Prediction-Guided Multi-Objective Reinforcement Learning for Continuous Robot Control

Jie Xu [1] Yunsheng Tian [1] Pingchuan Ma [1] Daniela Rus [1] Shinjiro Sueda [2] Wojciech Matusik [1]

Abstract

Many real-world control problems involve conflicting objectives where we desire a dense and high-quality set of control policies that are optimal for different objective preferences (called Pareto-optimal). While extensive research in multi-objective reinforcement learning (MORL) has been conducted to tackle such problems, multi-objective optimization for complex continuous robot control is still under-explored. In this work, we propose an efficient evolutionary learning algorithm to find the Pareto set approximation for continuous robot control problems, by extending a state-of-the-art RL algorithm and presenting a novel prediction model to guide the learning process. In addition to efficiently discovering the individual policies on the Pareto front, we construct a continuous set of Pareto-optimal solutions by Pareto analysis and interpolation. Furthermore, we design seven multi-objective RL environments with continuous action space, which is the first benchmark platform to evaluate MORL algorithms on various robot control problems. We test the previous methods on the proposed benchmark problems, and the experiments show that our approach is able to find a much denser and higher-quality set of Pareto policies than the existing algorithms.

1. Introduction

Multi-objective problems have received significant attention because most real-world scenarios involve making trade-offs with respect to different performance metrics. This is especially true in robotic control, in which the notion of performance usually involves different conflicting objectives.

[1]Computer Science & Artificial Intelligence Laboratory (CSAIL), Massachusetts Institute of Technology [2]Texas A&M University. Correspondence to: Jie Xu <jiex@csail.mit.edu>.

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

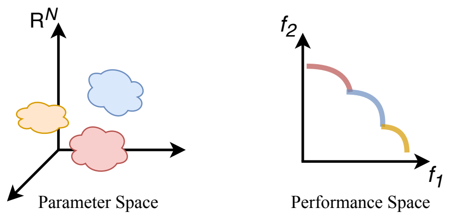


Figure 1. **Parameter space and performance space of the Pareto policies.** (Left) The Pareto set is composed from a disjoint set of policy families in the N dimensional parameter space. (Right) The policies from each family map to a continuous segment on the Pareto front in the performance space.

For example, when designing a control policy for a running quadruped robot, we need to consider two conflicting objectives: running speed and energy efficiency. In contrast to a *single-objective* environment, which measures performance using a single scalar value and where a single best solution exists, with a *multi-objective* problem, performance is measured using multiple objectives, and multiple optimal solutions exist. One optimal policy may prefer high speed at the cost of lower energy efficiency, whereas another optimal policy might prefer high energy efficiency at the cost of lower speed. In general, many optimal policies exist depending on the chosen trade-off between these two metrics. In the end, a human is responsible for selecting the preference among different metrics, and this determines the corresponding optimal policy.

One popular way of solving multi-objective control problems is to compute a meta policy (Chen et al., 2018). A meta policy is a general policy that is not necessarily optimal but can be relatively quickly adapted to different trade-offs between performance objectives. Unfortunately, such adapted control policies are not necessarily optimal. For instance, adapting a general meta control policy for a quadruped robot to run as fast as possible will often result in a suboptimal policy for this metric.

In this work, we show that an effective representation for obtaining the best performance trade-offs for multi-objective robot control is a Pareto set of control policies. We empirically show that a Pareto set cannot be effectively represented using a single continuous policy family. Rather, a Pareto set is composed from a set of disjoint policy families, each occupying a continuous manifold in the parameter space and being responsible for a segment on the Pareto front in the performance space (Figure 1).

To find such Pareto representations, we propose an efficient algorithm to compute the Pareto set of policies. Our algorithm works in two steps. In the first step, we find a dense and high-quality set of policies on the Pareto front using reinforcement learning strategies based on a novel prediction-guided evolutionary learning algorithm. In each generation, an analytical model is fitted for each policy to predict the expected improvement along each optimization direction. An optimization problem is then solved to select the policies and the associated optimization directions that are expected to best improve the quality of the Pareto. In the second step, we conduct a Pareto analysis on the computed Pareto-optimal policies to identify different policy families and to compute a continuous representation for each of these policy families.

In order to benchmark our proposed algorithm, we design a set of multi-objective robot control problems with a continuous action space. The performance of each policy can be evaluated using a physics-based simulation system (Todorov et al., 2012). Our experiments demonstrate that the proposed algorithm can efficiently find a significantly higher-quality set of Pareto-optimal policies than existing methods. Moreover, based on these policies it can reconstruct continuous policy families that span the whole Pareto front.

2. Background

2.1. Multi-Objective Markov Decision Process

A multi-objective control problem can be formulated as a multi-objective Markov Decision Process (MOMDP), which is defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \boldsymbol{R}, \boldsymbol{\gamma}, \mathcal{D} \rangle$ with state space $\mathcal{S}$, action space $\mathcal{A}$, state transition probability $\mathcal{P}(s' \mid s, a)$, vector of reward functions $\boldsymbol{R} = [\boldsymbol{r}_1, ..., \boldsymbol{r}_m]^\top$ with $\boldsymbol{r}_i : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, vector of discount factors $\boldsymbol{\gamma} = [\gamma_1, ..., \gamma_m]^\top \in [0,1]^m$, initial state distribution $\mathcal{D}$, and the number of objectives m.

