Repulsive Pheromones-Based Action Selection for Multi-Objective Reinforcement Learning in Continuous State and Action Spaces

Anonymous authors
Paper under double-blind review

Abstract

Many real-world control tasks contain multiple conflicting objectives that need to be optimized simultaneously. Multi-objective reinforcement learning (MORL) is one method to solve these problems. One challenge in multi-objective environments with large-dimensional action spaces, such as those in robotics, is that it becomes difficult to properly explore the entire search space while training. This paper introduces four novel action selection strategies within the MORL/D framework for multi-objective control tasks which are designed to improve the exploration of the action space while training, and are compared in six bi-objective simulated control tasks. Two of the proposed approaches, LI-τ and LD-τ, which promote more exploration at the end and at the beginning of training respectively, had superior performance in the majority of the tested environments, and an adaptive approach which adjusts exploration depending on whether the performance has improved in the previous episode had the best performance in the largest action spaces which were tested. The experimental results showed that by promoting more exploration at different stages of the learning process, agents are better able to explore the action space, resulting in better quality final Pareto-optimal fronts (POFs), and are also often able to find a better-quality approximation of the POF earlier on in the training process over the baseline method.

1 Introduction

Many real-world problems contain multiple objectives which need to be optimized simultaneously. These objectives are often in conflict with each other, where improving in one objective worsens the results in another, such as in robotics control tasks, where one objective could be to move to a specific point as quickly as possible while another is to minimize the control cost. Rather than having a single optimal solution, multi-objective problems (MOPs) have a set of optimal solutions, where each solution corresponds to a different trade-off between the objectives.

Multi-objective reinforcement learning (MORL) is one method for solving MOPs, and has been successfully applied to problems in robotics (Huang et al., 2019; Hayes et al., 2022), traffic signal control (Saiki and Arai, 2023), smart buildings (Huynh et al., 2025), journey planning (Chen et al., 2023), and water management (Castelletti et al., 2013). MORL algorithms learn by interacting with the environment and earning a reward based on the objective functions. The goal of a MORL agent is to learn a policy which maximizes its cumulative reward.

Meta-heuristics have been used in combination with machine learning (ML) methods to achieve better results over using just the ML method alone (Kharitonov et al., 2025). These hybrid approaches are applicable to real-world problems, such as decision-making for smart manufacturing (Kharitonov et al., 2025) and a protein tertiary structure problem (Venske et al., 2024). It is noted by Kharitonov et al. (2025) that work related to combining reinforcement learning (RL) and meta-heuristics for decision-making tasks is currently limited. Work combining RL with meta-heuristics for robotic control problems is also similarly limited, although Amirabadi and Fard (2024) combined RL with a meta-heuristic for a non-linear control problem.

One of the main difficulties with MOPs, as well as continuous action spaces, is the resulting increase in the size of the search space which makes it difficult for agents to fully explore the action space while training. This work proposes the use of a repulsive-pheromones based meta-heuristic during the training process of an MORL method in order to better explore the action space in continuous action spaces, based on work done by Felten et al. (2022) in discrete action spaces. In summary, the main contributions of this work are as follows:

1. The introduction of four novel action selection strategies for multi-objective problems with continuous action spaces within the MORL/D framework (Felten et al., 2024). These action selection strategies focus on improving the exploration of actors by adapting the idea of the repulsive-pheromones action selection approach used by Felten et al. (2022) to control problems with continuous action spaces.

2. A comparison of the proposed approaches in a set of bi-objective robotic control environments with varying action space sizes to compare their scalability from simple to complex tasks.

3. A comparison of the introduced action selection strategies with the baseline action selection method in MORL/D to show that improving the exploration of actors leads to better performance in complex environments.

4. Make the open-source code available for future work at: Anonymous Github Repository Link

The remainder of this paper is organized as follows: Section 2 gives a summary of related work as well as background details on MORL, Section 3 gives the details on the proposed action selection approaches, Section 4 gives the implementation details, Section 5 details the experimental setup, Section 6 gives the experimental results, and Section 7 gives the conclusion and ideas for future work.

2 Background and Related Work

In some cases, the desired weights for each objective of an MOP may be known beforehand and the scalarized reward can be used as a single objective. However, oftentimes in real-world problems decision makers (DMs) do not know their preferences beforehand, or their preferences change over time. Therefore, it is often desired to generate a set of optimal trade-off solutions according to the Pareto-dominance relation. Pareto-dominance relation states that one solution x_1 dominates another x_2 if and only if x_1 is strictly better than x_2 in at least one objective, and is equal to or better than x_2 in every other objective Keller (2019). Formally, the set of solutions in the decision space which are non-dominated with each other is referred to as the Pareto-optimal set (POS), and the corresponding set of solutions in the objective space is called the Pareto-optimal front (POF). The DM can then select the desired policy from the set of Pareto-optimal solutions.

The goal of multi-objective reinforcement learning (MORL) is to generate a set of policies which approximates the true POF. MORL algorithms learn policies by interacting with the environment and earning a reward for each objective, where a higher reward indicates better performance. Single-policy MORL algorithms use the scalarized reward to train the policy of a single RL agent (Liu et al., 2015). However, they need to run multiple times to generate an approximate POF, taking up additional computational resources and time. Multi-policy RL addresses this limitation by generating multiple Pareto-optimal policies in a single run (Liu et al., 2015).[1]

Lu et al. (2023) introduce CAPQL, where a strongly concave term is added to the immediate reward function in order to improve the found POF of linear scalarization approaches. Although this method was shown to have good performance it also introduces a highly problem-dependent augmentation strength hyper-parameter which needs to be adjusted for each problem. Prediction-Guided MORL (PGMORL) is introduced by Xu et al. (2020), which uses an improvement prediction model for each policy to select the best policy-weight pairs to be used when updating the population and prediction models. PGMORL also introduces additional

[1]Multi-policy MORL algorithms have been introduced for environments with discrete state and action spaces (Moffaert and Nowé, 2014; Van Moffaert et al., 2013; Roijers et al., 2018), and continuous action spaces and discrete state spaces (Yang et al., 2019). Since the focus of this paper is on problems with continuous state and action spaces, these methods will not be discussed in detail.

hyper-parameters, however these are much less problem-dependent compared to CAPQL, and the same values are used for all but one of their tested environments.

A model-based approach is introduced by Alegre et al. (2023), which uses Generalized Policy Improvement (GPI) to prioritize different parts of the task in order to improve the sample-efficiency of RL. Although the model-based approach can improve sample-efficiency by reducing the number of environment interactions required, learning a model can be difficult for complex problems when the model lacks enough data. Additionally, a model-based RL approach was shown to have worse performance over model-free RL in real-world robotic control tasks by Valencia et al. (2023).

Reymond et al. (2022) introduce Pareto-Conditioned Networks (PCN), which use a single NN to represent all non-dominated policies. Although PCN could be adapted for continuous action spaces, only environments with discrete action spaces were used for experimentation by Reymond et al. (2022).

The main benefits of our proposed action selection approaches over recent work are that the exploration of the action space is heavily emphasized during training by getting agents to take actions which differ from those they have recently taken, no model of the environment is required, it is designed for environments with continuous action spaces which are often the case in real-world control problems, and we avoid introducing any additional problem-dependent hyperparameters.

3 Methodology

Multi-objective reinforcement learning based on decomposition (MORL/D) (Felten et al., 2024) divides the MOP into multiple SOPs and uses weighted sum scalarization to weigh each of the objectives. Each SOP is solved using soft actor-critic (SAC), where actors have a stochastic policy. SAC optimizes both the maximum reward and the maximum entropy in order to improve the robustness of the policy (Haarnoja et al., 2019). SAC is an off-policy approach which samples from an episode replay buffer while training in order to improve sample-efficiency by re-using old episodes. The SAC actors use a shared episode replay buffer for all policies in the population, where each policy samples uniformly from the buffer. In Felten et al. (2024), this version of MORL/D was shown to have comparable or superior performance to state-of-the art approaches, and so it was selected to be the baseline method for the action selection methods proposed in this paper.

