A Distributional View on Multi-Objective Policy Optimization

Abbas Abdolmaleki [*1] **Sandy H. Huang** [*1] **Leonard Hasenclever** [1] **Michael Neunert** [1] **H. Francis Song** [1] **Martina Zambelli** [1] **Murilo F. Martins** [1] **Nicolas Heess** [1] **Raia Hadsell** [1] **Martin Riedmiller** [1]

Abstract

Many real-world problems require trading off multiple competing objectives. However, these objectives are often in different units and/or scales, which can make it challenging for practitioners to express numerical preferences over objectives in their native units. In this paper we propose a novel algorithm for multi-objective reinforcement learning that enables setting desired preferences for objectives in a scale-invariant way. We propose to learn an action distribution for each objective, and we use supervised learning to fit a parametric policy to a combination of these distributions. We demonstrate the effectiveness of our approach on challenging high-dimensional real and simulated robotics tasks, and show that setting different preferences in our framework allows us to trace out the space of nondominated solutions.

1. Introduction

Reinforcement learning (RL) algorithms do an excellent job at training policies to optimize a single scalar reward function. Recent advances in deep RL have made it possible to train policies that exceed human-level performance on Atari (Mnih et al., 2015) and Go (Silver et al., 2016), perform complex robotic manipulation tasks (Zeng et al., 2019), learn agile locomotion (Tan et al., 2018), and even obtain reward in unanticipated ways (Amodei et al., 2016).

However, many real-world tasks involve *multiple*, possibly competing, objectives. For instance, choosing a financial portfolio requires trading off between risk and return; controlling energy systems requires trading off performance and cost; and autonomous cars must trade off fuel costs, efficiency, and safety. Multi-objective reinforcement learning (MORL) algorithms aim to tackle such problems (Roijers et al., 2013; Liu et al., 2015). A common approach is *scalarization*: based on *preferences* across objectives, transform the multi-objective reward vector into a single scalar reward (e.g., by taking a convex combination), and then use standard RL to optimize this scalar reward.

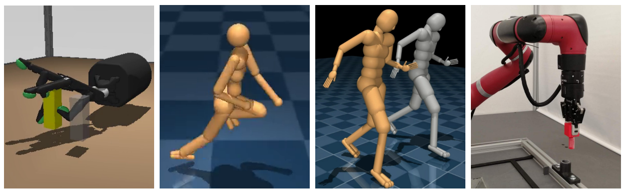
Figure 1. We demonstrate our approach in four complex continuous control domains, in simulation and in the real world. Videos are at http://sites.google.com/view/mo-mpo.

It is tricky, though, for practitioners to pick the appropriate scalarization for a desired preference across objectives, because often objectives are defined in different units and/or scales. For instance, suppose we want an agent to complete a task while minimizing energy usage and mechanical wear-and-tear. Task completion may correspond to a sparse reward or to the number of square feet a vacuuming robot has cleaned, and reducing energy usage and mechanical wear-and-tear could be enforced by penalties on power consumption (in kWh) and actuator efforts (in N or Nm), respectively. Practitioners would need to resort to using trial and error to select a scalarization that ensures the agent prioritizes actually doing the task (and thus being useful) over saving energy.

Motivated by this, we propose a scale-invariant approach for encoding preferences, derived from the RL-as-inference perspective. Instead of choosing a scalarization, practitioners set a constraint per objective. Based on these constraints, we learn an action distribution per objective that improves on the current policy. Then, to obtain a single updated policy that makes these trade-offs, we use supervised learning to fit a policy to the combination of these action distributions. The constraints control the influence of each objective on the policy, by constraining the KL-divergence between each objective-specific distribution and the current policy. The higher the constraint value, the more influence the objective has. Thus, a desired preference over objectives can be encoded as the relative magnitude of these constraint values.

[*]Equal contribution [1]DeepMind. Correspondence to: Abbas Abdolmaleki <aabdolmaleki@google.com>, Sandy H. Huang <shhuang@google.com>.

Proceedings of the 37th International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

Fundamentally, scalarization combines objectives in reward space, whereas our approach combines objectives in *distribution* space, thus making it invariant to the scale of rewards. In principle, our approach can be combined with any RL algorithm, regardless of whether it is off-policy or on-policy. We combine it with maximum a posteriori policy optimization (MPO) (Abdolmaleki et al., 2018a;b), an off-policy actor-critic RL algorithm, and V-MPO (Song et al., 2020), an on-policy variant of MPO. We call these two algorithms multi-objective MPO (MO-MPO) and multi-objective V-MPO (MO-V-MPO), respectively. Code for MO-MPO will be made available online.[1]

Our main contribution is providing a distributional view on MORL, which enables scale-invariant encoding of preferences. We show that this is a theoretically-grounded approach, that arises from taking an RL-as-inference perspective of MORL. Empirically, we analyze the mechanics of MO-MPO and show it finds all Pareto-optimal policies in a popular MORL benchmark task. Finally, we demonstrate that MO-MPO and MO-V-MPO outperform scalarized approaches on multi-objective tasks across several challenging high-dimensional continuous control domains (Fig. 1).

2. Related Work

2.1. Multi-Objective Reinforcement Learning

Multi-objective reinforcement learning (MORL) algorithms are either *single-policy* or *multiple-policy* (Vamplew et al., 2011). Single-policy approaches seek to find the optimal policy for a given scalarization of the multi-objective problem. Often this scalarization is linear, but other choices have also been explored (Van Moffaert et al., 2013).

However, the scalarization may be unknown at training time, or it may change over time. *Multiple-policy* approaches handle this by finding a set of policies that approximates the true Pareto front. Some approaches repeatedly call a single-policy MORL algorithm with strategically-chosen scalarizations (Natarajan & Tadepalli, 2005; Roijers et al., 2014; Mossalam et al., 2016; Zuluaga et al., 2016). Other approaches learn a set of policies simultaneously, by using a multi-objective variant of the Q-learning update rule (Barrett & Narayanan, 2008; Moffaert & Nowé, 2014; Reymond & Nowé, 2019; Yang et al., 2019) or by modifying gradient-based policy search (Parisi et al., 2014; Pirotta et al., 2015).

Most existing approaches for finding the Pareto front are limited to discrete state and action spaces, in which tabular algorithms are sufficient. Although recent work combining MORL with deep RL handles high-dimensional observations, this is in domains with low-dimensional and usually discrete action spaces (Mossalam et al., 2016; van Seijen et al., 2017; Friedman & Fontaine, 2018; Abels et al., 2019; Reymond & Nowé, 2019; Yang et al., 2019; Nottingham et al., 2019). In contrast, we evaluate our approach on continuous control tasks with more than 20 action dimensions.[2]

A couple of recent works have applied deep MORL to find the Pareto front in continuous control tasks; these works assume scalarization and rely on additionally learning either a meta-policy (Chen et al., 2019) or inter-objective relationships (Zhan & Cao, 2019). We take an orthogonal approach to existing approaches: one encodes preferences via constraints on the influence of each objective on the policy update, instead of via scalarization. MO-MPO can be run multiple times, with different constraint settings, to find a Pareto front of policies.