In MOMDPs, a policy $\pi_{\boldsymbol{\theta}} : \mathcal{S} \rightarrow \mathcal{A}$ is associated with a vector of expected returns $\boldsymbol{J}^\pi = [J_1^\pi, ..., J_m^\pi]^T$, where

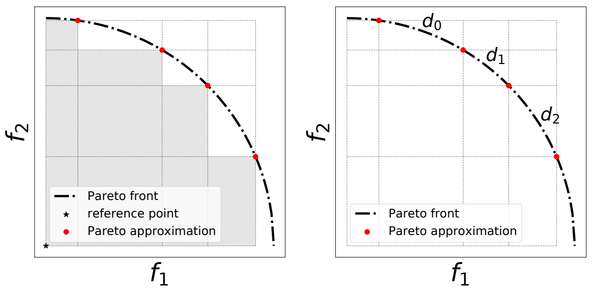
$$J_i^\pi = \mathbb{E}\left[\sum_{t=0}^{T} \gamma_i^t \boldsymbol{r}_i(s_t, a_t) \mid s_0 \sim \mathcal{D}, a_t \sim \pi_{\boldsymbol{\theta}}(s_t)\right].$$

The state s_{t+1} is reached from state s_t by action a_t, and T is the horizon. We use π for $\pi_{\boldsymbol{\theta}}$ for brevity.

2.2. Multi-Objective Optimization

A multi-objective optimization problem is formulated as:

$$\max_\pi \mathbf{F}(\pi) = \max_\pi [f_1(\pi), f_2(\pi), ..., f_m(\pi)],$$



(a) Hypervolume metric (b) Sparsity metric

Figure 2. (a) Hypervolume metric in 2-objective space is the area (shaded) dominated by the Pareto front approximation and dominating the reference point. (b) Sparsity metric in 2-objective space measures the average square distance between consecutive points in Pareto approximation. In this case, $\mathcal{S} = \frac{1}{3}(d_0^2 + d_1^2 + d_2^2)$.

where m is the number of objectives, π is the policy, and in our problem $f_i(\pi) = J_i^\pi$.

In multi-objective optimization problems, no single optimal policy exists that maximizes all the objectives. Instead a set of non-dominated solutions called the Pareto set is desired:

Definition 2.1 *(Pareto optimality) We say policy π dominates policy π' if $\boldsymbol{F}(\pi) \geq \boldsymbol{F}(\pi')$ and $\boldsymbol{F}(\pi) \neq \boldsymbol{F}(\pi')$. A policy π is Pareto optimal if and only if it is not dominated by any other policies. The set of all such policies is called the Pareto set, and the image of the Pareto set in the objective space is called the Pareto front.*

Since the true (optimal) Pareto set is usually impossible to obtain in complex problems, the goal of multi-objective optimization is to find the set of solutions that best approximates the optimal Pareto set. To measure the quality of an approximated Pareto front, two factors are usually considered (Riquelme et al., 2015): the convergence towards the true Pareto front and the uniformity of the solution distribution, which are best measured by hypervolume metric (Zitzler & Thiele, 1999) (illustrated in Figure 2(a)):

Definition 2.2 *(Hypervolume metric) Let P be a Pareto front approximation in an m-dimensional objective space and $\boldsymbol{r} \in \mathbb{R}^m$ be the reference point. Then the hypervolume metric $\mathcal{H}(P)$ is*

$$\mathcal{H}(P) = \int_{\mathbb{R}^m} \mathbb{1}_{H(P)}(z) dz, \tag{1}$$

where $H(P) = \{\boldsymbol{z} \in Z \mid \exists 1 \leq i \leq |P| : \boldsymbol{r} \preceq \boldsymbol{z} \preceq P(i)\}$. $P(i)$ is the i-th solution in P, $\preceq$ is the relation operator of objective dominance, and $\mathbb{1}_{H(P)}$ is a Dirac delta function that equals 1 if $z \in H(P)$ and 0 otherwise.

A dense policy set is always preferred for better Pareto approximation. Therefore, a sparsity metric is also defined to measure that property (illustrated in Figure 2(b)):

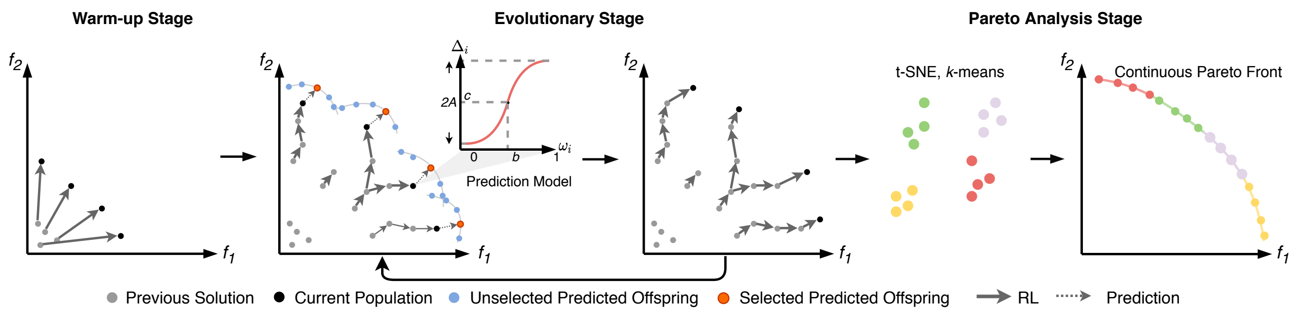


Figure 3. **Overview of the algorithm.** *Warm-up stage*: optimize n initial policies with different weights. *Evolutionary stage*: build an improvement prediction model for each policy and solve a prediction-guided optimization to select n best policy-weight pairs to be processed. The resulting polices are used to update the population and the prediction models. *Pareto analysis stage*: identify different policy families and construct a continuous Pareto representation.

Definition 2.3 *(Sparsity metric) Let P be a Pareto front approximation in an m-dimensional objective space. Then the Sparsity metric $\mathcal{S}(P)$ is*

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

where $\tilde{P}_j$ is the sorted list for the j-th objective values in P, and $\tilde{P}_j(i)$ is the i-th value in this sorted list.

In a word, a desired Pareto set approximation is expected to have high hypervolume metric and low sparsity metric.