In order to fully explore the possible set of Pareto-optimal solutions, the objective weights are adapted over time using Pareto-simulated annealing (PSA) (Czyzżak and Jaszkiewicz, 1998). Since the performance of an agent can worsen once its objective weights are updated, an external archive is used to store all non-dominated solutions which have been found. Once training is over, the archive represents the found POF. For more details on the implementation of MORL/D, refer to Appendix B.

The proposed action selection approaches are based on the work in Felten et al. (2022). In Felten et al. (2022), different meta-heuristics were used to improve the exploration of RL in environments with discrete action spaces by encouraging agents to select actions which are different from previously selected actions. The Ant-Q algorithm (Gambardella and Dorigo, 1995) is used in as the metaheuristic in our work, based on the results from Felten et al. (2022). The main idea is for agents to leave pheromones on the path which they have taken, and when selecting the next action to take, paths with higher pheromone levels are avoided. The pheromones evaporate after every episode, so that over time actions become available again once some time has passed.

3.1 Repulsive Pheromones-Based Metaheuristic

In order to adapt the pheromone-based approach in Felten et al. (2022) to continuous action spaces, a number of changes had to be made to the action selection process. First, a buffer of the most recently-selected actions is maintained. At each timestep, multiple actions are sampled from the returned probability distribution of each SAC actor. Each of these actions are then evaluated using a heuristic, and the action with the best (in this case the highest) heuristic value is taken.

In order to encourage the exploration of the action space while training, the heuristic function considers both the distance from the sampled action to an action a in the action buffer A, and the amount of time which

has passed since a was selected. The heuristic function $H(x)$ where x is the sampled action being evaluated, is given by:

$$H(x) = \min_{a \in A} \left(dist(x, a) + \frac{1}{\rho(a)} \right), \tag{1}$$

where $dist(x, a)$ is the Euclidean distance from x to a, and ρ is the pheromone value of a. The ρ value for an action is initially set to 1 when it is first added to the buffer.

After each episode, ρ is decreased for all actions in the buffer using:

$$\rho = \rho(1 - \tau). \tag{2}$$

The heuristic for each sampled action is minimized so that it represents the "worst case" heuristic value, and then when selecting an action the one with the highest heuristic value is taken (i.e. best of the worst-case for each of the selected actions). By modifying the value of τ, more or less weight will be put on how recently a given action occurred. More specifically, a high τ leads to more exploration, while a small τ leads to more exploitation.

3.2 Action Selection Strategies

Four methods for selecting a τ value were implemented and compared to the baseline approach, and are called the static-τ, linearly decreasing τ (LD-τ), linearly increasing τ (LI-τ), and adaptive-τ.

In the static-τ approach, the value of τ is set to a static value which stays the same as long as the model is training. This approach is the simplest of the introduced methods to implement, with the downside of τ becoming an additional hyper-parameter. The value of τ was kept the same for all environments in our work.

In the LD-τ approach, the value of τ is initialized to 1 and is decreased at a fixed rate across the training episodes until it reaches a value just above zero.[2] By starting with a high value of τ, the pheromone values are initially evaporated rapidly, so there is more focus on exploration. By the end of training, the pheromone values are decreased slowly, leading the actor to focus on selecting actions which are far from the most recently selected actions.

The LI-τ approach is the opposite of LD-τ, where a value of τ starts just above 0 and increases toward a value of 1 with a fixed step size.

In the adaptive-τ approach, the value of τ is adjusted using performance feedback rather than at a fixed interval. More specifically, if the performance improves i.e. the reward from the most recent episode is better than the previous one, then τ is decreased. If not, then τ is increased in order to increase exploration.

The value of τ at timestep t is updated using:

$$\tau_t = \begin{cases} \tau_{t-1} & \text{if static-}\tau \\ \tau_{t-1} - (1/T) & \text{if LD-}\tau \text{ or adaptive-}\tau \text{ (improves)} \\ \tau_{t-1} + (1/T) & \text{if LI-}\tau\text{, or adaptive-}\tau \text{ (degrades)} \end{cases}, \tag{3}$$

where τ_{t-1} is the previous value of τ, and T is the total number of steps used for training. The value of τ is adjusted after an episode is finished. Since episodes have varying lengths and tend to be shorter at the beginning, this will lead to τ being updated more frequently at the beginning of a run (i.e. before it has found a good control policy) and less frequently once the policy lasts for the maximum allowed duration.

4 Implementation Details

The hyper-parameters for MORL/D are the same as those given in Felten et al. (2024). The default action selection strategy is to sample one action from the actor's outputted probability distribution, and this is the baseline method which is used for comparison. Although other action selection methods have been introduced for RL agents such as greedy or epsilon-greedy, these are typically used for deterministic polices. Since SAC

[2]The value of τ is set to be greater than zero so that the pheromone is decreased at each iteration.

Table 1: Environment action and observation space details: note that all observations are within $(-\infty, \infty)$

Environment	Action Space	Observation Space	Objective 1	Objective 2
MO-Hopper	3, within $[-1.0, 1.0]$	11	Move along x-axis	Move along z-axis
MO-HalfCheetah	6, within $[-1.0, 1.0]$	17	Move along x-axis	Control cost
MO-Walker2D	6, within $[-1.0, 1.0]$	17	Move along x-axis	Control cost
MO-Ant	8, within $[-1.0, 1.0]$	105	Move along x-axis	Move along y-axis
MO-Swimmer	2, within $[-1.0, 1.0]$	8	Move along x-axis	Control cost
MO-Humanoid	17, within $[-0.4, 0.4]$	348	Move along x-axis	Control cost

uses a stochastic policy, these methods are not used in this case, and therefore the standard action-selection method of sampling one action from the actor's distribution is used as the baseline method for comparison. This method is denoted as "Original MOSAC" for the remainder of this paper.

For the proposed approaches, ten actions were sampled at each timestep, and an action buffer size of 50 was used. Once the buffer becomes full, actions are removed in a FIFO manner. For the static-τ approach, a value of $\tau = 0.7$ was used for experimentation, and this value is also used as the initial value for the adaptive-τ approach. All other parameters were kept the same for each of the compared approaches. Each actor samples 10 actions at each timestep, and does its own action selection using its own sampled actions and buffer.

Since multiple actions are selected at each step, additional computational complexity is introduced. To help reduce the additional time requirements of evaluating multiple actions, the code was implemented to use a GPU while training. More specifically, determining the distance between the sampled actions and those in the buffer and evaluating the heuristics of the sampled actions were performed in parallel. More details on hyper-parameter settings can be found in Appendix D.

5 Experimental Setup

Six two-objective environments from the MO-Gymnasium benchmark set (Felten et al., 2023) were used for experimentation. A summary of each environment is given in Table 1. For the MO-Hopper and the MO-Ant environments, the control cost is included as an additive reward for each of the objectives. Figure 1 gives example states from each environment. For more details on the environments, refer to Appendix C.

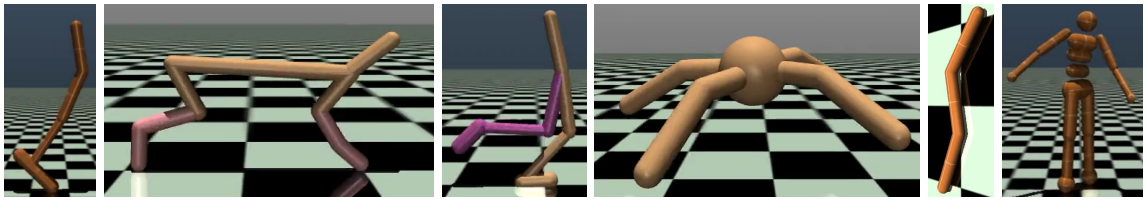


Figure 1: Example states from each environment (from left to right): MO-Hopper, MO-HalfCheetah, MO-Walker2D, MO-Ant, MO-Swimmer, and MO-Humanoid. Note that the MO-Swimmer image has been rotated counterclockwise by 90 degrees.

The hypervolume is used to evaluate both the convergence and diversity of a found POF, and is the volume of the region dominated by points in the found POF and a reference point from the objective space (Zitzler, 1999). The expected maximum utility (EUM) (Zintgraf et al., 2015) measures the weighted sum of objective function values for each solution in the found POF. The final performance measure used is the cardinality of the found POF ($|POF|$). For each of these performance measures, a higher value is better.