2.2. Constrained Reinforcement Learning

An alternate way of setting preferences is to enforce that policies meet certain constraints. For instance, threshold lexicographic ordering approaches optimize a (single) objective while meeting specified threshold values on the other objectives (Gábor et al., 1998), optionally with slack (Wray et al., 2015). Similarly, safe RL is concerned with learning policies that optimize a scalar reward while not violating safety constraints (Achiam et al., 2017; Chow et al., 2018); this has also been studied in the off-policy batch RL setting (Le et al., 2019). Related work minimizes costs while ensuring the policy meets a constraint on the minimum expected return (Bohez et al., 2019), but this requires that the desired or achievable reward is known a priori. In contrast, MO-MPO does not require knowledge of the scale of rewards. In fact, often there is no easy way to specify constraints on objectives, e.g., it is difficult to figure out a priori how much actuator effort a robot will need to use to perform a task.

2.3. Multi-Task Reinforcement Learning

Multi-task reinforcement learning can also be cast as a MORL problem. Generally these algorithms learn a separate policy for each task, with shared learning across tasks (Teh et al., 2017; Riedmiller et al., 2018; Wulfmeier et al., 2019). In particular, Distral (Teh et al., 2017) learns a shared prior that regularizes the per-task policies to be similar to each other, and thus captures essential structure that is shared across tasks. MO-MPO differs in that the goal is to learn a *single* policy that must *trade off* across different objectives.

Other multi-task RL algorithms seek to train a single agent

[1]http://github.com/deepmind/acme

[2]MORL for continuous control tasks is difficult because the policy can no longer output arbitrary action distributions, which limits how well it can compromise between competing objectives. In state-of-the-art RL algorithms for continuous control, policies typically output a single action (e.g., D4PG (Barth-Maron et al., 2018)) or a Gaussian (e.g., PPO (Schulman et al., 2017), MPO (Abdolmaleki et al., 2018b), and SAC (Haarnoja et al., 2018)).

to solve different tasks, and thus need to handle different reward scales across tasks. Prior work uses adaptive normalization for the targets in value-based RL, so that the agent cares equally about all tasks (van Hasselt et al., 2016; Hessel et al., 2019). Similarly, prior work in multi-objective optimization has dealt with objectives of different units and/or scales by normalizing objectives to have similar magnitudes (Marler & Arora, 2005; Grodzevich & Romanko, 2006; Kim & de Weck, 2006; Daneshmand et al., 2017; Ishibuchi et al., 2017). MO-MPO can also be seen as doing adaptive normalization, but for *any* preference over objectives, not just equal preferences.

In general, invariance to reparameterization of the function approximator has been investigated in optimization literature resulting in, for example, natural gradient methods (Martens, 2014). The common tool here is measuring distances in function space instead of parameter space, using KL-divergence. Similarly in this work, to achieve invariance to the scale of objectives, we use KL-divergence over policies to encode preferences.

3. Background and Notation

Multi Objective Markov Decision Process. In this paper, we consider a multi-objective RL problem defined by a multi-objective Markov Decision Process (MO-MDP). The MO-MDP consists of states $s \in \mathcal{S}$ and actions $a \in \mathcal{A}$, an initial state distribution $p(s_0)$, transition probabilities $p(s_{t+1}|s_t, a_t)$ which define the probability of changing from state s_t to s_{t+1} when taking action a_t, reward functions $\{r_k(s,a) \in \mathbb{R}\}_{k=1}^N$ per objective k, and a discount factor $\gamma \in [0, 1)$. We define our policy $\pi_\theta(a|s)$ as a state conditional distribution over actions parametrized by θ. Together with the transition probabilities, this gives rise to a state visitation distribution $\mu(s)$. We also consider per-objective action-value functions. The action-value function for objective k is defined as the expected return (i.e., cumulative discounted reward) from choosing action a in state s for objective k and then following policy π: $Q_k^\pi(s,a) = \mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t r_k(s_t,a_t)|s_0 = s, a_0 = a]$. We can represent this function using the recursive expression $Q_k^\pi(s_t,a_t) = \mathbb{E}_{p(s_{t+1}|s_t,a_t)}\left[r_k(s_t,a_t) + \gamma V_k^\pi(s_{t+1})\right]$, where $V_k^\pi(s) = \mathbb{E}_\pi[Q_k^\pi(s,a)]$ is the value function of π for objective k.

Problem Statement. For any MO-MDP there is a set of nondominated policies, i.e., the Pareto front. A policy is nondominated if there is no other policy that improves its expected return for an objective without reducing the expected return of at least one other objective. Given a preference setting, our goal is to find a nondominated policy π_θ that satisfies those preferences. In our approach, a setting of constraints does not directly correspond to a particular scalarization, but we show that by varying these constraint settings, we can indeed trace out a Pareto front of policies.

Algorithm 1 MO-MPO: One policy improvement step

```
 1: given batch size ($L$), number of actions to sample ($M$), ($N$) Q-functions $\{Q_k^{\pi_\text{old}}(s,a)\}_{k=1}^N$, preferences $\{\epsilon_k\}_{k=1}^N$, previous policy $\pi_\text{old}$, previous temperatures $\{\eta_k\}_{k=1}^N$, replay buffer $\mathcal{D}$, first-order gradient-based optimizer $\mathcal{O}$
 2:
 3: initialize $\pi_\theta$ from the parameters of $\pi_\text{old}$
 4: repeat
 5:   // Collect dataset $\{s^i, a^{ij}, Q_k^{ij}\}_{i,j,k}^{L,M,N}$, where
 6:   // $M$ actions $a^{ij} \sim \pi_\text{old}(a|s^i)$ and $Q_k^{ij} = Q_k^{\pi_\text{old}}(s^i, a^{ij})$
 7:
 8:   // Compute action distribution for each objective
 9:   for k = 1, ..., $N$ do
10:     $\delta_{\eta_k} \leftarrow \nabla_{\eta_k} \eta_k \epsilon_k + \eta_k \sum_i^L \frac{1}{L} \log\left(\sum_j^M \frac{1}{M} \exp\left(\frac{Q_k^{ij}}{\eta_k}\right)\right)$
11:     Update $\eta_k$ based on $\delta_{\eta_k}$, using optimizer $\mathcal{O}$
12:     $q_k^{ij} \propto \exp(\frac{Q_k^{ij}}{\eta_k})$
13:   end for
14:
15:   // Update parametric policy
16:   $\delta_\pi \leftarrow -\nabla_\theta \sum_i^L \sum_j^M \sum_k^N q_k^{ij} \log \pi_\theta(a^{ij}|s^i)$
17:     (subject to additional KL regularization, see Sec. 4.2.2)
18:   Update $\pi_\theta$ based on $\delta_\pi$, using optimizer $\mathcal{O}$
19:
20: until fixed number of steps
21: return $\pi_\text{old} = \pi_\theta$
```

4. Method

We propose a policy iteration algorithm for multi-objective RL. Policy iteration algorithms decompose the RL problem into two sub-problems and iterate until convergence:

1. *Policy evaluation*: estimate Q-functions given policy
2. *Policy improvement*: update policy given Q-functions

Algorithm 1 summarizes this two-step multi-objective policy improvement procedure. In Appendix E, we explain how this can be derived from the "RL as inference" perspective.

We describe multi-objective MPO in this section and explain multi-objective V-MPO in Appendix D. When there is only one objective, MO-(V-)MPO reduces to (V-)MPO.