3. Prediction-Guided MORL

In this section, we propose our main contributions: a prediction-guided evolutionary learning algorithm for multi-objective control problems, and a Pareto analysis tool to construct a continuous Pareto representation. An overview of the algorithm is provided in Section 3.1, and the details of our main contributions are described in Sections 3.2-3.5.

3.1. Overview

As shown in Figure 3 and Algorithm 1, we propose an efficient algorithm to compute the Pareto set of policies. Our algorithm starts from a warm-up stage. In this stage, n policies are randomly initialized, and each of them is optimized by multi-objective policy gradient (MOPG) (Algorithm 2 and Section 3.2) with one of n evenly distributed non-negative weights $\{\boldsymbol{\omega}_i\}$ ($\sum_j \omega_{i,j} = 1, 1 \leq i \leq n$) for a specified number of iterations. The resulting policies form the first generation of the policy population. The warm-up stage is crucial for the whole algorithm to get the initial policies out of the low-performance region, where the learning process is usually highly noisy and unpredictable.

Next, the algorithm proceeds with the evolutionary stage. In each generation, an analytical model for each policy in the population is learned from past reinforcement learning data to predict the expected improvement along each optimization weight (Section 3.3). This prediction model is then used to guide a selection optimization algorithm to select n policy-weight pairs (we call RL tasks), which are expected to improve the quality of the Pareto set the most (Section 3.4). Finally, the selected tasks are optimized by multi-objective policy gradient algorithms for a fixed number of iterations in parallel to produce the new offspring policies, which are used to update the policy population. For the population update, we adopt the performance buffer strategy (Schulz et al., 2018) to maintain the performance and diversity of the solutions. The evolutionary stage terminates when reaching the maximum number of generations. Through the whole evolutionary stage, an external Pareto archive is maintained to store all non-dominated intermediate policies and output as the approximated Pareto set when the evolutionary stage ends.

Once a discrete set of Pareto policies has been found, the algorithm conducts a Pareto analysis on the computed policies to identify different policy families, and then a continuous representation of the Pareto set is extracted by intra-family interpolation (Section 3.5).

3.2. Multi-Objective Policy Gradient

Given a policy $\pi_{\boldsymbol{\theta}}$ and a weight vector $\boldsymbol{\omega}(\sum_i \omega_i = 1)$, our multi-objective policy gradient worker aims to optimize the policy to maximize the weighted-sum reward $\mathcal{J}(\boldsymbol{\theta}, \boldsymbol{\omega})$:

$$\mathcal{J}(\boldsymbol{\theta}, \boldsymbol{\omega}) = \boldsymbol{\omega}^\top \mathbf{F}(\pi) = \sum_{i=1}^{m}\omega_i f_i(\pi) = \sum_{i=1}^{m}\omega_i J_i^{\pi}.$$

With our evolutionary learning algorithm, a policy will be selected to be optimized with different weights during the whole learning process. It is inefficient to simply modify the environment to return a scalar weighted-sum reward and optimize the policy by a single-objective policy gradient

Algorithm 1 Prediction-Guided MORL Algorithm

```
Input: #parallel tasks $n$, #warm-up iterations $m_w$, #task iterations $m_t$, #generations $M$.
Initialize population $\mathcal{P}$, external pareto archive EP, and RL history record $\mathcal{R}$.
▷ Warm-up Stage
Generate task set $\mathcal{T} = \{(\pi_i, \boldsymbol{\omega}_i)\}_{i=1}^{n}$ by random initial policies and evenly distributed weight vectors.
$\mathcal{P}' \leftarrow$ MOPG($\mathcal{T}, m_w, \mathcal{R}$) (Section 3.2)
Update $\mathcal{P}$ and EP with $\mathcal{P}'$.
▷ Evolutionary Stage
for $generation \leftarrow 1, 2, ..., M$ do
    Fit improvement prediction models $\{\boldsymbol{\Delta}^i\}$ for each policy in $\mathcal{P}$ from data in $\mathcal{R}$. (Section 3.3)
    $\mathcal{T} \leftarrow$TaskSelection($n, \mathcal{P}, \{\boldsymbol{\Delta}^i\}$, EP) (Section 3.4)
    $\mathcal{P}' \leftarrow$ MOPG($\mathcal{T}, m_t, \mathcal{R}$) (Section 3.2)
    Update $\mathcal{P}$ and EP with $\mathcal{P}'$.
end for
▷ Pareto Analysis Stage
Compute families in EP and construct a continuous Pareto representation. (Section 3.5)
Output: The continuous Pareto representation.
```

algorithm. With this naive approach, the value network trained with previous weights would be invalid for the new weight and would need to be trained from scratch. Therefore, we improve the single-objective policy gradient algorithm by extending the value function to be vectorized, which shares a similar strategy as applied in multi-objective Q-learning (Yang et al., 2019).

Specifically, the vectorized value function $\boldsymbol{V}^{\pi}(s)$ maps a state s to the vector of expected returns under the current policy π. The parameters of the value function are updated by a squared-error loss $\|\boldsymbol{V}^{\pi}(s) - \hat{\boldsymbol{V}}(s)\|^2$, where $\hat{\boldsymbol{V}}(s)$ is the target value. With this extension, the value function trained in the previous learning process can be directly adapted to optimize the same policy with the new weights.

To update the policy, the policy gradient is extended to be:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}, \boldsymbol{\omega}) &= \sum_{i=1}^{m} \omega_i \nabla_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta}) \\ &= \mathbb{E}\left[\sum_{t=0}^{T} \boldsymbol{\omega}^{\top} \boldsymbol{A}^{\pi}(s_t, a_t) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t)\right] \\ &= \mathbb{E}\left[\sum_{t=0}^{T} A^{\pi}_{\boldsymbol{\omega}}(s_t, a_t) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t)\right], \end{aligned} \tag{3}$$

where $\boldsymbol{A}^{\pi}(s_t, a_t)$ is the vectorized advantage function. In our extension, the new advantage function $A^{\pi}_{\boldsymbol{\omega}}(s_t, a_t)$ is simply represented as a weighted-sum scalarization of the advantage functions for individual objectives. Please see Appendix A.1 for more details.