The sparsity (Xu et al., 2020) measures the distance between points in the found POF. The sparsity is not used as a performance measure in the main paper since it has some problems (for example a POF with only one solution has the best sparsity value, even though we are looking for a set of multiple solutions), and so it

Table 2: Summary of rankings of the final performance measure values across all environments: the percentage of times each algorithm was the best in each metric is given next to the total first ranking, top-2 count, total last ranking, and bottom-2 count in parentheses. All performance measures are used to evaluate the final POFs. A ↓ beside a measure indicates that a lower value is better, and a ↑ beside a measure indicates that a higher value is better. The background of each cell is also shaded according to its ranking such that the best-ranked approach in each row has a dark blue colour and the worst-ranked approach has a dark red colour, and the best value is also highlighed in bold test.

Metric	Original MOSAC	Static-τ	LI-τ	LD-τ	Adaptive-τ
Average Rank(↓)	2.88	3.78	2.44	**2.22**	3.17
Median Rank(↓)	3.00	4.00	**2.00**	**2.00**	3.00
Total First(↑)	**6.00(33.33%)**	1.00(5.56%)	4.00(22.22%)	5.00(27.78%)	3.00(16.67%)
Top-2 Count(↑)	9.00(50.00%)	2.00(11.11%)	10.00(55.56%)	**11.00(61.11%)**	7.00(38.89%)
Total Last	4.00(22.22%)	5.00(27.78%)	2.00(11.11%)	**1.00(5.56%)**	6.00(33.33%)
Bottom-2 Count(↓)	10.00(55.56%)	15.00(83.33%)	3.00(16.67%)	**1.00(5.56%)**	8.00(44.44%)

can be misleading on its own. The results for the sparsity are included in the supplementary material for reference.

Due to the high computational demand of the DRL methods used, a total of five runs were conducted each approach in each environment. For a fair comparison, an approximation of the IQM is used which takes the average performance measure values of the middle three runs, in order to avoid the data becoming skewed as a result of any outliers.

6 Experimental Results

To compare the proposed approaches to the original MOSAC, the performance of each approach with respect to final performance metric values, the performance metric values over time, and POFs are evaluated. Additional results and runtime analysis are given in Appendix F.

6.1 Overall Comparison

Table 2 gives a summary of the rankings of the final performance metric values across all environments. For each environment, the performance of each approach was ranked according to each metric. The average and median rankings across all performance measures in all environments, and the number of times each approach ranked first, last, in the top 2, and in the bottom 2 are provided.

The LD-τ approach had the best performance overall, and had the best average and median ranks. LD-τ was also ranked first or second in 61.11% of cases, and was only ranked in the bottom two in 5.56% of cases, which is greatly improved over the original MOSAC approach which was ranked in the bottom two in 55.56% of cases. The LI-τ approach also had good performance overall, and was ranked second in most measures. In contrast, the static-τ and adaptive-τ approaches generally underperformed compared to the original MOSAC, and static-τ specifically was ranked in the bottom two in 83.33% of cases.

The rankings for each environment individually are given in Table 7. The results show that the performance of each approach is dependent on the complexity of the environment as well as on which exploration approach is used. In simpler environments such as the MO-Swimmer, and in MO-HalfCheetah and MO-Walker2D, the original MOSAC approach performs well. This shows that using only the learned actor distribution in MOSAC is sufficient in simpler tasks, but is inadequate in more complex environments with large action and observation spaces.

Both the LI-τ and LD-τ approaches had much more consistently competitive results compared to the original MOSAC. For the LI-τ approach, the low τ values help to avoid excessive exploration before a base policy is learned, allowing it to achieve good results in simpler environments. The LD-τ approach had better performance in the MO-Ant and MO-Humanoid environments compared to LI-τ, showing that

more exploration earlier in training is beneficial in complex environments with large action and observation spaces. LD-τ also had good performance in the MO-Hopper environment, where it was ranked first for every performance measure. Additionally, the LD-τ approach is never in the bottom-two more than 33.33% of the time, and this only occurs in a single environment (MO-HalfCheetah), and for all other environments it is never ranked last. In contrast, the original MOSAC has strong performance in some environments, its performance degrades significantly MO-Hopper as well as in the more complex MO-Ant and MO-Humanoid environments, and is in the bottom two 100% of the time for MO-Hopper and MO-Ant. Therefore, the LD-τ approach is more robust to different environment types compared to the original MOSAC.

The adaptive-τ approach was ranked poorly in the simpler environments, but had much better performance in the more difficult MO-Ant and MO-Humanoid environments. In cases where the action space is large, it is more difficult to determine the exact degree of exploration which is needed, and so being able to adjust τ depending on the performance at a given time is beneficial. The static-τ approach underperforms as well, and ranks first only a single time across all environments, showing that a fixed value of τ is not the ideal approach to be able to learn in different tasks.

To illustrate the variance of the results, Figure 2 gives boxplots of the final performance metric values for all environments. The LI-τ and LD-τ approaches had good performance in most metrics, and were often first or second for the hypervolume and EUM. These approaches often had a relatively low variance and were also rarely ranked last, compared to the original MOSAC which has the lowest median and a much higher variance than the LI-τ and LD-τ approaches, especially in the hypervolume and EUM for MO-Ant and MO-Hopper environments. The static-τ often had a large degree of variance in its results for the hypervolume and the EUM, showing less consistency in its results compared to the LD-τ and LI-τ approaches.

The adaptive-τ approach often has a low median, however in the MO-Ant and MO-Humanoid environments it has the best median performance in hypervolume and EUM. This shows that although the performance of the adaptive-τ approach is environment-dependent, in environments with large action spaces it consistently does well. Similarly to the results discussed above, the adaptive approach allows for more or less exploration as needed, and additional adaptive strategies, such as environment-aware strategies, will be explored in future work.