4.1. Multi-Objective Policy Evaluation

In this step we learn Q-functions to evaluate the previous policy π_old. We train a separate Q-function per objective, following the Q-decomposition approach (Russell & Zimdars, 2003). In principle, any Q-learning algorithm can be used, as long as the target Q-value is computed with respect to π_old.[3] In this paper, we use the Retrace objective (Munos

[3]Russell & Zimdars (2003) prove critics suffer from "illusion of control" if they are trained with conventional Q-learning (Watkins, 1989). In other words, if each critic computes its target Q-value

et al., 2016) to learn a Q-function $Q_k^{\pi_{\text{old}}}(s,a;\phi_k)$ for each objective k, parameterized by ϕ_k, as follows:

$$\min_{\{\phi_k\}_1^N} \sum_{k=1}^{N} \mathbb{E}_{(s,a)\sim\mathcal{D}}\Big[\big(\hat{Q}_k^{\text{ret}}(s,a) - Q_k^{\pi_{\text{old}}}(s,a;\phi_k)\big)^2\Big],$$

where $\hat{Q}_k^{\text{ret}}$ is the Retrace target for objective k and the previous policy π_{old}, and $\mathcal{D}$ is a replay buffer containing gathered transitions. See Appendix C for details.

4.2. Multi-Objective Policy Improvement

Given the previous policy $\pi_{\text{old}}(a|s)$ and associated Q-functions $\{Q_k^{\pi_{\text{old}}}(s,a)\}_{k=1}^N$, our goal is to improve the previous policy for a given visitation distribution $\mu(s)$.[4] To this end, we learn an action distribution for each Q-function and combine these to obtain the next policy $\pi_{\text{new}}(a|s)$. This is a multi-objective variant of the two-step policy improvement procedure employed by MPO (Abdolmaleki et al., 2018b).

In the first step, for each objective k we learn an improved action distribution $q_k(a|s)$ such that $\mathbb{E}_{q_k(a|s)}[Q_k^{\pi_{\text{old}}}(s,a)] \geq \mathbb{E}_{\pi_{\text{old}}(a|s)}[Q_k^{\pi_{\text{old}}}(s,a)]$, where states s are drawn from a visitation distribution $\mu(s)$.

In the second step, we combine and distill the improved distributions q_k into a new parametric policy π_{new} (with parameters θ_{new}) by minimizing the KL-divergence between the distributions and the new parametric policy, i.e,

$$\theta_{\text{new}} = \operatorname*{argmin}_{\theta} \sum_{k=1}^{N} \mathbb{E}_{\mu(s)}\Big[\mathrm{KL}\Big(q_k(a|s)\|\pi_\theta(a|s)\Big)\Big]. \quad (1)$$

This is a supervised learning loss that performs maximum likelihood estimate of distributions q_k. Next, we will explain these two steps in more detail.

4.2.1. Obtaining action distributions per objective (Step 1)

To obtain the per-objective improved action distributions $q_k(a|s)$, we optimize the standard RL objective for *each objective* Q_k:

$$\max_{q_k} \int_s \mu(s) \int_a q_k(a|s)\, Q_k(s,a)\, \mathrm{d}a\, \mathrm{d}s \quad (2)$$
$$\text{s.t.} \int_s \mu(s)\, \mathrm{KL}(q_k(a|s)\|\pi_{\text{old}}(a|s))\, \mathrm{d}s < \epsilon_k\,,$$

where ϵ_k denotes the allowed expected KL divergence for objective k. We use these ϵ_k to encode preferences over objectives. More concretely, ϵ_k defines the allowed influence of objective k on the change of the policy.

based on its own best action for the next state, then they overestimate Q-values, because in reality the parametric policy π_θ (that considers all critics' opinions) is in charge of choosing actions.

[4]In practice, we use draws from the replay buffer to estimate expectations over the visitation distribution $\mu(s)$.

For nonparametric action distributions $q_k(a|s)$, we can solve this constrained optimization problem in closed form for each state s sampled from $\mu(s)$ (Abdolmaleki et al., 2018b),

$$q_k(a|s) \propto \pi_{\text{old}}(a|s) \exp\Big(\frac{Q_k(s,a)}{\eta_k}\Big), \quad (3)$$

where the temperature η_k is computed based on the corresponding ϵ_k, by solving the following convex dual function:

$$\eta_k = \operatorname*{argmin}_{\eta}\, \eta\,\epsilon_k + \quad (4)$$
$$\eta \int_s \mu(s) \log \int_a \pi_{\text{old}}(a|s) \exp\Big(\frac{Q_k(s,a)}{\eta}\Big)\, \mathrm{d}a\, \mathrm{d}s\,.$$

In order to evaluate $q_k(a|s)$ and the integrals in (4), we draw L states from the replay buffer and, for each state, sample M actions from the current policy π_{old}. In practice, we maintain one temperature parameter η_k per objective. We found that optimizing the dual function by performing a few steps of gradient descent on η_k is effective, and we initialize with the solution found in the previous policy iteration step. Since η_k should be positive, we use a projection operator after each gradient step to maintain $\eta_k > 0$. Please refer to Appendix C for derivation details.

Application to Other Deep RL Algorithms. Since the constraints ϵ_k in (2) encode the preferences over objectives, solving this optimization problem with good satisfaction of constraints is key for learning a policy that satisfies the desired preferences. For nonparametric action distributions $q_k(a|s)$, we can satisfy these constraints exactly. One could use any policy gradient method (e.g. Schulman et al., 2015; 2017; Heess et al., 2015; Haarnoja et al., 2018) to obtain $q_k(a|s)$ in a *parametric* form instead. However, solving the constrained optimization for parametric $q_k(a|s)$ is not exact, and the constraints may not be well satisfied, which impedes the use of ϵ_k to encode preferences. Moreover, assuming a parametric $q_k(a|s)$ requires maintaining a function approximator (e.g., a neural network) per objective, which can significantly increase the complexity of the algorithm and limits scalability.

Choosing ϵ_k. It is more intuitive to encode preferences via ϵ_k rather than via scalarization weights, because the former is invariant to the scale of rewards. In other words, having a desired preference across objectives narrows down the range of reasonable choices for ϵ_k, but does not narrow down the range of reasonable choices for scalarization weights. In order to identify reasonable scalarization weights, a RL practitioner needs to additionally be familiar with the scale of rewards for each objective. In practice, we have found that learning performance is robust to a wide range of scales for ϵ_k. It is the *relative* scales of the ϵ_k that matter for encoding preferences over objectives—the larger a particular ϵ_k is with respect to others, the more that objective k is preferred. On the other hand, if $\epsilon_k = 0$, then objective k will have no influence and will effectively be ignored. In Appendix A.1, we provide suggestions for setting ϵ_k, given a desired preference over objectives.

4.2.2. Fitting a new parametric policy (Step 2)

In the previous section, for each objective k, we have obtained an improved action distribution $q_k(a|s)$. Next, we want to combine these distributions to obtain a single parametric policy that trades off the objectives according to the constraints ϵ_k that we set. For this, we solve a supervised learning problem that fits a parametric policy to the per-objective action distributions from step 1,

$$\theta_{\text{new}} = \operatorname*{argmax}_{\theta} \sum_{k=1}^{N} \int_s \mu(s) \int_a q_k(a|s) \log \pi_\theta(a|s) \, \mathrm{d}a \, \mathrm{d}s$$
$$\text{s.t.} \int_s \mu(s) \, \mathrm{KL}(\pi_{\text{old}}(a|s) \,\|\, \pi_\theta(a|s)) \, \mathrm{d}s < \beta \,, \quad (5)$$

where θ are the parameters of our policy (a neural network) and the KL constraint enforces a trust region of size β that limits the overall change in the parametric policy. The KL constraint in this step has a regularization effect that prevents the policy from overfitting to the sample-based action distributions, and therefore avoids premature convergence and improves stability of learning (Schulman et al., 2015; Abdolmaleki et al., 2018a;b).