Algorithm 2 MOPG

```
Input: task set $\mathcal{T}$, #iterations $m$, RL history record $\mathcal{R}$.
Initialize offspring population $P'$.
for each task $(\pi_i, \boldsymbol{\omega}_i) \in \mathcal{T}$ do
    Run multi-objective policy gradient for task ($\pi_i$, $\boldsymbol{\omega}_i$) for $m$ iterations by Eq. 3.
    Collect the result policy $\pi'_i$ in $P'$.
    Store $(\mathbf{F}(\pi_i), \mathbf{F}(\pi'_i), \boldsymbol{\omega}_i)$ in $\mathcal{R}$.
end for
Output: Offspring population $\mathcal{P}'$.
```

Such value network and policy gradient extension can be easily applied to most existing policy gradient methods. In our implementation, we choose to adapt the Proximal Policy Optimization (PPO) (Schulman et al., 2017) into our multi-objective weighted-sum version, where the clipped surrogate objective is applied to update the policy parameters, and the Generalized Advantage Estimation (Schulman et al., 2015) is used to compute the advantage function and the target values.

3.3. Policy Improvement Prediction Model

In this section, we present our prediction model for policy improvement. Given a policy π and a weight $\boldsymbol{\omega}$, the prediction model aims to predict the improvement of the objectives after applying the policy gradient on the policy π with the weight $\boldsymbol{\omega}$ for m_t iterations. However it is challenging due to the small amount of reinforcement learning history data we can collect during the learning process. Therefore, a concise analytical model with few parameters needs to be considered.

We propose a monotonic hyperbolic model based on an intuitive observation that the more weight put on one objective, the better that objective can be optimized. Formally speaking, if we run multi-objective policy gradient for a policy π with objectives $\mathbf{F}(\pi) = [f_1(\pi), f_2(\pi), ..., f_m(\pi)]$ with weights $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$ separately (where $\omega_{1,1} > \omega_{2,1}$), the resulting policies π_1 and π_2 should satisfy the monotonic property that $\Delta f_1(\pi_1) = f_1(\pi_1) - f_1(\pi) \geq f_1(\pi_2) - f_1(\pi) = \Delta f_1(\pi_2)$. Furthermore, the improvement function should be bounded on two sides. Based on these observations, we construct the following four-parameter hyperbolic model $\Delta^i_j(\omega_j)$ for each policy π_i and each objective f_j:

$$\Delta^i_j(\omega_j) = A \cdot \frac{e^{a(\omega_j - b)} - 1}{e^{a(\omega_j - b)} + 1} + c. \tag{4}$$

The function is illustrated in Figure 3 (middle left), where $\boldsymbol{\xi} = \{A, a, b, c\}$ are the four parameters that need to be determined for each model. In order to fit the parameters, we record the objective improvements of reinforcement learning every m_t iterations in a record data structure $\mathcal{R}$. Each entry in $\mathcal{R}$ is a triplet $(\mathbf{F}(\pi), \mathbf{F}(\pi'), \boldsymbol{\omega})$, where $\mathbf{F}(\pi)$ and

$\mathbf{F}(\pi')$ are the objectives for the policy before and after being optimized by reinforcement learning for m_t iterations, and $\boldsymbol{\omega}$ is the optimizing weight. As shown in Figure 3, $\mathcal{R}$ is a directed graph (precisely a directed rooted forest) storing the full RL optimization history for each policy.

In each generation, for each policy π_i, the data $\{(\boldsymbol{\omega}, \Delta\mathbf{F})\} = \{(\boldsymbol{\omega}, \mathbf{F}(\pi') - \mathbf{F}(\pi))\}$ in the neighborhood of the policy π_i (*i.e.,* $\|\mathbf{F}(\pi) - \mathbf{F}(\pi_i)\| < \delta\|\mathbf{F}(\pi_i)\|$) is collected, and a nonlinear least-square regression is applied to fit the parameters of the hyperbolic model.

3.4. Prediction-Guided Optimization for Task Selection

Taking into account the hypervolume (Eq. 1) and sparsity (Eq. 2) metrics, we propose a prediction-guided algorithm for task selection from first principles.

In each generation, our algorithm aims to select the most important tasks (pairs of policy and weight) that can best improve the Pareto metrics. Specifically, the algorithm needs to select n tasks $\mathcal{T}_i$ to be processed by multi-objective policy gradient for m_t iterations. Here each task $\mathcal{T}_i$ is composed by a pair of policy π_i from current population $\mathcal{P}$ and an optimization weight $\boldsymbol{\omega}_i$. The selected tasks seek to maximize a weighted mixture metric $\mathcal{H}(\boldsymbol{F}(\texttt{EP}^*)) + \alpha\mathcal{S}(\boldsymbol{F}(\texttt{EP}^*))$ ($\alpha < 0$ for minimizing the sparsity metric), where $\texttt{EP}^*$ is the new Pareto set after inserting the offspring policies from those tasks. Guided by the prediction models trained for each policy, we can predict the expected objectives of the new offspring policy for each task as $\mathbf{F}(\pi_i) + \boldsymbol{\Delta}^i(\boldsymbol{\omega}_i)$, and can formulate this optimization problem as:

$$\begin{aligned} \max_{\mathcal{T}=\{(\pi_i,\omega_i)\}_{i=1}^n} \quad & \mathcal{Q}(\texttt{EP}, \mathcal{T}) = \mathcal{H}(P) + \alpha\mathcal{S}(P) \qquad (5) \\ \text{with } P &= \mathbf{F}(\texttt{EP}^*) \\ &= \texttt{Pareto}(\mathbf{F}(\texttt{EP}) \cup \{\mathbf{F}(\pi_i) + \boldsymbol{\Delta}^i(\boldsymbol{\omega}_i)\}), \end{aligned}$$

where $\texttt{EP}$ is the current Pareto archive, and $\texttt{Pareto}$ is the function computing the Pareto front from a set of objectives.