6.2 Performance Over Time

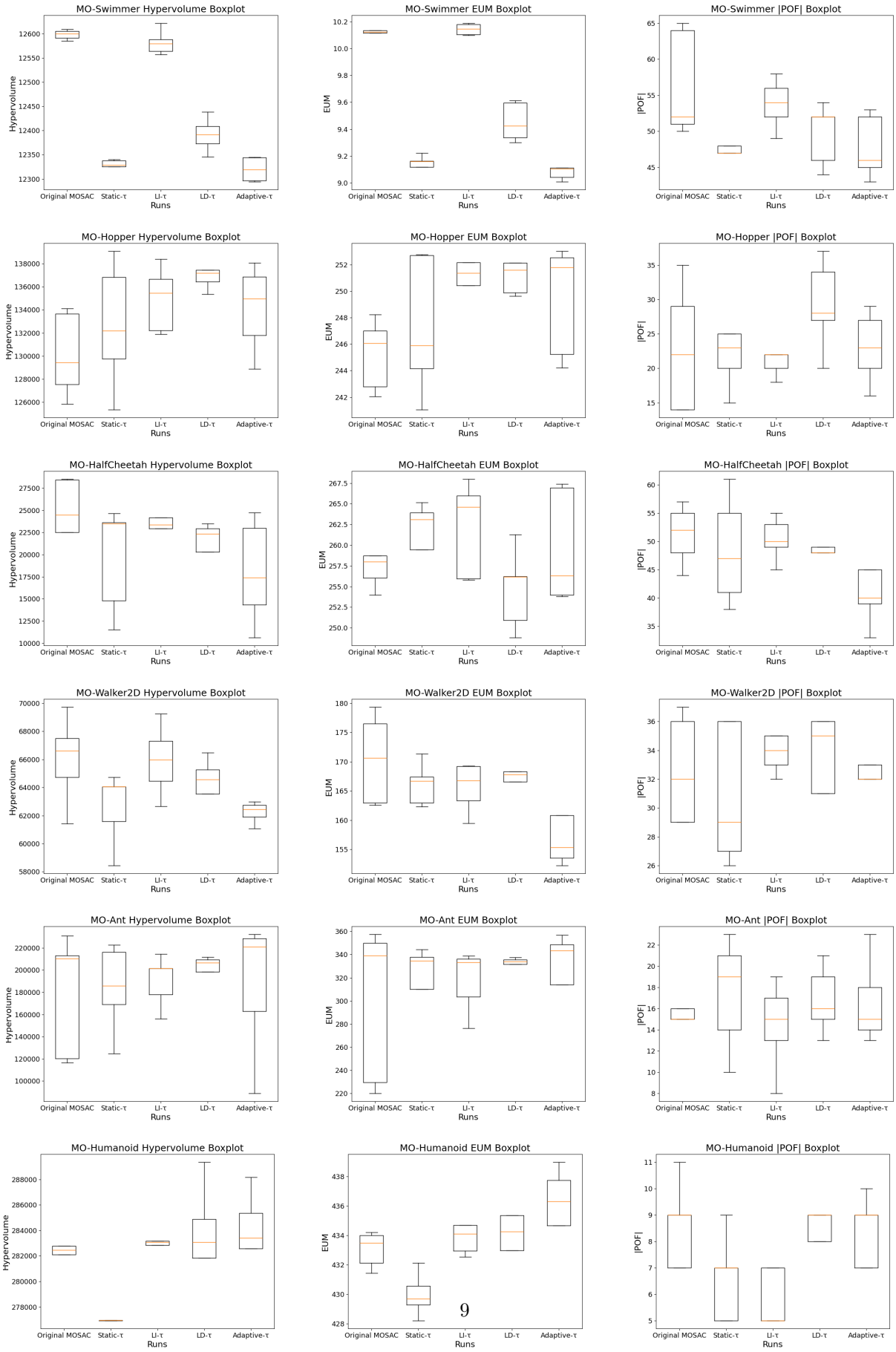
Graphs of the IQM of the performance metric values during training for the MO-Hopper, MO-Ant, and MO-Humanoid are provided in Figure 3. In the MO-Hopper environment, the LD-τ approach outperformed the other approaches from close to the beginning of training until the end for all three performance measures. In the MO-Walker envrionment, the LI-τ approach had higher performance metric values during most of training. Although the original MOSAC had a similar final value, the LI-τ approach was able to find a better POF earlier on, and continue to improve as training continued. In the MO-Ant, all of the introduced approaches outperformed the original MOSAC throughout most of training. The LD-τ and adaptive-τ approaches had the best overall performance, in all metric except for the $|POF|$, where static-τ was best. In the MO-Humanoid environment, all approaches were fairly similar, although up to about 0.75e6 timesteps the static-τ, LI-τ, and LD-τ approaches all had higher hypervolume and EUM values compared to the original MOSAC. For MO-Swimmer, the LI-τ and original MOSAC are the best performing overall, but similarly to other environments, the LI-τ approach converges faster than the original MOSAC, although they end with similar final values.

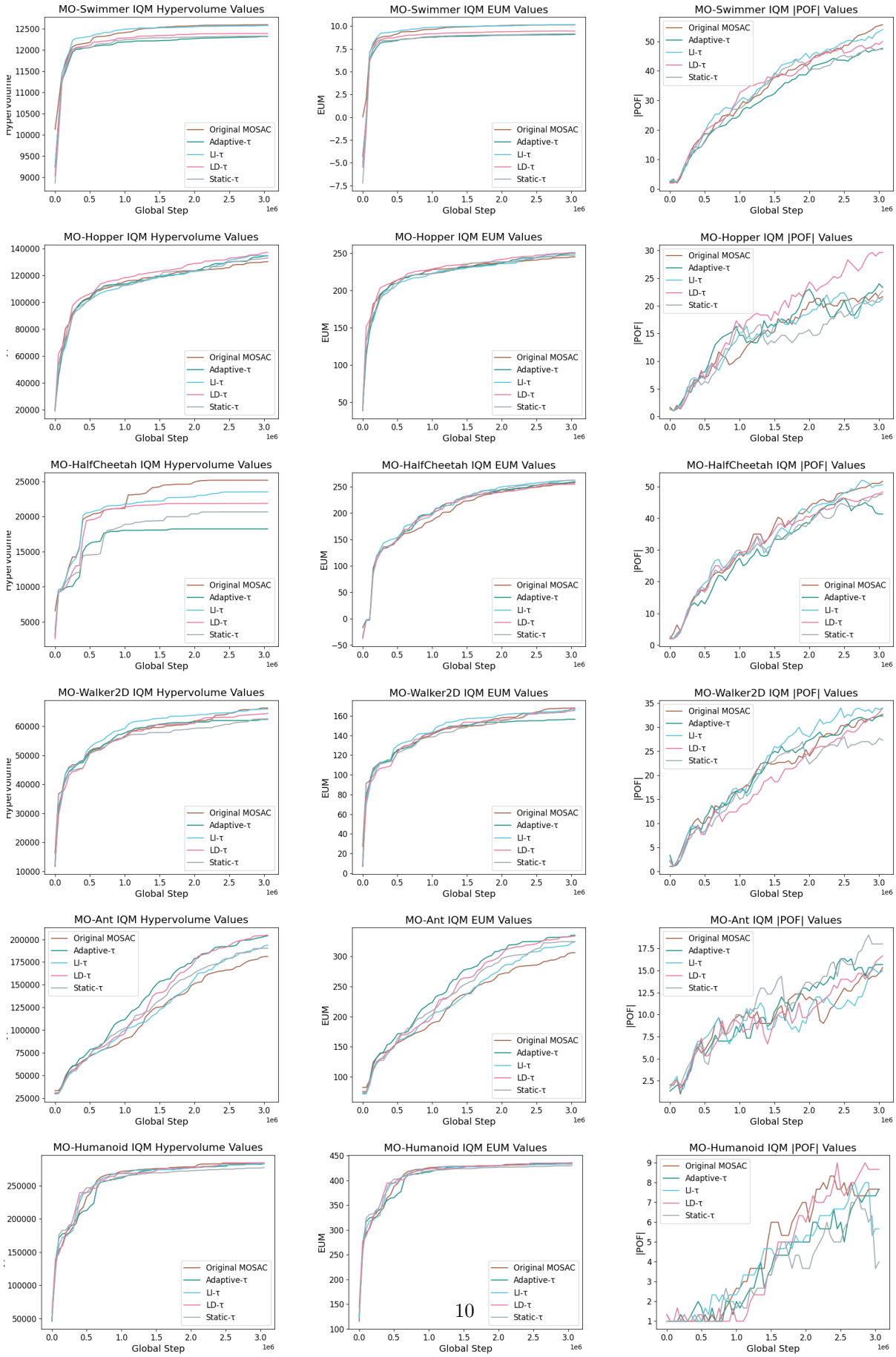
In the MO-HalfCheetah environment, the original MOSAC as the best performance from about 1.0e6 steps onward. Although in this particular environment the original MOSAC is the top performer, this is not the case in any of the other environments, suggesting that the performance of the original MOSAC is more problem dependant compared to the proposed LI-τ and LD-τ approaches.

Overall, the proposed LI-τ and LD-τ approaches had improved performance during initial learning for the majority of environments. These approaches are able to find better solutions faster, even when the final results (mainly with respect to the hypervolume and EUM) often converge to be more similar to the baseline. Therefore, the proposed LI-τ and LD-τ approaches would be particularly valuable in cases where training time is costly.

Table 3: The average and median rankings, the total number of times each algorithm ranked first, in the top-2, last, and in the bottom-2 according to the IQM. The percentage of times each algorithm ranked first, last, in the top-2, and in the bottom-2 is given next to each of these counts in parentheses. A ↓ beside a metric indicates that a lower value is better, and a ↑ beside an metric indicates that a higher value is better. The background of each cell is also shaded according to its ranking such that the best-ranked approach in each row has a dark blue colour and the worst-ranked approach has a dark red colour.