Similar to the first policy improvement step, we evaluate the integrals by using the L states sampled from the replay buffer and the M actions per state sampled from the old policy. In order to optimize (5) using gradient descent, we employ Lagrangian relaxation, similar to in MPO (Abdolmaleki et al., 2018a) (see Appendix C for more detail).

5. Experiments: Toy Domains

In the empirical evaluation that follows, we will first demonstrate the mechanics and scale-invariance of MO-MPO in a single-state environment (Sec. 5.1), and then show that MO-MPO can find all Pareto-dominant policies in a popular MORL benchmark (Sec. 5.2). Finally, we show the benefit of using MO-MPO in high-dimensional continuous control domains, including on a real robot (Sec. 6). Appendices A and B contain a detailed description of all domains and tasks, experimental setups, and implementation details.

Baselines. The goal of our empirical evaluation is to analyze the benefit of using our proposed multi-objective policy improvement step (Sec. 4.2.2), that encodes preferences over objectives via constraints ϵ_k on expected KL-divergences, rather than via weights w_k. Thus, we primarily compare MO-MPO against *scalarized MPO*, which relies on linear scalarization weights w_k to encode preferences. The only difference between MO-MPO and scalarized MPO is the policy improvement step: for scalarized MPO, a single improved action distribution $q(a|s)$ is computed, based on $\sum_k w_k Q_k(s,a)$ and a single KL constraint ϵ.

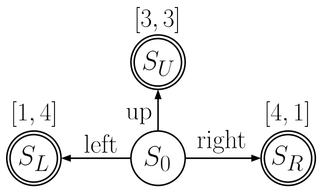

Figure 2. A simple environment in which the agent starts at state S_0, and chooses to navigate to one of three terminal states. There are two objectives. Taking the `left` action, for instance, leads to a reward of 1 for the first objective and 4 for the second.

State-of-the-art approaches that combine MORL with deep RL assume linear scalarization as well, either learning a separate policy for each setting of weights (Mossalam et al., 2016) or learning a single policy conditioned on scalarization weights (Friedman & Fontaine, 2018; Abels et al., 2019). Scalarized MPO addresses the former problem, which is easier. The policy evaluation step in scalarized MPO is analagous to *scalarized Q-learning*, proposed by Mossalam et al. (2016). As we show later in Sec. 6, even learning an optimal policy for a single scalarization is difficult in high-dimensional continuous control domains.

5.1. Simple World

First, we will examine the behavior of MO-MPO in a simple multi-armed bandit with three actions (`up`, `right`, and `left`) (Fig. 2), inspired by Russell & Zimdars (2003). We train policies with scalarized MPO and with MO-MPO. The policy evaluation step is exact because the Q-value function for each objective is known: it is equal to the reward received for that objective after taking each action, as labeled in Fig. 2.[5]

We consider three possible desired preferences: equal preference for the two objectives, preferring the first, and preferring the second. Encoding preferences in scalarized MPO amounts to choosing appropriate linear scalarization weights, and in MO-MPO amounts to choosing appropriate ϵ's. We use the following weights and ϵ's:

- *equal preference*: weights $[0.5, 0.5]$ or ϵ's $[0.01, 0.01]$
- *prefer first*: weights $[0.9, 0.1]$ or ϵ's $[0.01, 0.002]$
- *prefer second*: weights $[0.1, 0.9]$ or ϵ's $[0.002, 0.01]$

We set $\epsilon = 0.01$ for scalarized MPO. If we start with a

[5]The policy improvement step can also be computed exactly, because solving for the optimal temperature η (or η_1 and η_2 in the MO-MPO case) is a convex optimization problem, and the KL-constrained policy update is also a convex optimization problem when there is only one possible state. We use CVXOPT (Andersen et al., 2020) as our convex optimization solver.

uniform policy and run MPO with $\beta = 0.001$ until the policy converges, scalarized MPO and MO-MPO result in similar policies (Fig. 3, solid bars): up for *equal preference*, right for *prefer first*, and left for *prefer second*.

However, if we make the rewards *imbalanced* by multiplying the rewards obtained for the first objective by 20 (e.g., left now obtains a reward of $[20, 4]$), we see that the policies learned by scalarized MPO shift to preferring the optimal action for the first objective (right) in both the *equal preference* and *prefer second* cases (Fig. 3, striped bars). In contrast, the final policies for MO-MPO are the same as for balanced rewards, because in each policy improvement step, MO-MPO optimizes for a separate temperature η_k that scales each objective's Q-value function. This η_k is computed based on the corresponding allowed KL-divergence ϵ_k, so when the rewards for any objective k are multiplied by a factor but ϵ_k remains the same, the computed η_k ends up being scaled by that factor as well, neutralizing the effect of the scaling of rewards (see Eq. (4)).

Even in this simple environment, we see that MO-MPO's scale-invariant way of encoding preferences is valuable. In more complex domains, in which the Q-value functions must be learned in parallel with the policy, the (automatic) dynamic adjustment of temperatures η_k per objective becomes more essential (Sec. 6).

The scale of ϵ_k controls the amount that objective k can influence the policy's update. If we set $\epsilon_1 = 0.01$ and sweep over the range from 0 to 0.01 for ϵ_2, the resulting policies go from always picking right, to splitting probability across right and up, to always picking up (Fig. 4, right). In contrast, setting weights leads to policies quickly converging to placing all probability on a single action (Fig. 4, left). We hypothesize this limits the ability of scalarized MPO to explore and find compromise policies (that perform well with respect to all objectives) in more challenging domains.

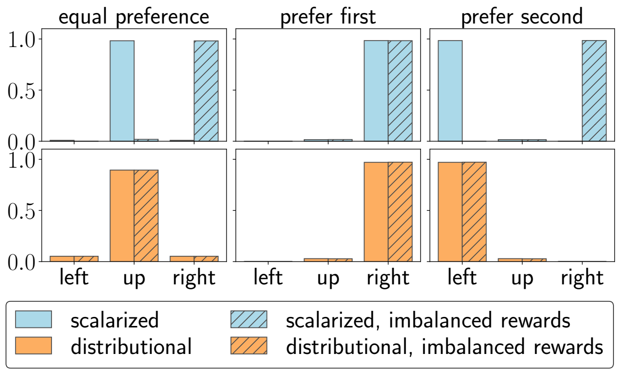

Figure 3. When two objectives have comparable reward scales (solid bars), scalarized MPO (first row) and MO-MPO (second row) learn similar policies, across three different preferences. However, when the scale of the first objective is much higher (striped bars), scalarized MPO shifts to always preferring the first objective. In contrast, MO-MPO is scale-invariant and still learns policies that satisfy the preferences. The y-axis denotes action probability.