The optimization problem in Eq. 5 is a mixed-integer programming problem, which is difficult to solve directly. Therefore, we approximate it by discretizing the continuous weight to K candidate sample weights (Figure 3 middle left) and instead solve a knapsack problem: given $K \times |\mathcal{P}|$ candidate points in the objective space, we want to select n of them to maximize the mixture metric after inserting them into the current Pareto archive $\texttt{EP}$. Although in the two objective case, it can be solved by dynamic programming in polynomial time complexity, exactly solving the knapsack problem in general is an NP-hard problem. Therefore, in order to improve the generalizability of the algorithm, we adopt a greedy algorithm (Algorithm 3). Our greedy algorithm maintains a virtual policy set $\texttt{EP}^*$ for the predicted Pareto archive. It then iteratively selects the task that best improves the Pareto metric of $\texttt{EP}^*$ and then updates $\texttt{EP}^*$ by inserting the predicted offspring policy of the selected task.

Algorithm 3 Prediction-Guided Task Selection

```
Input: #tasks $n$, population $\mathcal{P}$, improvement prediction models $\{\boldsymbol{\Delta}^i\}$, Pareto archive $\texttt{EP}$.
Initialize task set $\mathcal{T}$ and virtual Pareto archive $\texttt{EP}^* = \texttt{EP}$.
for $i \leftarrow 1, 2, ..., n$ do
  Initialize task $\mathcal{T}_i \leftarrow$ None
  for each $\pi_i \in \mathcal{P}$ and $\boldsymbol{\omega} \in$ candidate weights do
    if $(\pi_i, \boldsymbol{\omega})$ has not been selected and
        $\mathcal{Q}(\texttt{EP}^*, (\pi_i, \boldsymbol{\omega})) > \mathcal{Q}(\texttt{EP}^*, \mathcal{T}_i)$ then
      $\mathcal{T}_i \leftarrow (\pi_i, \boldsymbol{\omega})$
    end if
  end for
  Append task $\mathcal{T}_i$ into $\mathcal{T}$.
  Update $\texttt{EP}^*$ by inserting the predicted offspring of $\mathcal{T}_i$.
end for
Output: Selected task set $\mathcal{T}$.
```

3.5. Continuous Pareto Representation

Once a set of Pareto optimal policies is computed from the evolutionary stage, we conduct a Pareto analysis to analyze the structure of policy parameters on the Pareto front.

Since the deep neural network policies are not linearly co-related, we use t-SNE (Maaten & Hinton, 2008), which is a standard nonlinear dimensionality reduction method, to embed the high dimensional policy parameter space into a lower dimensional space for better visualization. In our case, we found mapping to two dimensional space to work well.

For the purpose of dimensionality reduction, there are also other available methods (e.g., LLE, PCA, Isomap). We choose t-SNE due to its better visualization effect. We provide a comparison with other dimensionality reduction methods in Appendix E.5.

Once the embedding is generated, we use k-means to cluster the reduced policies into several families as illustrated in Figure 3 (right). As expected, the whole Pareto-optimal set is composed from several disjoint policy families, and each family is responsible for a continuous segment on the Pareto front. A continuous Pareto representation is then constructed by linearly interpolating the policies inside the same family. For any target objectives on the continuous Pareto front approximation, we first identify which policy family can cover those target objectives and then linearly interpolate the parameters of the nearby policies. Although the deep neural networks are not linearly co-related, we surprisingly find that by computing a dense Pareto approximation set and conducting the Pareto analysis, such intra-family interpolation works successfully, which is demonstrated by the results in Section 4.3.2.

4. Experiments

4.1. Benchmark Problems

In order to benchmark our proposed algorithm, we design seven multi-objective RL environments with continuous action space based on Mujoco (Todorov et al., 2012). Our benchmark problems include six two-objective environments and one three-objective environment:

HalfCheetah-v2: Two objectives: forward speed, energy efficiency ($\mathcal{S} \subseteq \mathbb{R}^{17}$, $\mathcal{A} \subseteq \mathbb{R}^{6}$).

Hopper-v2: Two objectives: forward speed, jumping height ($\mathcal{S} \subseteq \mathbb{R}^{11}$, $\mathcal{A} \subseteq \mathbb{R}^{3}$).

Swimmer-v2: Two objectives: forward speed, energy efficiency ($\mathcal{S} \subseteq \mathbb{R}^{8}$, $\mathcal{A} \subseteq \mathbb{R}^{2}$).

Ant-v2: Two objectives: x-axis speed, y-axis speed ($\mathcal{S} \subseteq \mathbb{R}^{27}$, $\mathcal{A} \subseteq \mathbb{R}^{8}$).

Walker2d-v2: Two objectives: forward speed, energy efficiency ($\mathcal{S} \subseteq \mathbb{R}^{17}$, $\mathcal{A} \subseteq \mathbb{R}^{6}$).

Humanoid-v2: Two objectives: forward speed, energy efficiency ($\mathcal{S} \subseteq \mathbb{R}^{376}$, $\mathcal{A} \subseteq \mathbb{R}^{17}$).

Hopper-v3: Three objectives: forward speed, jumping height, energy efficiency ($\mathcal{S} \subseteq \mathbb{R}^{11}$, $\mathcal{A} \subseteq \mathbb{R}^{3}$).

The detailed description of the environment designs can be found in Appendix C.