Metric	Original MOSAC	Static τ	LI-τ	LD-τ	Adaptive-τ
MO-Swimmer					
Average Rank(↓)	**1.33**	4.33	1.67	3.00	4.67
Median Rank(↓)	**1.00**	4.00	2.00	3.00	5.00
Total First(↑)	**2.00(66.67%)**	0.00(0.00%)	1.00(33.33%)	0.00(0.00%)	0.00(0.00%)
Top-2 Count(↑)	**3.00(100.00%)**	0.00(0.00%)	**3.00(100.00%)**	0.00(0.00%)	0.00(0.00%)
Total Last(↓)	**0.00(0.00%)**	1.00(33.33%)	**0.00(0.00%)**	**0.00(0.00%)**	2.00(66.67%)
Bottom-2 Count(↓)	**0.00(0.00%)**	3.00(100.00%)	**0.00(0.00%)**	**0.00(0.00%)**	3.00(100.00%)
MO-Hopper					
Average Rank(↓)	4.67	4.33	2.00	**1.33**	2.67
Median Rank(↓)	5.00	4.00	2.00	**1.00**	3.00
Total First(↑)	0.00(0.00%)	0.00(0.00%)	1.00(33.33%)	**2.00(66.67%)**	0.00(0.00%)
Top-2 Count(↑)	0.00(0.00%)	0.00(0.00%)	2.00(66.67%)	**3.00(100.00%)**	1.00(33.33%)
Total Last(↓)	2.00(66.67%)	1.00(0.00%)	**0.00(0.00%)**	**0.00(0.00%)**	**0.00(0.00%)**
Bottom-2 Count(↓)	3.00(100.00%)	3.00(100.00%)	**0.00(0.00%)**	**0.00(0.00%)**	**0.00(0.00%)**
MO-HalfCheetah					
Average Rank(↓)	2.00	3.33	**1.67**	3.67	4.33
Median Rank(↓)	**1.00**	4.00	2.00	3.00	5.00
Total First(↑)	**2.00(66.67%)**	0.00(0.00%)	1.00(33.33%)	0.00(0.00%)	0.00(0.00%)
Top-2 Count(↑)	2.00(66.67%)	1.00(33.33%)	**3.00(100.00%)**	0.00(0.00%)	0.00(0.00%)
Total Last(↓)	**0.00(0.00%)**	**0.00(0.00%)**	**0.00(0.00%)**	1.00(33.33%)	2.00(66.67%)
Bottom-2 Count(↓)	1.00(33.33%)	2.00(66.67%)	**0.00(0.00%)**	1.00(33.33%)	2.00(66.67%)
MO-Walker2D					
Average Rank(↓)	**1.33**	3.67	2.00	2.00	4.00
Median Rank(↓)	**1.00**	4.00	2.00	2.00	5.00
Total First(↑)	**2.00(66.67%)**	0.00(0.00%)	1.00(33.33%)	1.00(33.33%)	0.00(0.00%)
Top-2 Count(↑)	**3.00(100.00%)**	0.00(0.00%)	2.00(66.67%)	2.00(66.67%)	1.00(33.33%)
Total Last(↓)	**0.00(0.00%)**	1.00(0.00%)	**0.00(0.00%)**	**0.00(0.00%)**	2.00(66.67%)
Bottom-2 Count(↓)	1.00(33.33%)	3.00(100.00%)	**0.00(0.00%)**	**0.00(0.00%)**	3.00(100.00%)
MO-Ant					
Average Rank(↓)	4.67	2.67	4.00	**1.67**	2.00
Median Rank(↓)	5.00	3.00	4.00	**2.00**	**2.00**
Total First(↑)	0.00(0.00%)	**1.00(33.33%)**	0.00(0.00%)	**1.00(33.33%)**	**1.00(33.33%)**
Top-2 Count(↑)	0.00(0.00%)	1.00(33.33%)	0.00(0.00%)	**3.00(100.00%)**	2.00(66.67%)
Total Last(↓)	2.00(66.67%)	**0.00(0.00%)**	1.00(33.33%)	**0.00(0.00%)**	**0.00(0.00%)**
Bottom-2 Count(↓)	3.00(100.00%)	1.00(33.33%)	2.00(66.67%)	**0.00(0.00%)**	**0.00(0.00%)**
MO-Humanoid					
Average Rank(↓)	3.33	4.33	3.33	1.67	**1.33**
Median Rank(↓)	4.00	5.00	3.00	2.00	**1.00**
Total First(↑)	0.00(0.00%)	0.00(0.00%)	0.00(0.00%)	1.00(33.33%)	**2.00(66.67%)**
Top-2 Count(↑)	1.00(33.33%)	0.00(0.00%)	0.00(0.00%)	**3.00(100.00%)**	**3.00(100.00%)**
Total Last(↓)	**0.00(0.00%)**	2.00(66.67%)	1.00(33.33%)	**0.00(0.00%)**	**0.00(0.00%)**
Bottom-2 Count(↓)	2.00(66.67%)	3.00(100.00%)	1.00(33.33%)	**0.00(0.00%)**	**0.00(0.00%)**


MO-Swimmer Hypervolume Boxplot
MO-Swimmer EUM Boxplot
MO-Swimmer |POF| Boxplot
MO-Hopper Hypervolume Boxplot
MO-Hopper EUM Boxplot
MO-Hopper |POF| Boxplot
MO-HalfCheetah Hypervolume Boxplot
MO-HalfCheetah EUM Boxplot
MO-HalfCheetah |POF| Boxplot
MO-Walker2D Hypervolume Boxplot
MO-Walker2D EUM Boxplot
MO-Walker2D |POF| Boxplot
MO-Ant Hypervolume Boxplot
MO-Ant EUM Boxplot
MO-Ant |POF| Boxplot
MO-Humanoid Hypervolume Boxplot
MO-Humanoid EUM Boxplot
MO-Humanoid |POF| Boxplot
Hypervolume
EUM
|POF|
Original MOSAC
Static-τ
LI-τ
LD-τ
Adaptive-τ
Runs


MO-Swimmer IQM Hypervolume Values
MO-Swimmer IQM EUM Values
MO-Swimmer IQM |POF| Values
MO-Hopper IQM Hypervolume Values
MO-Hopper IQM EUM Values
MO-Hopper IQM |POF| Values
MO-HalfCheetah IQM Hypervolume Values
MO-HalfCheetah IQM EUM Values
MO-HalfCheetah IQM |POF| Values
MO-Walker2D IQM Hypervolume Values
MO-Walker2D IQM EUM Values
MO-Walker2D IQM |POF| Values
MO-Ant IQM Hypervolume Values
MO-Ant IQM EUM Values
MO-Ant IQM |POF| Values
MO-Humanoid IQM Hypervolume Values
MO-Humanoid IQM EUM Values
MO-Humanoid IQM |POF| Values
Hypervolume
EUM
|POF|
Global Step
1e6
Original MOSAC
Adaptive-τ
LI-τ
LD-τ
Static-τ

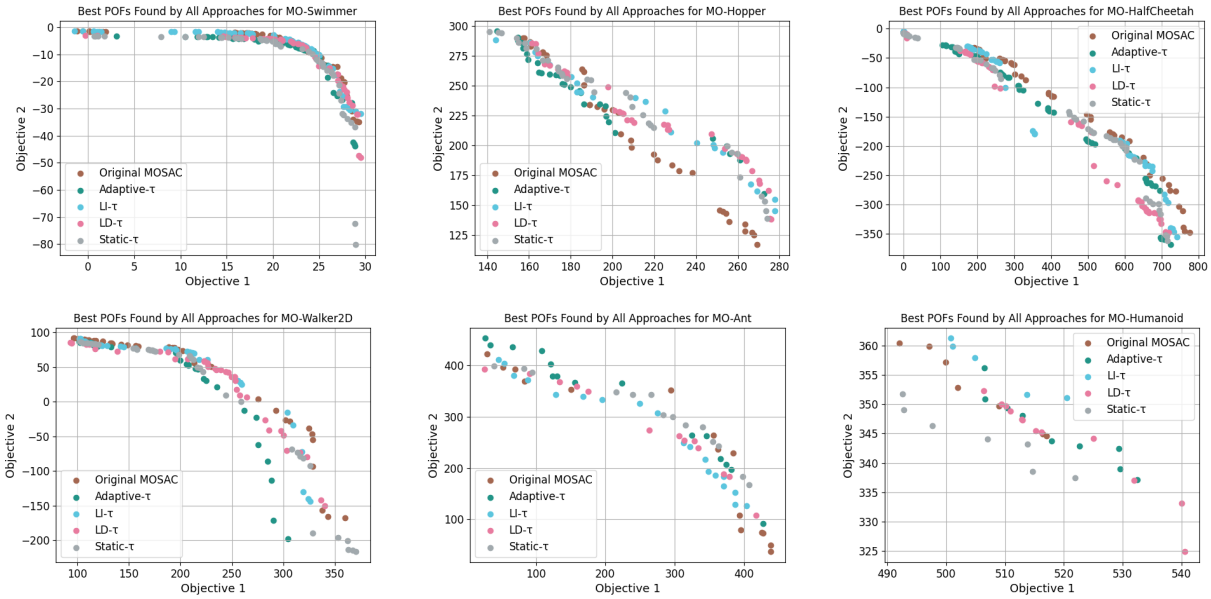


Figure 4: All best found POFs based on the final hypervolume value.