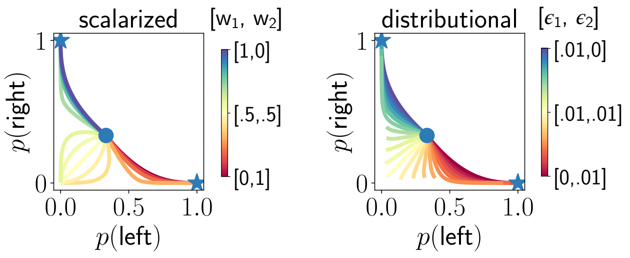

Figure 4. A visualization of policies during learning—each curve corresponds to a particular setting of weights (left) or ϵ's (right). Policies are initialized as uniform (the blue dot), and are trained until convergence. Each point (x, y) corresponds to the policy with $p(\texttt{left}) = x$, $p(\texttt{right}) = y$, and $p(\texttt{up}) = 1 - x - y$. The top left and bottom right blue stars denote the optimal policy for the first and second objectives, respectively.

5.2. Deep Sea Treasure

An important quality of any MORL approach is the ability to find a variety of policies on the true Pareto front (Roijers et al., 2013). We demonstrate this in Deep Sea Treasure (DST) (Vamplew et al., 2011), a popular benchmark for testing MORL approaches. DST consists of a 11×10 grid world with ten treasure locations. The agent starts in the upper left corner ($s_0 = (0, 0)$) and has a choice of four actions (moving one square up, right, down, and left). When the agent picks up a treasure, the episode terminates. The agent has two objectives: treasure value and time penalty. The time penalty is -1 for each time step that the agent takes, and farther-away treasures have higher treasure values. We use the treasure values in Yang et al. (2019).

We ran scalarized MPO with weightings $[w, 1 - w]$ and $w \in [0, 0.01, 0.02, \ldots, 1]$. All policies converged to a point on the true Pareto front, and all but three found the optimal policy for that weighting. In terms of coverage, policies were found for eight out of ten points on the Pareto front.[6]

We ran MO-MPO on this task as well, for a range of ϵ: $\epsilon_{\text{time}} \in [0.01, 0.02, 0.05]$ and $\epsilon_{\text{treasure}} = c * \epsilon_{\text{time}}$, where $c \in [0.5, 0.51, 0.52, \ldots, 1.5]$. All runs converged to a policy on the true Pareto front, and MO-MPO finds policies for all ten points on the front (Fig. 5, left). Note that it is the *ratio* of ϵ's that matters, rather than the exact settings—across all settings of ϵ_{time}, similar ratios of $\epsilon_{\text{treasure}}$ to ϵ_{time} result in similar policies; as this ratio increases, policies tend to prefer higher-value treasures (Fig. 5, right).

[6]Since scalarized MPO consistently finds the optimal policy for any given weighting in this task, with a more strategic selection of weights, we expect policies for all ten points would be found.

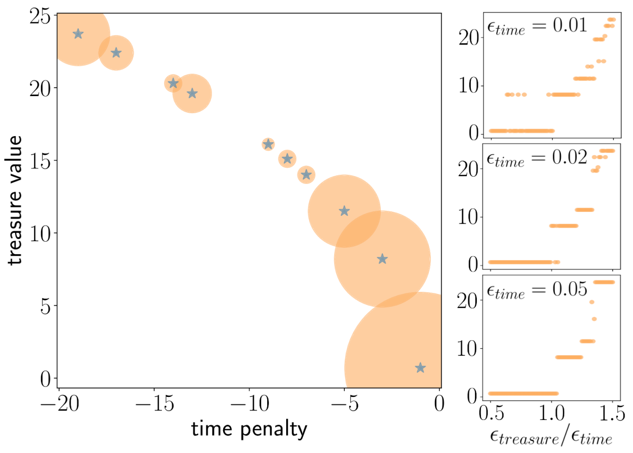

Figure 5. Left: Blue stars mark the true Pareto front for Deep Sea Treasure. MO-MPO, with a variety of settings for ϵ_k, discovers all points on the true Pareto front. The area of the orange circles is proportional to the number of ϵ_k settings that converged to that point. Right: As more preference is given to the treasure objective (i.e., as x increases), policies tend to prefer higher-value treasures. Each orange dot in the scatterplot corresponds to a particular setting of ϵ_k.

6. Experiments: Continuous Control Domains

The advantage of encoding preferences via ϵ's, rather than via weights, is apparent in more complex domains. We compared our approaches, MO-MPO and MO-V-MPO, against scalarized MPO and V-MPO in four high-dimensional continuous control domains, in MuJoCo (Todorov et al., 2012) and on a real robot. The domains we consider are:

Humanoid: We use the humanoid ***run*** task defined in DeepMind Control Suite (Tassa et al., 2018). Policies must optimize for horizontal speed h while minimizing energy usage. The task reward is $\min(h/10, 1)$ where h is in meters per second, and the energy usage penalty is action $\ell 2$-norm. The humanoid has 21 degrees of freedom, and the observation consists of joint angles, joint velocities, head height, hand and feet positions, torso vertical orientation, and center-of-mass velocity, for a total of 67 dimensions.

Shadow Hand: We consider three tasks on the Shadow Dexterous Hand: ***touch***, ***turn***, and ***orient***. In the ***touch*** and ***turn*** tasks, policies must complete the task while minimizing "pain." A sparse task reward of 1.0 is given for pressing the block with greater than 5N of force or for turning the dial from a random initial location to the target location. The pain penalty penalizes the robot for colliding with objects at high speed; this penalty is defined as in Huang et al. (2019). In the ***orient*** task, there are three aligned objectives: touching the rectangular peg, lifting it to a given height, and orienting it to be perpendicular to the ground. All three rewards are between 0 and 1. The Shadow Hand has five fingers and 24 degrees of freedom, actuated by 20 motors. The observation consists of joint angles, joint velocities, and touch sensors, for a total of 63 dimensions. The ***touch*** and ***turn*** tasks terminate when the goal is reached or after 5 seconds, and the ***orient*** task terminates after 10 seconds.

Humanoid Mocap: We consider the large-scale humanoid motion capture tracking task from Hasenclever et al. (2020), in which policies must learn to follow motion capture reference data. There are five objectives, each capturing a different aspect of the similarity of the pose between the simulated humanoid and the mocap target: joint orientations, joint velocities, hand and feet positions, center-of-mass positions, and certain body positions and joint angles. These objectives are described in detail in Appendix B.4. In order to balance these multiple objectives, prior work relied on heavily-tuned reward functions (e.g. Peng et al., 2018). The humanoid has 56 degrees of freedom and the observation is 1021-dimensional, consisting of proprioceptive observations as well as six steps of motion capture reference frames. In total, we use about 40 minutes of locomotion mocap data, making this an extremely challenging domain.

Sawyer Peg-in-Hole: We train a Rethink Robotics Sawyer robot arm to insert a cylindrical peg into a hole, while minimizing wrist forces. The task reward is shaped toward positioning the peg directly above the hole and increases for insertion, and the penalty is the $\ell 1$-norm of Cartesian forces measured by the wrist force-torque sensor. The latter implicitly penalizes contacts and impacts, as well as excessive directional change (due to the gripper's inertia inducing forces when accelerating). We impose a force threshold to protect the hardware—if this threshold is exceeded, the episode is terminated. The action space is the end effector's Cartesian velocity, and the observation is 102-dimensional, consisting of Cartesian position, joint position and velocity, wrist force-torque, and joint action, for three timesteps.