4.2. Experiment Setup

We implement our prediction-guided evolutionary learning algorithm as described in Section 3, and implement five baseline algorithms for comparison. [1]

RA: The Radial Algorithm assigns a set of weights and runs reinforcement learning to optimize the policies for each weight separately. (Parisi et al., 2014)

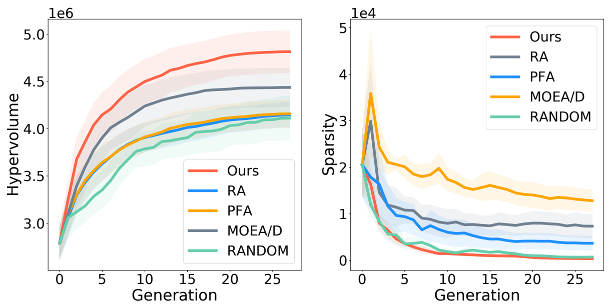
PFA: In the evolutionary stage, we gradually fine tune the weight of the RL to cover the whole Pareto front, which is an adaptation of original PFA algorithm (Parisi et al., 2014) to DRL setting.

MOEA/D: Multi-Objective Evolutionary Algorithm based on Decomposition (Zhang & Li, 2007) decomposes the problem into subproblems by different weights and solves them in a collaborative way.

RANDOM: A random selection strategy is designed to uniformly sample RL task in each generation.

META: A meta-learning based MORL method (Chen et al., 2018) trains a meta policy and then adapts the meta policy to the policies for different preferences in a few iterations.

[1]The code can be found at https://github.com/mit-gfx/PGMORL



(a) Hypervolume (b) Sparsity

Figure 4. **The learning curves of our algorithm and baseline algorithms on Walker2d-v2.** The x-axis is the generation, the y-axis is the metric and the shadow area is the standard deviation. The Hypervolume at generation 0 is measured after the warm-up stage. The learning curve of META is not plotted as its metrics can only be measured during the final adaptation stage. (a) Hypervolume metric (higher is better). (b) Sparsity metric (lower is better).

To fairly compare the baseline algorithms to ours, we implement the first four baselines in a common framework with our proposed algorithm and apply the same population strategy and external Pareto archive to them. For the META, our implementation is based on the codebase (Deleu, 2018) which implements Model-Agnostic Meta-Learning (Finn et al., 2017) and generates the Pareto approximation by adapting the meta-policy to N uniformly sampled weights (we set N as a large number compared to the number of solutions in other methods). Furthermore, we set all the shared hyperparameters to be the same and run all algorithms with same amount of simulation steps. More details about the experiment setup are described in Appendix D.1.

4.3. Results

We test the performance of our algorithm and all the baselines on the proposed benchmark problems. The training details and parameters are reported in Appendix D.2. We provide more visual results in the supplementary video. [2]

4.3.1. Pareto Quality Comparison

We first use the hypervolume metric (Eq. 1) and the sparsity metric (Eq. 2) to compare the quality of the computed Pareto set approximations. We run each algorithm on each problem for six times and report the average metrics in Table 1. The training curves on Walker2d-v2 problem are shown in Figure 4. We provide the learning curve and Pareto front comparison results on other problems in Appendix E.1.

The results in Table 1 demonstrate that our proposed algorithm outperforms all the baselines on most benchmark

[2]https://people.csail.mit.edu/jiex/papers/PGMORL/video.mp4

Table 1. **Evaluation of our algorithm and baseline algorithms on the proposed benchmark problems.** We run all algorithms on each problem for 6 runs and report the average Hypervolume (Hv) and Sparsity (Sp) metrics. Bold number is the best in each row.

EXAMPLE	METRIC	OURS	RA	PFA	MOEA/D	RANDOM	META
HALFCHEETAH-V2	HV $(\times 10^6)$	**5.77**	5.66	5.75	5.61	5.69	5.18
	SP $(\times 10^3)$	**0.44**	15.87	3.81	16.96	1.09	2.13
HOPPER-V2	HV $(\times 10^7)$	2.02	1.96	1.90	**2.03**	1.88	1.25
	SP $(\times 10^4)$	**0.50**	5.99	3.96	2.73	1.20	4.84
SWIMMER-V2	HV $(\times 10^4)$	**2.57**	2.33	2.35	2.42	2.38	1.23
	SP $(\times 10^1)$	**0.99**	4.43	2.49	5.64	1.94	2.44
ANT-V2	HV $(\times 10^6)$	**6.35**	5.98	6.23	6.28	5.54	2.40
	SP $(\times 10^4)$	**0.37**	5.50	1.56	1.97	1.13	1.56
WALKER2D-V2	HV $(\times 10^6)$	**4.82**	4.15	4.16	4.44	4.11	2.10
	SP $(\times 10^4)$	**0.04**	0.74	0.37	1.28	0.07	2.10
HUMANOID-V2	HV $(\times 10^7)$	4.64	3.53	3.70	**4.65**	3.21	-
	SP $(\times 10^4)$	**0.19**	4.50	0.38	3.82	0.42	-
HOPPER-V3	HV $(\times 10^{10})$	**3.74**	3.50	-	3.64	3.36	2.15
	SP $(\times 10^3)$	**0.03**	0.61	-	0.58	0.27	12.48

problems in both metrics. The training curves show that our prediction-guided algorithm is able to select the important reinforcement learning tasks to improve the Pareto quality much more efficiently than the baseline methods.

RA archives high-performance solutions in some regions on the Pareto but the solutions are spread sparsely in the performance space because RA assigns all computing resources into optimizing for those pre-selected weights.

PFA generates denser Pareto approximations than RA because it finetunes the optimization weights to cover the whole weight range. However, because it blindly changes a weight to its neighboring weight, a good policy is unable to transfer its knowledge to wider range of weights. Therefore, it can only recover some pieces of the Pareto front. In our algorithm, a policy can be optimized along the whole Pareto front as long as it can improve the Pareto quality. Moreover, PFA is hard to extend to the three-objective case as the sequence of weights in three dimension is undefined.