6.3 Comparison of Best Found Pareto-Optimal Fronts

Figure 4 gives the best (with respect to the final hypervolume) POFs for each environment in order to illustrate the results in the best-case performance we were able to acheive in our experiments.

For the MO-Swimmer environment, all algorithms have converged to a similar POF. However, the LD-τ and LI-τ contain many solutions which dominate those found by the rest of the compared approaches. For the MO-Hopper environment, the static-τ, LI-τ, and LD-τ approaches are able to find many solutions which dominate those found by the original MOSAC, especially in the right half of the graph. The adaptive-τ is only able to find solutions in a limited area of the POF, and misses many non-dominated solutions that the other approaches are able to find. For the MO-HalfCheetah environment, the original MOSAC is able to find solutions which dominate many of those found by the other approaches, however it misses solutions in the top left area of the graph that other approaches are able to find. For the MO-Walker2D environment, all of the approaches had more difficulty finding Pareto-optimal solutions in the right half of the graph compared to the left. For the MO-Ant problem, the original MOSAC has several points which have the best objective function values, however there are large gaps within its best found POF. The POF found by LD-τ has much smaller gaps, and although the adaptive-τ also has some large gaps in its POF, it is able to find more solutions which dominate those found by the original MOSAC, especially in the top left area of the graph. LI-τ and LD-τ were able find Pareto-optimal solutions which also dominate many of those found by the remaining approaches for the MO-Humanoid environment.

These results show that the proposed LI-τ and LD-τ approaches are able to find high-quality POFs which contain better (with respect to Pareto-dominance) solutions compared to the original MOSAC in the majority of the compared environments. This is especially true in the case of MO-Ant, MO-Swimmer, MO-Hopper, and MO-Humanoid, which have action spaces ranging from 2 dimensions up to 17 dimensions, showing that the introduced approaches scale well to different sizes of action space.

7 Conclusion

This paper introduced four novel action selection strategies used within the MORL/D framework for multi-objective control tasks. The results showed that by promoting exploration at different stages of the learning

process, agents are better able to explore the action space and find better quality final POFs. The best two approaches overall were found to be LI-τ and LD-τ, which promotes more exploration at the end and beginning of training, respectively. In the case of larger action spaces, the adaptive-τ approach had the best performance.

7.1 Limitations and Future Work

Although the proposed LI-τ and LD-τ approaches had good performance overall, there are some limitations. Firstly, these approaches were tested only in environments with two objectives to ensure that any differences in performance in different environments were due to changes in the size of the action space, rather than in the objective space. Additionally, only one adaptive approach was tested, and there are many other options which could be tested in a future comparative study such as using the best reward found in the most recent few episodes, or the average reward across all members of the population. Since the performance of the adaptive-τ approach had good performance only in environments with large action spaces, an environment-aware approach could also be used.

The Euclidean distance was used as the distance metric in this work since in each problem, all actions had the same range. In some applications there are different ranges for different actions, and in these cases the Euclidean distance becomes biased. Therefore, a comparison of different distance metrics will be done in future work. Future work could also include considering the state space alongside the action space in the heuristic function, and using the introduced LD-τ or LI-τ to train a model for a real-world robot.

References

Alegre, L. N., Bazzan, A. L. C., Roijers, D. M., Nowé, A., and da Silva, B. C. (2023). Sample-efficient multi-objective learning via generalized policy improvement prioritization. In *Proceedings of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023)*.

Amirabadi, R. K. and Fard, O. S. (2024). Combining hybrid metaheuristic algorithms and reinforcement learning to improve the optimal control of nonlinear continuous-time systems with input constraints. *Computers and Electrical Engineering*, 116:109179.

Castelletti, A., Pianosi, F., and Restelli, M. (2013). A multiobjective reinforcement learning approach to water resources systems operation: Pareto frontier approximation in a single run. *Water Resources Research*, 49(6):3476–3486.

Chen, L., Zhu, G., Liang, W., and Wang, Y. (2023). Multi-objective reinforcement learning approach for trip recommendation. *Expert Systems with Applications*, 226:120145.

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

Felten, F., Alegre, L. N., Nowé, A., Bazzan, A. L. C., Talbi, E. G., Danoy, G., and Silva, B. C. da. (2023). A toolkit for reliable benchmarking and research in multi-objective reinforcement learning. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS 2023)*.

Felten, F., Danoy, G., Talbi, E.-G., and Bouvry, P. (2022). Metaheuristics-based exploration strategies for multi-objective reinforcement learning. In *Proceedings of the 14th International Conference on Agents and Artificial Intelligence (ICAART 2022)*, volume 2, pages 662–673.

Felten, F., Talbi, E.-G., and Danoy, G. (2024). Multi-objective reinforcement learning based on decomposition: A taxonomy and framework. *Journal of Artificial Intelligence Research*, 79:679–723.

Gambardella, L. M. and Dorigo, M. (1995). Ant-q: A reinforcement learning approach to the traveling salesman problem. In Prieditis, A. and Russell, S., editors, *Machine Learning Proceedings 1995*, pages 252–260. Morgan Kaufmann, San Francisco (CA).

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., and Levine, S. (2019). Soft actor-critic algorithms and applications.

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

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

Huynh, T. N. Y., Nguyen, A. T., Ahn, Y., Oo, B. L., and Lim, B. T. (2025). Multi objectives reinforcement learning for smart buildings: A systematic review of algorithms, applications and future perspectives. *Energy and Buildings*, 345:116045.

Keller, A. A. (2019). *Multi-Objective Optimization in Theory and Practice II: Metaheuristic Algorithms*. Bentham Science Publishers Ltd., Publisher Address.

Kharitonov, A., Abani, J. I., Nahhas, A., and Turowski, K. (2025). Literature survey on combining machine learning and metaheuristics for decision-making. *Procedia Computer Science*, 253:199–208. 6th International Conference on Industry 4.0 and Smart Manufacturing.

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

Lu, H., Herman, D., and Yu, Y. (2023). Multi-objective reinforcement learning: Convexity, stationarity and pareto optimality. In *The Eleventh International Conference on Learning Representations (ICLR 2023)*.

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

Reymond, M., Bargiacchi, E., and Nowé, A. (2022). Pareto conditioned networks. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022)*, pages 1110–1118.

Roijers, D., Steckelmacher, D., and Nowe, A. (2018). Multi-objective reinforcement learning for the expected utility of the return. In *Adaptive Learning Agents (ALA) workshop at AAMAS/ICML/IJCAI 2018*.

Saiki, T. and Arai, S. (2023). Flexible traffic signal control via multi-objective reinforcement learning. *IEEE Access*, 11:75875–75883.

Valencia, D., Jia, J., Li, R., Hayashi, A., Lecchi, M., Terezakis, R., Gee, T., Liarokapis, M., MacDonald, B. A., and Williams, H. (2023). Comparison of model-based and model-free reinforcement learning for real-world dexterous robotic manipulation tasks. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 871–878.

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

Venske, S. M. S., de Almeida, C. P., and Delgado, M. R. (2024). Metaheuristics and machine learning: an approach with reinforcement learning assisting neural architecture search. *Journal of Heuristics*, 30:199–224.

Xu, J., Tian, Y., Ma, P., Rus, D., Sueda, S., and Matusik, W. (2020). Prediction-guided multi-objective reinforcement learning for continuous robot control. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, pages 10607–10616.