6.1. Evaluation Metric

We run MO-(V-)MPO and scalarized (V-)MPO with a wide range of constraint settings ϵ_k and scalarization weights w_k, respectively, corresponding to a wide range of possible desired preferences. (The exact settings are provided in Appendix A.) For tasks with two objectives, we plot the Pareto front found by each approach. We also compute the *hypervolume* of each found Pareto front; this metric is commonly-used for evaluating MORL algorithms (Vamplew et al., 2011). Given a set of policies Π and a reference policy r that is dominated by all policies in this set, this metric is the hypervolume of the space of all policies that dominate r and are dominated by at least one policy in Π. We use DEAP (Fortin et al., 2012) to compute hypervolumes.

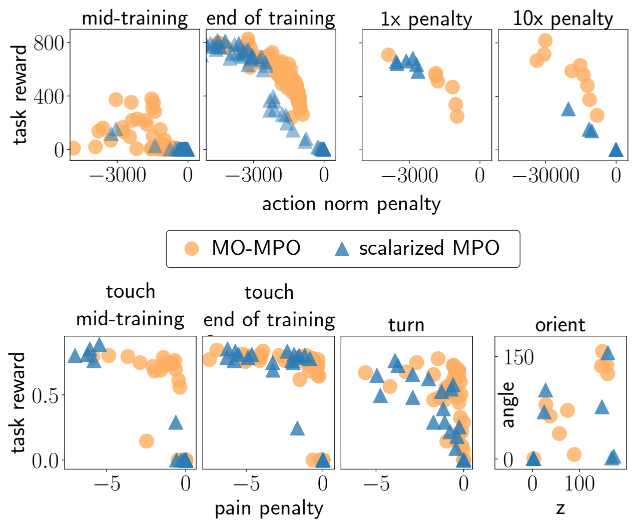

Figure 6. Pareto fronts found by MO-MPO and scalarized MO-MPO for humanoid ***run*** (top row) and Shadow Hand tasks. Each dot represents a single trained policy. Corresponding hypervolumes are in Table 1. Task reward is discounted for ***touch*** and ***turn***, with a discount factor of 0.99. For ***orient***, the x and y axes are the total reward for the lift and orientation objectives, respectively.

6.2. Results: Humanoid and Shadow Hand

In the ***run***, ***touch***, and ***turn*** tasks, the two objectives are competing—a very high preference for minimizing action norm or pain, as opposed to getting task reward, will result in a policy that always chooses zero-valued actions. Across these three tasks, the Pareto front found by MO-MPO is superior to the one found by scalarized MPO, with respect to the hypervolume metric (Table 1).[7] In particular, MO-MPO finds more policies that perform well with respect to *both* objectives, i.e., are in the upper right portion of the Pareto front. MO-MPO also speeds up learning on ***run*** and ***touch*** (Fig. 6). Qualitatively, MO-MPO trains ***run*** policies that look more natural and "human-like"; videos are at `http://sites.google.com/view/mo-mpo`.

When we scale the action norm penalty by 10× for ***run***, scalarized MPO policies no longer achieve high task reward, whereas MO-MPO policies do (Fig. 6, top right). This supports that MO-MPO's encoding of preferences is indeed scale-invariant. When the objectives are aligned and have similarly scaled rewards, as in the ***orient*** task, MO-MPO and scalarized MPO perform similarly, as expected.

Ablation. We also ran vanilla MPO on humanoid ***run***, with the same range of weight settings as for scalarized MPO, to investigate how useful Q-decomposition is. In vanilla MPO, we train a single critic on the scalarized reward function, which is equivalent to removing Q-decomposition from scalarized MPO. Vanilla MPO trains policies that achieve similar task reward (up to 800), but with twice the action norm penalty (up to -10^4). As a result, the hypervolume of the Pareto front that vanilla MPO finds is more than a magnitude worse than that of scalarized MPO and MO-MPO (2.2×10^6 versus 2.6×10^7 and 6.9×10^7, respectively).

Task	**scalarized MPO**	**MO-MPO**
Humanoid ***run***, mid-training	1.1×10^6	$\mathbf{3.3\times10^6}$
Humanoid ***run***	6.4×10^6	$\mathbf{7.1\times10^6}$
Humanoid ***run***, normal scale	5.0×10^6	$\mathbf{5.9\times10^6}$
Humanoid ***run***, 10× penalty	2.6×10^7	$\mathbf{6.9\times10^7}$
Shadow ***touch***, mid-training	14.3	**15.6**
Shadow ***touch***	16.2	**16.4**
Shadow ***turn***	14.4	**15.4**
Shadow ***orient***	$\mathbf{2.8\times10^4}$	$\mathbf{2.8\times10^4}$
Humanoid Mocap	$\mathbf{3.86\times10^{-6}}$	3.41×10^{-6}

Table 1. Hypervolume measurements across tasks and approaches.

6.3. Results: Humanoid Mocap

The objectives in this task are mostly aligned. In contrast to the other experiments, we use V-MPO (Song et al., 2020) as the base algorithm because it outperforms MPO in learning this task. In addition, since V-MPO is an on-policy variant of MPO, this enables us to evaluate our approach in the on-policy setting. Each training run is very computationally expensive, so we train only a handful of policies each.[8] None of the MO-V-MPO policies are dominated by those found by V-MPO. In fact, although the weights span a wide range of "preferences" for the joint velocity, the only policies found by V-MPO that are *not* dominated by a MO-V-MPO policy are those with extreme values for joint velocity reward (either ≤ 0.006 or ≥ 0.018), whereas it is between 0.0103 and 0.0121 for MO-V-MPO policies.

Although the hypervolume of the Pareto front found by V-MPO is higher than that of MO-V-MPO (Table 1), finding policies that over- or under- prioritize any objective is undesirable. Qualitatively, the policies trained with MO-V-MPO look more similar to the mocap reference data—they exhibit less feet jittering, compared to those trained with scalarized V-MPO; this can be seen in the corresponding video.

6.4. Results: Sawyer Peg-in-Hole

In this task, we would like the robot to prioritize successful task completion, while minimizing wrist forces. With this

[7]We use hypervolume reference points of $[0, -10^4]$ for ***run***, $[0, -10^5]$ for ***run*** with 10× penalty, $[0, -20]$ for ***touch*** and ***turn***, and the zero vector for ***orient*** and humanoid mocap.

[8]For MO-V-MPO, we set all $\epsilon_k = 0.01$. Also, for each objective, we set $\epsilon_k = 0.001$ and set all others to 0.01. For V-MPO, we fix the weights of four objectives to reasonable values, and try weights of $[0.1, 0.3, 0.6, 1, 5]$ for matching joint velocities.