MOEA/D is the most competitive baseline on hypervolume metric as it periodically shares the better solutions across subproblems. However, it also suffers from high sparsity.

RANDOM computes the densest Pareto approximation in all baselines as it distributes the RL tasks evenly onto every weight and policy. However, the random task selection strategy leads to the low-performance of the computed Pareto front, which is reflected by the low hypervolume metric.

META computes a compromise policy family that can perform well for every preference but not achieve the optimal

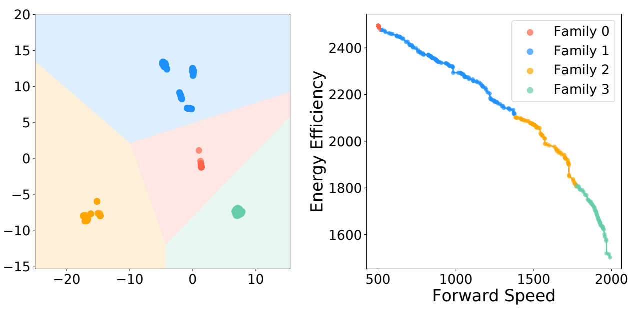
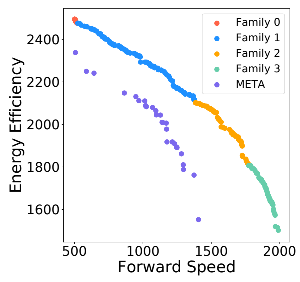


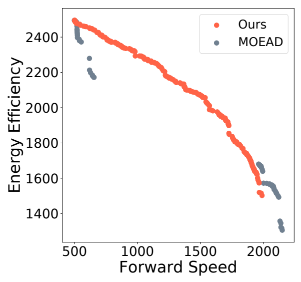
Figure 5. **Pareto analysis for Walker2d-v2 problem.** (Left) The policy families identified by t-SNE and k-means. (Right) Visualization of the families in objective space. The curve going through each family is the continuous Pareto approximation.

control. In contrast, the multi-family representation in our method allows a much better Pareto set approximation. We discuss it more in Section 4.3.3. The META results on Humanoid-v2 are not reported, since in our experiments, META is not able to generate a Pareto front in the first quadrant.

In summary, none of the baseline algorithms distribute the computing resource to the RL tasks that best improve the Pareto quality. In contrast, by using the policy improvement prediction model, our algorithm is able to identify which regions on the Pareto are already near optimal and which regions can still be improved. Therefore the best tasks can be selected and a high-quality Pareto can be generated efficiently and effectively.



(a) **META** v.s. **Ours**



(b) **MOEA/D** v.s. **Ours**

Figure 6. **Pareto front comparison on Walker2d-v2 problem.** (a) **META** v.s. **Ours**. The multi-family representation in our method helps achieve better control for different preferences. (b) **MOEA/D** v.s. **Ours**. Our algorithm is unable to recover the Pareto on the bottom right corner due to the *long-term local minima problem*.

4.3.2. Pareto Analysis Results

For each Pareto set solution computed by our evolutionary learning algorithm, we conduct the Pareto set analysis described in Section 3.5 to identify different families in the solution set. The family identification for Walker2d-v2 problem is illustrated in Figure 5. For Walker2d-v2, the whole Pareto set is split into four families in the parameter space. The Pareto front corresponding to each policy family comprise a continuous segment in the performance space.

We further construct a continuous Pareto representation for each family as described in Section 3.5. The constructed continuous Pareto front is shown in Figure 5 (right). To test the accuracy of our continuous representation, we sample points on the continuous Pareto front, and evaluate the relative error between the desired objectives and the objectives of the interpolated policy. The relative errors on most problems are smaller than 1%. We further evaluate the relative error of interpolating the policies from different families and validate that the different families are disjoint in the parameter space. The detailed Pareto analysis and interpolation results are reported in Appendix E.2. We also demonstrate in the supplementary video that the policies in different families show different behaviors (e.g., gait patterns), and such different behaviors help achieve the optimal control under the different objective preferences.

For the purpose of dimensionality reduction, there are also other available methods (e.g., LLE, PCA, Isomap). We choose t-SNE due to its much clearer distinguishing effect. Furthermore, in our experiments, t-SNE shows its robustness to the parameters. We provide a discussion of parameters and comparison with other dimensionality reduction methods in Appendix E.4 and E.5.

4.3.3. Meta Policy or Multi-Family?

By comparing META and our algorithm on Walker2d-v2 (Figure 6(a)), we empirically show that a typical Pareto set is composed from a set of disjoint policy families. Therefore, it is natural to compare this representation to meta policy. Meta policy method provides generalizability and represents the Pareto solutions by a single policy family. However, it sacrifices the optimality of the control. On the contrary, the multi-family representation can help achieve optimal control, but a hard switch between policy families is required while changing the preference on the boundary of each family.

4.3.4. Failure Case

As shown in Figure 6(b), in Walker2d-v2, although our algorithm can generate a much denser and higher-quality Pareto set than MOEA/D, we are unable to recover the bottom right corner of the Pareto front.

This problem is caused by the fact that in order to reach the bottom right corner of the Pareto front, the policy can be trapped in local minima for an extended time before moving towards a better solution. We call this the *long-term local minima problem*. Furthermore, as our algorithm predicts potential improvements based on the RL history data, it can fail to predict potential improvements for such policies. This is, indeed, a trade-off in our algorithm design: spending more time on the local minima area in order to reach better performance, or spending time on optimizing the other regions of the Pareto front.