Yang, R., Sun, X., and Narasimhan, K. (2019). A generalized algorithm for multi-objective reinforcment learning and policy adaptation. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, pages 14636–14647.

Zintgraf, L. M., Kanters, T. V., Roijers, D. M., Oliehoek, F. A., and Beau, P. (2015). Quality assessment of MORL algorithms: A utility-based approach. In *Benelearn 2015: Proceedings of the 24th Annual Machine Learning Conference of Belgium and the Netherlands*.

Zitzler, E. (1999). *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology Zurich (ETH).

A Appendices Overview and Additional Resources

These appendices provide additional details and experimental results. The appendices are organized as follows:

- Appendix B gives a more detailed explanation of the algorithms which were used for experimentation,
- Appendix C gives details on the environments which were used,
- Appendix D gives more details on the experimental setup including performance metrics and hyper-parameter settings,
- Appendix E gives details on the parameter selection studies which were conducted on the introduced approaches, and
- Appendix F gives additional experimental results including runtime analysis.

Our code is available at Anonymous Github Repository Link with instructions for its use in future work.

Video demos of some of the various behaviours of the policies depending on the weights of the objectives are available here: Video Link.

B Additional Algorithm Details

This section gives additional information on the algorithms used. Since the focus of this work is mainly on the action selection strategy of actors for multi-objective problems with continuous state and action spaces rather than other implementation details, we only give a brief overview here. For a more detailed explanation of multi-objective reinforcement learning based on decomposition (MORL/D), soft actor-critic (SAC) and their implementations including loss functions, please refer to Felten et al. (2024) and Haarnoja et al. (2019).

B.1 Multi-Objective Reinforcement Learning Based on Decomposition

In MORL/D, the multi-objective problem is split into multiple single-objective problems which each use the scalarized reward. MORL/D uses linear scalarization, or the weighted sum of each of the objective functions. Given a set of weights $\lambda \in \mathbb{R}^m$ and objective function values $\mathbf{f}$ corresponding to a policy x, the scalarized reward is given by:

$$g(x) = \sum_{i=1}^{m} \lambda_i f_i(x), \tag{4}$$

where m is the number of objectives in the problem.

MORL/D maintains a population of individuals, each of which corresponds to one of the weight vectors. The weight vectors are uniformly generated using the Riesz s-Energy method on a unit simplex as is done by Felten et al. (2024).

In order to find solutions corresponding to different areas of the Pareto-optimal front (POF), the weights are adapted over time using Pareto-simulated annealing (PSA) (Czyż̇ak and Jaszkiewicz, 1998). PSA adjusts the weights of a policy x depending on its current performance as well as its proximity to its nearest non-dominated neighbour x' with:

$$\lambda_i^x = \begin{cases} \delta\lambda_i^x & \text{if } f_i(x) \geq f_i(x') \\ \lambda_i^x/\delta & \text{if } f_i(x) < f_i(x') \end{cases}, \tag{5}$$

where $\delta = 1.1$. Weights are adapted every $50,000$ timesteps. To improve cooperation between solutions, a shared experience buffer is used by all individuals in the population. Candidates are selected to sample experiences using a roulette-wheel strategy in order to uniformly train the policies.

The archive of non-dominated solutions only uses the Pareto-dominance relation to maintain it. The critics each output m values, i.e. a multi-objective regression is used. Samples are taken by following each of the candidate policies, and policy updates are performed on the currently followed policy.

Algorithm 1, gives the high-level framework of MORL/D. In Algorithm 1, the population of policies is denoted by Π. To start, the policies, weights, archive, and replay buffer are initialized. During each iteration, samples are taken from the environment using a specific policy and are added to the buffer. Then u batches are sampled from the shared episode replay buffer for each individual in the population. The policies are then evaluated, and any Pareto-optimal ones are added to the archive. Weights are adapted using PSA if $50,000$ timesteps have passed. Once the stopping criterion is met, the final set of non-dominated policies is returned.

Algorithm 1 MORL/D Framework, adapted from Felten et al. (2024)

Input: Stopping criterion *stop*, population size n, scalarization method g, environment *env*, update passes u
Output: Approximate POS stored in the archive population $\mathbb{EP}$
$\Pi, \mathbb{W} = \text{Initialize}(n)$
$evals = \text{EvaluatePolicies}(\Pi)$
$\mathbb{EP} = \text{Prune}(\Pi, evals)$
$\mathbb{B} = \emptyset$
while not *stop* **do**
  $\pi, \vec{\lambda} = \text{Select}(\Pi, \mathbb{W})$
  $experiences = \text{Sample}(env, \pi, g, \vec{\lambda})$
  $\mathbb{B} = \text{UpdateBuffer}(\mathbb{B}, experiences)$
  $\Pi = \text{ImprovePolicies}(\Pi, g, \mathbb{W}, \mathbb{B}, u)$
  $evals = \text{EvaluatePolicies}(\Pi)$
  $\mathbb{EP} = \text{Prune}(\mathbb{EP} \cup \Pi, evals)$
  if $50,000$ timesteps since weights last adapted **then**
    $\mathbb{W} = \text{AdaptWeightsPSA}(\mathbb{W}, \mathbb{EP})$
  end if
end while
return $\mathbb{EP}$

B.2 Soft Actor-Critic

Soft actor-critic (SAC) (Haarnoja et al., 2019) is an off-policy maximum-entropy actor-critic RL method. In MORL/D, a policy is found for each weight vector using SAC. The actor in SAC aims to maximize both the expected return and the entropy of the policy in order to encourage both stability and exploration.

The objective of SAC is to learn the ideal policy π^* that maximizes the reward(which in the case of MORL/D is the scalarized reward) along with the entropy. A temperature parameter is used to determine the weight of the entropy, and is adapted automatically using the approach in Haarnoja et al. (2019). Additionally, the replay pool is shared between all of the actors in the population of MORL/D.

Algorithm 2 gives the pseudocode for SAC.

SAC uses two soft Q-functions which are represented using neural networks (NNs), and the minimum of the two is used for updating the policy. It was found by Haarnoja et al. (2019) that using two soft Q-functions rather than one significantly speeds up the training process.

C Environment Details

This section gives further details on each of the environments used for comparison, including the objective functions. All of these environments are implemented in the MO-Gymnasium library (Felten et al., 2023). The purpose of this section is to give an overview of the objectives for each environment, and all additional

Algorithm 2 Soft Actor-Critic (adapted from Haarnoja et al. (2019))

```
Initialize the target networks weights
for each iteration do
  for each environment step do
    Sample an action from the policy
    Get the next state from the environment
    Add the transition to the shared replay pool
  end for
  for each gradient step do
    Update Q-function parameters
    Update policy weights
    Update target network weights
    Update the value of τ if using the LD-τ, LI-τ, or Adaptive-τ approaches
    Adjust temperature using automatic tuning
  end for
end for
return The parameterized Q-functions and policy
```

parameters are kept as the same as those suggested by Felten et al. (2023). For the MO-Humanoid environment, all actions can be in the range $[-0.4, 0.4]$. For all other environments, actions can be in the range $[-1.0, 1.0]$.

C.1 MO-Hopper

The MO-Hopper is a 2D figure with one leg. The MO-Hopper has four body parts: a torso, a thigh, a leg, and a foot. The goal is for the Hopper to move by applying torque to the joints connecting each of the body parts together. The 3D action space corresponds to the torque applied at each of the three joints. The 11D observation space consists of the positions and velocities of each of the robot's body parts, including both the body parts and the joints.

The cost of taking an action is given by:

$$ctrl_cost(a) = -w_{ctrl} \times ||a||_2^2, \tag{6}$$

where w_{ctrl} is used to control the weight of the control cost, and has a default value of 10^{-3}.

For the MO-Hopper, the control cost is added to each of the objective functions along with a reward for staying healthy. The healthy reward($healthy_reward$) gives an additional reward of $+1$ for each timestep that the hopper remains healthy.

The first objective of the hopper environment (the velocity in the x-direction) is given by:

$$x_vel(a) = (x_pos_after - x_pos_before)/dt + ctrl_cost(a) + healthy_reward, \tag{7}$$

where x_pos_after is the x-position of the hopper after the action has been taken, x_pos_before is the x-position of the hopper before the action was taken, and the time between actions is $dt = 0.008$ by default.