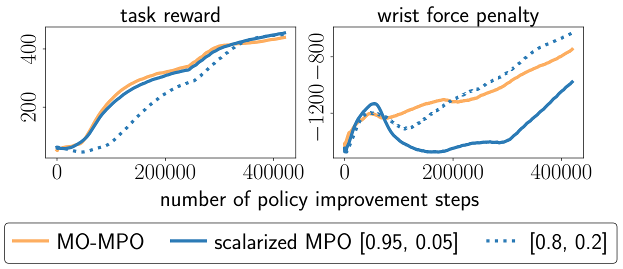

Figure 7. Task reward (left) and wrist force penalty (right) learning curves for the Sawyer peg-in-hole task. The policy trained with MO-MPO quickly learns to optimize *both* objectives, whereas the other two do not. Each line represents a single trained policy.

in mind, for MO-MPO we set $\epsilon_{\text{task}} = 0.1$ and $\epsilon_{\text{force}} = 0.05$, and for scalarized MPO we try $[w_{\text{task}}, w_{\text{force}}] = [0.95, 0.05]$ and $[0.8, 0.2]$. We find that policies trained with scalarized MPO focus all learning on a single objective at the beginning of training; we also observed this in the ***touch*** task, where scalarized MPO policies quickly learn to either maximize task reward or minimize pain penalty, but not both (Fig. 6, bottom left). In contrast, the policy trained with MO-MPO simultaneously optimizes for *both* objectives throughout training. In fact, throughout training, the MO-MPO policy does just as well with respect to task reward as the scalarized MPO policy that cares more about task reward, and similarly for wrist force penalty (Fig. 7).

7. Discussion

Choosing ϵ's Versus Weights. Our approach requires an RL practitioner to encode their preferences across objectives by selecting appropriate hyperparameters ϵ_k, rather than choosing scalarization weights. We believe choosing ϵ_k is easier than choosing scalarization weights, because the former is invariant to the scale of rewards. In other words, having a desired preference across objectives narrows down the range of reasonable choices for ϵ_k, but *not* for scalarization weights. In order to identify reasonable scalarization weights, the RL practitioner needs to additionally be familiar with the scale of rewards for each objective.

To see why this is the case, suppose there are two objectives that are equally important. Then ϵ_1 and ϵ_2 should be approximately equal to each other. However, if the two objectives have different scales of rewards (as is typically the case), then having equal scalarization weights does *not* correspond to equal preference between the two objectives. Instead, the scalarization weight for the objective with lower-magnitude rewards should be higher than the weight for the other objective, to counteract the difference in reward scales; how much higher depends on the relative reward scales. In Appendix A.1, we provide guidelines for how to choose ϵ_k, depending on what the desired preference over objectives is.

Outperforming Scalarization. Empirically, MO-MPO finds a better Pareto front than scalarized MPO does, but why is this the case? One reason may be that the relative scales of rewards between objectives depends on the (behavioral) policy. Since the policy changes over the course of training, the relative scales of acquired rewards also changes. Thus, a fixed setting of weights w will likely encode different preferences over the course of training.

One solution to this would be to adapt the weights during training. MO-MPO can be seen as a principled approach to adaptive weighting, since it scales the Q-values of each objective by a learned temperature η_k per objective, as in (3). These per-objective temperatures vary over the course of training: for each policy improvement step, they are chosen in order to satisfy the constraint imposed by ϵ_k, which defines the influence of objective k. If we instead fixed the temperatures, then we would then suffer from the same problems as the scalarization baseline, because the influence of each objective would vary per policy improvement step, depending on what the range of learned Q-values is at that point in training.

8. Conclusions and Future Work

In this paper we presented a new distributional perspective on multi objective reinforcement learning, that is derived from the RL-as-inference perspective. This view leads to two novel multi-objective RL algorithms, namely MO-MPO and MO-V-MPO. We showed that these algorithms enable practitioners to encode preferences in a scale-invariant way, and empirically lead to faster learning and convergence to a better Pareto front, compared to linear scalarization.

A limitation of this work is that MO-(V-)MPO is a single-policy approach to MORL, so producing a Pareto front of policies requires iterating over a relatively large number of ϵ's, which is computationally expensive. In future work, we plan to extend MO-(V-)MPO into a true multiple-policy MORL approach, either by conditioning the policy on settings of ϵ's or by developing a way to strategically select ϵ's to train policies for, analogous to what prior work has done for weights (e.g., Roijers et al. (2014)).

Acknowledgments

The authors would like to thank Guillaume Desjardins for providing valuable feedback on an earlier draft of the paper. We would also like to thank Csaba Szepesvari, Jost Tobias Springenberg, Steven Bohez, Philemon Brakel, Brendan Tracey, Jonas Degrave, Jonas Buchli, Alex Lee, Adam Kosiorek, Leslie Fritz, Chloe Rosenberg, and many others at DeepMind for their support and input during this project.

References

Abdolmaleki, A., Springenberg, J. T., Degrave, J., Bohez, S., Tassa, Y., Belov, D., Heess, N., and Riedmiller, M. Relative entropy regularized policy iteration. *arXiv preprint arXiv:1812.02256*, 2018a.

Abdolmaleki, A., Springenberg, J. T., Tassa, Y., Munos, R., Heess, N., and Riedmiller, M. Maximum a posteriori policy optimisation. In *Proceedings of the Sixth International Conference on Learning Representations (ICLR)*, 2018b.

Abels, A., Roijers, D. M., Lenaerts, T., Nowé, A., and Steckelmacher, D. Dynamic weights in multi-objective deep reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

Achiam, J., Held, D., Tamar, A., and Abbeel, P. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pp. 22–31, 2017.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.

Andersen, M. S., Dahl, J., and Vandenberghe, L. CVXOPT: A Python package for convex optimization, version 1.2. https://cvxopt.org/, 2020.

Barrett, L. and Narayanan, S. Learning all optimal policies with multiple criteria. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pp. 41–47, 2008.

Barth-Maron, G., Hoffman, M. W., Budden, D., Dabney, W., Horgan, D., TB, D., Muldal, A., Heess, N., and Lillicrap, T. Distributed distributional deterministic policy gradients. In *Proceedings of the Sixth International Conference on Learning Representations (ICLR)*, 2018.

Bohez, S., Abdolmaleki, A., Neunert, M., Buchli, J., Heess, N., and Hadsell, R. Value constrained model-free continuous control. *arXiv preprint arXiv:1902.04623*, 2019.

Chen, X., Ghadirzadeh, A., Björkman, M., and Jensfelt, P. Meta-learning for multi-objective reinforcement learning. *arXiv preprint arXiv:1811.03376*, 2019.

Chentanez, N., Müller, M., Macklin, M., Makoviychuk, V., and Jeschke, S. Physics-based motion capture imitation with deep reinforcement learning. In *International Conference on Motion, Interaction, and Games (MIG)*. ACM, 2018.

Chow, Y., Nachum, O., Duenez-Guzman, E., and Ghavamzadeh, M. A Lyapunov-based approach to safe reinforcement learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)*, pp. 8092–8101, 2018.

Daneshmand, M., Tale Masouleh, M., Saadatzi, M. H., Ozcinar, C., and Anbarjafari, G. A robust proportion-preserving composite objective function for scale-invariant multi-objective optimization. *Scientia Iranica*, 24(6):2977–2991, 2017.

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., and Gagné, C. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, July 2012.

Friedman, E. and Fontaine, F. Generalizing across multi-objective reward functions in deep reinforcement learning. *arXiv preprint arXiv:1809.06364*, 2018.

Gábor, Z., Kalmár, Z., and Szepesvári, C. Multi-criteria reinforcement learning. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML)*, pp. 197–205, 1998.

Grodzevich, O. and Romanko, O. Normalization and other topics in multi-objective optimization. In *Proceedings of the Fields-MITACS Industrial Problems Workshop*, 2006.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pp. 1861–1870, 2018.

