Different Scalarizing Functions in MOEA/D. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, GECCO '10, pp. 519–526, New York, NY, USA. Association for Computing Machinery. event-place: Portland, Oregon, USA.

Ke, L., Zhang, Q., & Battiti, R. (2013). MOEA/D-ACO: A Multiobjective Evolutionary Algorithm Using Decomposition and AntColony. *IEEE Transactions on Cybernetics*, *43*(6), 1845–1859. Conference Name: IEEE Transactions on Cybernetics.

Liu, Y., Ishibuchi, H., Masuyama, N., & Nojima, Y. (2020). Adapting Reference Vectors and Scalarizing Functions by Growing Neural Gas to Handle Irregular Pareto Fronts. *IEEE Transactions on Evolutionary Computation*, *24*(3), 439–453. Conference Name: IEEE Transactions on Evolutionary Computation.

Lu, H., Herman, D., & Yu, Y. (2023). Multi-Objective Reinforcement Learning: Convexity, Stationarity and Pareto Optimality. In *The Eleventh International Conference on Learning Representations*.

Marler, R., & Arora, J. (2010). The weighted sum method for multi-objective optimization: New insights. *Structural and Multidisciplinary Optimization*, *41*, 853–862.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.

Moerland, T. M., Broekens, J., Plaat, A., & Jonker, C. M. (2023). Model-based Reinforcement Learning: A Survey. *Foundations and Trends® in Machine Learning*, *16*(1), 1–118.

Mossalam, H., Assael, Y. M., Roijers, D. M., & Whiteson, S. (2016). Multi-Objective Deep Reinforcement Learning. *CoRR*, *abs/1610.02707*. arXiv: 1610.02707.

Murata, T., Ishibuchi, H., & Gen, M. (2001). Specification of Genetic Search Directions in Cellular Multi-objective Genetic Algorithms. In Zitzler, E., Thiele, L., Deb, K.,

Coello Coello, C. A., & Corne, D. (Eds.), *Evolutionary Multi-Criterion Optimization*, Lecture Notes in Computer Science, pp. 82–95, Berlin, Heidelberg. Springer.

Natarajan, S., & Tadepalli, P. (2005). *Dynamic preferences in multi-criteria reinforcement learning*. Association for Computing Machinery. Pages: 608.

Parker-Holder, J., Rajan, R., Song, X., Biedenkapp, A., Miao, Y., Eimer, T., Zhang, B., Nguyen, V., Calandra, R., Faust, A., Hutter, F., & Lindauer, M. (2022). Automated Reinforcement Learning (AutoRL): A Survey and Open Problems. *Journal of Artificial Intelligence Research*, *74*.

Qi, Y., Ma, X., Liu, F., Jiao, L., Sun, J., & Wu, J. (2013). MOEA/D with adaptive weight adjustment. *Evolutionary computation*, *22*.

Reymond, M., & Nowe, A. (2019). Pareto-DQN: Approximating the Pareto front in complex multi-objective decision problems. In *Proceedings of the Adaptive and Learning Agents Workshop 2019 (ALA-19) at AAMAS*.

Roijers, D., Steckelmacher, D., & Nowe, A. (2018). Multi-objective Reinforcement Learning for the Expected Utility of the Return. In *Proceedings of the ALA workshop at ICML/AAMAS/IJCAI 2018*.

Roijers, D., Whiteson, S., & Oliehoek, F. (2015). Computing Convex Coverage Sets for Faster Multi-objective Coordination. *Journal of Artificial Intelligence Research*, *52*, 399–443.

Roijers, D. M., & Whiteson, S. (2017). Multi-Objective Decision Making. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, *11*(1), 1–129. Publisher: Morgan & Claypool Publishers.

Roijers, D. M., Whiteson, S., & Oliehoek, F. A. (2015). Point-Based Planning for Multi-Objective POMDPs. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pp. 1666–1672. AAAI Press. Place: Buenos Aires, Argentina.

Roijers, D. M., Vamplew, P., Whiteson, S., & Dazeley, R. (2013). A Survey of Multi-Objective Sequential Decision-Making. *Journal of Artificial Intelligence Research*, *48*, 67–113. arXiv: 1402.0590.

Ruiz-Montiel, M., Mandow, L., & Pérez-de-la Cruz, J.-L. (2017). A temporal difference method for multi-objective reinforcement learning. *Neurocomputing*, *263*, 15–25.

Régin, J.-C., Rezgui, M., & Malapert, A. (2013). Embarrassingly Parallel Search. In Schulte, C. (Ed.), *Principles and Practice of Constraint Programming*, Lecture Notes in Computer Science, pp. 596–610, Berlin, Heidelberg. Springer.

Röpke, W., Hayes, C. F., Mannion, P., Howley, E., Nowé, A., & Roijers, D. M. (2023). Distributional Multi-Objective Decision Making. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pp. 5711–5719, Macau, SAR China. International Joint Conferences on Artificial Intelligence Organization.

Santiago, A., Huacuja, H. J. F., Dorronsoro, B., Pecero, J. E., Santillan, C. G., Barbosa, J. J. G., & Monterrubio, J. C. S. (2014). A Survey of Decomposition Methods for Multi-objective Optimization. In Castillo, O., Melin, P., Pedrycz, W., & Kacprzyk, J. (Eds.),

Recent Advances on Hybrid Approaches for Designing Intelligent Systems, Studies in Computational Intelligence, pp. 453–465. Springer International Publishing, Cham.

Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2016). Prioritized Experience Replay. In Bengio, Y., & LeCun, Y. (Eds.), *Proceedings of the 4th International Conference on Learning Representations, (ICLR 2016)*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms.. *CoRR*, *abs/1707.06347*.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489. Bandiera_abtest: a Cg_type: Nature Research Journals Number: 7587 Primary_atype: Research Publisher: Nature Publishing Group Subject_term: Computational science;Computer science;Reward Subject_term_id: computational-science;computer-science;reward.

Stanley, K. O., Clune, J., Lehman, J., & Miikkulainen, R. (2019). Designing neural networks through neuroevolution. *Nature Machine Intelligence*, *1*(1), 24–35. Number: 1 Publisher: Nature Publishing Group.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2 edition). Adaptive Computation and Machine Learning series. A Bradford Book, Cambridge, MA, USA.

Talbi, E.-G. (2009). *Metaheuristics: From Design to Implementation*. Wiley Publishing.

Todorov, E., Erez, T., & Tassa, Y. (2012). MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. ISSN: 2153-0866.

Vamplew, P., Dazeley, R., Barker, E., & Kelarev, A. (2009). Constructing Stochastic Mixture Policies for Episodic Multiobjective Reinforcement Learning Tasks. In Nicholson, A., & Li, X. (Eds.), *AI 2009: Advances in Artificial Intelligence*, Lecture Notes in Computer Science, pp. 340–349, Berlin, Heidelberg. Springer.

Vamplew, P., Dazeley, R., Berry, A., Issabekov, R., & Dekker, E. (2011). Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine Learning*, *84*(1), 51–80.

Vamplew, P., Smith, B. J., Källström, J., Ramos, G., Rădulescu, R., Roijers, D. M., Hayes, C. F., Heintz, F., Mannion, P., Libin, P. J. K., Dazeley, R., & Foale, C. (2022). Scalar reward is not enough: a response to Silver, Singh, Precup and Sutton (2021). *Autonomous Agents and Multi-Agent Systems*, *36*(2), 41.

Van Moffaert, K., Drugan, M. M., & Nowe, A. (2013). Scalarized multi-objective reinforcement learning: Novel design techniques. In *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pp. 191–199, Singapore, Singapore. IEEE.

Van Moffaert, K., & Nowé, A. (2014). Multi-objective reinforcement learning using sets of pareto dominating policies. *The Journal of Machine Learning Research*, *15*(1), 3483–3512. Publisher: JMLR. org.

Varrette, S., Bouvry, P., Cartiaux, H., & Georgatos, F. (2014). Management of an academic HPC cluster: The UL experience. In *2014 International Conference on High Performance Computing & Simulation (HPCS)*, pp. 959–967. IEEE.

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*(3), 279–292.

Wiering, M. A., Withagen, M., & Drugan, M. M. (2014). Model-based multi-objective reinforcement learning. In *2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pp. 1–6, Orlando, FL, USA. IEEE.

Wolpert, D., & Macready, W. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, *1*(1), 67–82. Conference Name: IEEE Transactions on Evolutionary Computation.

Wortsman, M., Ilharco, G., Gadre, S. Y., Roelofs, R., Gontijo-Lopes, R., Morcos, A. S., Namkoong, H., Farhadi, A., Carmon, Y., Kornblith, S., & Schmidt, L. (2022). Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 23965–23998. PMLR. ISSN: 2640-3498.

Wurman, P. R., Barrett, S., Kawamoto, K., MacGlashan, J., Subramanian, K., Walsh, T. J., Capobianco, R., Devlic, A., Eckert, F., Fuchs, F., Gilpin, L., Khandelwal, P., Kompella, V., Lin, H., MacAlpine, P., Oller, D., Seno, T., Sherstan, C., Thomure, M. D., Aghabozorgi, H., Barrett, L., Douglas, R., Whitehead, D., Dürr, P., Stone, P., Spranger, M., & Kitano, H. (2022). Outracing champion Gran Turismo drivers with deep reinforcement learning. *Nature*, *602*(7896), 223–228. Number: 7896 Publisher: Nature Publishing Group.

Xu, J., Tian, Y., Ma, P., Rus, D., Sueda, S., & Matusik, W. (2020a). Prediction-Guided Multi-Objective Reinforcement Learning for Continuous Robot Control. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 10607–10616. PMLR. ISSN: 2640-3498.

Xu, Q., Xu, Z., & Ma, T. (2020b). A Survey of Multiobjective Evolutionary Algorithms Based on Decomposition: Variants, Challenges and Future Directions. *IEEE Access*, *8*, 41588–41614. Conference Name: IEEE Access.

Yang, R., Sun, X., & Narasimhan, K. (2019). A Generalized Algorithm for Multi-Objective Reinforcement Learning and Policy Adaptation. In *Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc.

Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, Y., Bayen, A., & Wu, Y. (2022). The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Zhang, Q., & Li, H. (2007). MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation*, *11*(6), 712–731. Conference Name: IEEE Transactions on Evolutionary Computation.

Zintgraf, L. M., Kanters, T. V., Roijers, D. M., Oliehoek, F. A., & Beau, P. (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.*

Zitzler, E. (1999). Evolutionary algorithms for multiobjective optimization: methods and applications. In *Ph.D. Dissertation. ETH Zurich, Switzerland.*