5. Related Work

5.1. Multi-Objective Reinforcement Learning

Most of the previous MORL work can be classified into three categories. Single-policy methods convert the multi-objective problem into a single-objective problem (Gábor et al., 1998; Mannor & Shimkin, 2002) using a scalarization function. The main drawback of these methods is that the preference weights must be set in advance. Multi-policy methods compute a set of policies to approximate the real Pareto-optimal set (Parisi et al., 2014; Natarajan & Tadepalli, 2005; Li et al., 2019). The main bottleneck of these methods is the high computational requirement. This prevents these methods from finding dense Pareto solutions for complex control problems. Our work falls into this category but resolves this limitation by dynamically allocating computing resource by a prediction-guided selection optimization. Meta policy methods and single universal policy methods either compute a meta policy and adapt it to different preferences, or directly generate output control conditioned on input preference weights (Chen et al., 2018; Castelletti et al., 2011; Yang et al., 2019; Abels et al., 2019).

Those methods share the same shortcomings as the meta policy methods as discussed in Section 4.3.3. Finally, most methods in this class still only work for problems with discrete action space and simple mechanisms (*e.g.,* deep sea treasure environment).

5.2. Multi-Objective Evolutionary Algorithms

Previous work in MOEA uses various evolutionary algorithms to find the Pareto set by genetic operations such as mutation and crossover (Deb, 2011). However, such black-box optimization methods are highly inefficient in finding optimal solutions especially when the parameter space is extremely large (*e.g.,* thousands of dimensions for a neural-network). Thus, we replace the evolutionary algorithms by reinforcement learning but borrow the ideas from this stream of work to evolve the Pareto set. For example, we decompose the multi-objective optimization problem into several single-objective sub-problems by weighted-sum scalarization, which has been previously proposed by MOEA/D (Zhang & Li, 2007). Besides, we use the performance buffer strategy (Schulz et al., 2018) to maintain a large amount of diverse and high-quality population as optimization candidates in next generation, which is also similar to NGSA-II (Deb et al., 2002) that performs non-dominated sort and crowding sort to compute better populations.

6. Conclusion and Discussion

In this work, we show that an effective representation for obtaining the best performance trade-offs for multi-objective robot control is a Pareto set, which is composed from different policy families. We present an efficient algorithm to compute such Pareto representations. A prediction-guided evolutionary learning algorithm is first employed to find a high-quality set of policies on the Pareto set. Then we conduct a Pareto analysis on the computed Pareto-optimal policies to construct a continuous Pareto representation. Furthermore, we design a set of multi-objective RL environments with continuous action space, and conduct extensive experiments to validate the effectiveness of our algorithm.

There are several directions which can be explored in the future. First, we believe that the learning efficiency could be further improved by sharing the sampled trajectories through the whole learning process. Second, it is desired to develop a more robust model to solve the *long-term local minima problem*. Finally, it is worthwhile to apply this method to solve multi-objective control problems for real-world robots.

Acknowledgements

We thank the anonymous reviewers for their helpful comments in revising the paper. This work is supported by Intelligence Advanced Research Projects Activity (grant No. 2019-19020100001), Defense Advanced Research Projects Agency (grant No. N66001-15-C-4030), and the National Science Foundation (grant No. CMMI-1644558).

References

Abels, A., Roijers, D., Lenaerts, T., Nowé, A., and Steckelmacher, D. Dynamic weights in multi-objective deep reinforcement learning. In *International Conference on Machine Learning*, pp. 11–20, 2019.

Castelletti, A., Pianosi, F., and Restelli, M. Multi-objective fitted q-iteration: Pareto frontier approximation in one single run. In *2011 International Conference on Networking, Sensing and Control*, pp. 260–265. IEEE, 2011.

Chen, X., Ghadirzadeh, A., Björkman, M., and Jensfelt, P. Meta-learning for multi-objective reinforcement learning. *arXiv preprint arXiv:1811.03376*, 2018.

Deb, K. Multi-objective optimisation using evolutionary algorithms: an introduction. In *Multi-objective evolutionary optimisation for product design and manufacturing*, pp. 3–34. Springer, 2011.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2): 182–197, 2002.

Deleu, T. Model-Agnostic Meta-Learning for Reinforcement Learning in PyTorch, 2018. Available at: https://github.com/tristandeleu/pytorch-maml-rl.

Finn, C., Abbeel, P., and Levine, S. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *International Conference on Machine Learning (ICML)*, 2017. URL `http://arxiv.org/abs/1703.03400`.

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

Li, K., Zhang, T., and Wang, R. Deep reinforcement learning for multi-objective optimization. *arXiv preprint arXiv:1906.02386*, 2019.

Maaten, L. v. d. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov): 2579–2605, 2008.

Mannor, S. and Shimkin, N. The steering approach for multi-criteria reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 1563–1570, 2002.

Natarajan, S. and Tadepalli, P. Dynamic preferences in multi-criteria reinforcement learning. In *Proceedings of the 22nd international conference on Machine learning*, pp. 601–608, 2005.

Parisi, S., Pirotta, M., Smacchia, N., Bascetta, L., and Restelli, M. Policy gradient approaches for multi-objective sequential decision making. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pp. 2323–2330, July 2014. doi: 10.1109/IJCNN.2014.6889738.

Riquelme, N., Von Lücken, C., and Baran, B. Performance metrics in multi-objective optimization. In *2015 Latin American Computing Conference (CLEI)*, pp. 1–11, Oct 2015. doi: 10.1109/CLEI.2015.7360024.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

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

Schulz, A., Wang, H., Grinspun, E., Solomon, J., and Matusik, W. Interactive exploration of design trade-offs. *ACM Trans. Graph.*, 37(4), July 2018. ISSN 0730-0301. doi: 10.1145/3197517.3201385. URL https://doi.org/10.1145/3197517.3201385.

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

Yang, R., Sun, X., and Narasimhan, K. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. In Wallach, H., Larochelle, H., Beygelzimer, A., dAlché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 14610–14621. Curran Associates, Inc., 2019.

Zhang, Q. and Li, H. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731, 2007.

Zitzler, E. and Thiele, L. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, Nov 1999. ISSN 1941-0026. doi: 10.1109/4235.797969.