Hasenclever, L., Pardo, F., Hadsell, R., Heess, N., and Merel, J. CoMic: Complementary task learning and mimicry for reusable skills. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

Heess, N., Wayne, G., Silver, D., Lillicrap, T., Erez, T., and Tassa, Y. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems 28 (NIPS)*. 2015.

Hessel, M., Soyer, H., Espeholt, L., Czarnecki, W., Schmitt, S., and van Hasselt, H. Multi-task deep reinforcement learning with popart. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3796–3803, 2019.

Huang, S. H., Zambelli, M., Kay, J., Martins, M. F., Tassa, Y., Pilarski, P. M., and Hadsell, R. Learning gentle object manipulation with curiosity-driven deep reinforcement learning. *arXiv preprint arXiv:1903.08542*, 2019.

Ishibuchi, H., Doi, K., and Nojima, Y. On the effect of normalization in MOEA/D for multi-objective and many-objective optimization. *Complex & Intelligent Systems*, 3 (4):279–294, 2017.

Kim, I. and de Weck, O. Adaptive weighted sum method for multiobjective optimization: A new method for pareto front generation. *Structural and Multidisciplinary Optimization*, 31(2):105–116, 2006.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *Proceedings of the Third International Conference on Learning Representations (ICLR)*, 2015.

Le, H., Voloshin, C., and Yue, Y. Batch policy learning under constraints. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pp. 3703–3712, 2019.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

Liu, C., Xu, X., and Hu, D. Multiobjective reinforcement learning: A comprehensive overview. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(3):385–398, 2015.

Marler, R. T. and Arora, J. S. Function-transformation methods for multi-objective optimization. *Engineering Optimization*, 37(6):551–570, 2005.

Martens, J. New perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193*, 2014.

Merel, J., Ahuja, A., Pham, V., Tunyasuvunakool, S., Liu, S., Tirumala, D., Heess, N., and Wayne, G. Hierarchical visuomotor control of humanoids. In *International Conference on Learning Representations (ICLR)*, 2019.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529, 2015.

Moffaert, K. V. and Nowé, A. Multi-objective reinforcement learning using sets of Pareto dominating policies. *Journal of Machine Learning Research*, 15:3663–3692, 2014.

Mossalam, H., Assael, Y. M., Roijers, D. M., and Whiteson, S. Multi-objective deep reinforcement learning. *arXiv preprint arXiv:1610.02707*, 2016.

Munos, R., Stepleton, T., Harutyunyan, A., and Bellemare, M. Safe and efficient off-policy reinforcement learning. In *Proceedings of the 29th International Conference on Neural Information Processing Systems*, pp. 1054–1062, 2016.

Natarajan, S. and Tadepalli, P. Dynamic preferences in multi-criteria reinforcement learning. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pp. 601–608, 2005.

Nottingham, K., Balakrishnan, A., Deshmukh, J., Christopherson, C., and Wingate, D. Using logical specifications of objectives in multi-objective reinforcement learning. *arXiv preprint arXiv:1910.01723*, 2019.

Parisi, S., Pirotta, M., Smacchia, N., Bascetta, L., and Restelli, M. Policy gradient approaches for multi-objective sequential decision making. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pp. 2323–2330, 2014.

Peng, X. B., Abbeel, P., Levine, S., and van de Panne, M. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics*, 37(4), 2018.

Pirotta, M., Parisi, S., and Restelli, M. Multi-objective reinforcement learning with continuous Pareto frontier approximation. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI)*, 2015.

Reymond, M. and Nowé, A. Pareto-DQN: Approximating the Pareto front in complex multi-objective decision problems. In *Proceedings of the Adaptive and Learning Agents Workshop at the 18th International Conference on Autonomous Agents and MultiAgent Systems AAMAS*, 2019.

Riedmiller, M., Hafner, R., Lampe, T., Neunert, M., Degrave, J., Van de Wiele, T., Mnih, V., Heess, N., and Springenberg, J. T. Learning by playing - solving sparse reward tasks from scratch. *arXiv preprint arXiv:1802.10567*, 2018.

Roijers, D. M., Vamplew, P., Whiteson, S., and Dazeley, R. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48(1):67–113, 2013.

Roijers, D. M., Whiteson, S., and Oliehoek, F. A. Linear support for multi-objective coordination graphs. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pp. 1297–1304, 2014.

Russell, S. and Zimdars, A. L. Q-decomposition for reinforcement learning agents. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning (ICML)*, pp. 656–663, 2003.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shadow Robot Company. Shadow dexterous hand. https://www.shadowrobot.com/products/dexterous-hand/, 2020.

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., and Hassabis, D. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529:484–503, 2016.

Song, H. F., Abdolmaleki, A., Springenberg, J. T., Clark, A., Soyer, H., Rae, J. W., Noury, S., Ahuja, A., Liu, S., Tirumala, D., Heess, N., Belov, D., Riedmiller, M., and Botvinick, M. M. V-MPO: On-policy maximum a posteriori policy optimization for discrete and continuous control. In *International Conference on Learning Representations*, 2020.

Sutton, R. S. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

Tan, J., Zhang, T., Coumans, E., Iscen, A., Bai, Y., Hafner, D., Bohez, S., and Vanhoucke, V. Sim-to-real: Learning agile locomotion for quadruped robots. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., de Las Casas, D., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T., and Riedmiller, M. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Teh, Y. W., Bapst, V., Czarnecki, W. M., Quan, J., Kirkpatrick, J., Hadsell, R., Heess, N., and Pascanu, R. Distral: Robust multitask reinforcement learning. *CoRR*, abs/1707.04175, 2017. URL http://arxiv.org/abs/1707.04175.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5026–5033, 2012.

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

van Hasselt, H. P., Guez, A., Hessel, M., Mnih, V., and Silver, D. Learning values across many orders of magnitude. In *Advances in Neural Information Processing Systems*, pp. 4287–4295, 2016.

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

van Seijen, H., Fatemi, M., Romoff, J., Laroche, R., Barnes, T., and Tsang, J. Hybrid reward architecture for reinforcement learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)*, pp. 5398–5408, 2017.

Watkins, C. J. C. H. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, May 1989.

Wray, K. H., Zilberstein, S., and Mouaddib, A.-I. Multi-objective MDPs with conditional lexicographic reward preferences. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI)*, pp. 3418–3424, 2015.

Wulfmeier, M., Abdolmaleki, A., Hafner, R., Springenberg, J. T., Neunert, M., Hertweck, T., Lampe, T., Siegel, N., Heess, N., and Riedmiller, M. Regularized hierarchical policies for compositional transfer in robotics. *arXiv preprint arXiv:1906.11228*, 2019.

Yang, R., Sun, X., and Narasimhan, K. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NeurIPS)*, pp. 14610–14621, 2019.

Zeng, A., Song, S., Lee, J., Rodriguez, A., and Funkhouser, T. Tossingbot: Learning to throw arbitrary objects with residual physics. In *Proceedings of Robotics: Science and Systems (RSS)*, 2019.

Zhan, H. and Cao, Y. Relationship explainable multi-objective optimization via vector value function based reinforcement learning. *arXiv preprint arXiv:1910.01919*, 2019.

Zuluaga, M., Krause, A., and Püschel, M. ϵ-pal: An active learning approach to the multi-objective optimization problem. *Journal of Machine Learning Research*, 17(1):3619–3650, 2016.