The second objective(the height of the hopper) is given by:

$$height = 10z + ctrl_cost(a) + healthy_reward, \tag{8}$$

where z is the z-coordinate of the robot's torso.

A training episode is terminated if the Hopper becomes "unhealthy", which happens when any one of the following becomes true:

1. Any of the observed values, with the exception of the $x-$ and $z-$coordinates, are outside of the healthy range in the interval $[-100, 100]$, or

2. The height of the hopper is outside of the healthy range, where the healthy range is $[0.7, +\infty)$, or
3. The angle of the hopper's torso is outside of the healthy range, where the healthy range is $[-0.2, 0.2]$.

Otherwise, the episode is truncated after 1000 timesteps.

C.2 MO-HalfCheetah

The MO-HalfCheetah consists of a 2D robot made up of two thighs, two shins, two feet, a torso, and a head. The torso and the head remain fixed, and so torque is only applied to the joints connecting the feet to the shins, the shins to the thighs, and the thighs to the torso. The objectives are to run forward in the x-direction as fast as possible, and to minimize the control cost. The six-dimensional action space represents the torques which are applied to each of the joints. The 17D observation space consists of the positions and velocities of each of the robot's body parts. parameter The first objective gives a reward for running forward, given by:

$$forward_reward = (x_{after_action} - x_{before_action})/dt, \tag{9}$$

where $x_{after_action} - x_{before_action}$ gives the displacement of the "tip" of the robot, and dt represents the time between actions, given by $dt = 0.05$ by default.

The second objective is the control cost of the action, which is given by Eq. 6 and has a weight of $w_{ctrl} = -0.1$.

An episode does not terminate, however an episode is truncated after 1000 timesteps.

C.3 MO-Walker2D

The MO-Walker2D robot adds another set of legs to the MO-Hopper environment which allows it to walk. The walker is made up of a torso, two thighs, two legs, and two feet. The objectives are to move forward in the x-direction while minimizing the control cost of each action by applying force to each of the hinge joints that connect the body parts together. The 6D action space represents the torques which are applied at each joint. The 17D observation space contains the positions and velocities of each of the robot's body parts.

The first objective gives the reward for running forward using Eq. 9, with a value of $dt = 0.008$. The second objective gives the control cost of the action, given by Eq. 6 with a value of $w_{ctrl} = 10^{-3}$.

A healthy reward bonus is added to each objective, with a value of $+1$ for each timestep that the walker remains healthy. The walker is unhealthy if any of the following occur:

1. a state space value is no longer finite, or
2. the z-coordinate of the torso is outside of the healthy range of $[0.8, 2.0]$, or
3. the absolute value of the angle of the torso is outside of the healthy range of $[-1, 1]$.

An episode is terminated if the robot becomes unhealthy at any point, and otherwise it is truncated after 1000 timesteps.

C.4 MO-Ant-v5

The MO-Ant is a 3D quadruped which has a torso and four legs. Each leg has two body parts. The goal of the MO-Ant environment is to walk by applying torque to each of the joints connecting the legs to the body and the links of the legs.

The 8D action space represents the torque applied to each of the joints. The observation space has 105 dimensions, and contains the position values and velocities of each of the body parts, as well as the center-of-mass-based external forces on each body part.

The 2-objective MO-Ant problem has one objective for moving in the x-direction and one for moving in the y-direction, with the healthy reward, a contact cost, and the control cost added to each objective to get the final rewards.

The healthy reward gives a fixed reward of +1 for each timestep that the robot remains healthy. The ant is unhealthy if any of the state space values are no longer finite or if the z-coordinate of the torso is outside of the healthy range of $[0.2, 1.0]$ (indicating the ant has fallen).

The contact cost penalizes agents when their external contact forces are too large:

$$contact_cost(F_{contact}) = -w_{contact} \times ||F_{contact}||_2^2, \tag{10}$$

where $w_{contact}$ is the weight of the contact cost and has a value of 5×10^{-4}, and $F_{contact}$ are the external contact forces.

The reward for moving in the x-direction is given by:

$$x_reward = (x_{after_action} - x_{before_action})/dt + ctrl_cost + contact_cost + healthy_reward, \tag{11}$$

where $ctrl_cost$ is the control cost of the action given by Eq. 6 with a value of $w_{ctrl} = 0.5$, and $dt = 0.05$ represents the time between actions. Similarly, the reward for moving in the y-direction is given by:

$$y_reward = (y_{after_action} - y_{before_action})/dt + ctrl_cost + contact_cost + healthy_reward, \tag{12}$$

where y_{after_action} is the y-position of the ant robot after an action has been taken and y_{before_action} is the y-position of the ant before the action was taken.

An episode is terminated early if the ant becomes unhealthy, otherwise it is truncated after 1000 timesteps.

C.5 MO-Swimmer

The MO-Swimmer robot is made up of three links which are connected with two hinge joints. The swimmer is suspended in a “pool", and the goal is to move to the right by applying torque to each of the joints. The 2D action space consists of the torques applied between the links of the robot, and the 8D observation space consists of the positions and velocities of each of the robot’s body parts.

The reward contains a reward for moving forward using Eq. 9 using the front “tip" of the swimmer to measure the displacement and a value of $dt = 0.04$, along with a reward for minimizing the control cost of the action given by Eq. 6 with a value of $w_{ctrl} = 10^{-4}$.

Episodes are truncated after 1000 timesteps, and do not terminate early.

C.6 MO-Humanoid

The MO-Humanoid consists of a 3D bipedal humanoid robot with a torso, arms, and legs. The goal is to walk forward without falling over. The 17D action space represents the torques which are applied to each of the joints. The 348D observation space consists of the positions and velocities of each of the robot’s body parts, the mass and inertia of the rigid body parts relative to the robot’s center of mass, the center of mass based velocity, the actuator force at each joint, and the center of mass based external force on the robot’s body parts.

The two objectives are the forward reward using Eq. 9 with $dt = 0.015$ and minimizing the control cost using Eq. 6 with $w_{ctrl} = 0.1$. A healthy reward and contact cost is added to each of these objectives to get the final rewards. The healthy reward gives a value of +1 for each timestep that the robot is healthy, i.e. the z-coordinate of the torso stays within the healthy range of $[1.0, 2.0]$. The contact cost is given by Eq. 10, with a value of $w_{contact} = 5 \times 10^{-7}$.

The episode terminates early if the robot becomes unhealthy, and otherwise it is truncated after 1000 timesteps.

D Additional Details on the Experimental Setup

This section gives additional information on the experimental setup, including the performance metric definitions and hyper-parameter settings.

D.1 Performance Measures

The hypervolume Zitzler (1999) measures both the convergence of the found POF and the diversity of the policies in it. Given a POF and a reference point $\mathbf{v}_{ref}$, the hyper-volume is calculated by:

$$HV(POF, \mathbf{v}_{ref}) = \cup_{\mathbf{v}^\pi \in POF} \text{volume}(\mathbf{v}_{ref}, \mathbf{v}^\pi), \tag{13}$$

where $\text{volume}(\mathbf{v}_{ref}, \mathbf{v}^\pi)$ is the volume of the hypercube spanned by $\mathbf{v}_{ref}$, $\mathbf{v}_\pi$, and POF is the found POF. The reference vector represents the worst-possible return per objective, i.e. the lower bound of each objective.

Another performance measure is the expected maximum utility Zintgraf et al. (2015), given by:

$$EUM(POF) = \mathbb{E}_{\lambda \in \Lambda} \left[\max_{\mathbf{v}^\pi \in POF} u(\mathbf{v}^\pi, \lambda) \right], \tag{14}$$

where u is the utility function which in this case is the weighted sum, v is the vector of rewards earned using policy π and Λ is a set of weight vectors which are uniformly distributed in the objective space.

A third performance measure which could be used is the sparsity Xu et al. (2020), which evaluates the distance between the points in the found POF. The sparsity metric is calculated using Xu et al. (2020):

$$S(POF) = \frac{1}{|POF| - 1} \sum_{j=1}^{m} \sum_{i=1}^{|POF|-1} \left(\mathcal{L}_j(i) - \mathcal{L}_j(i+1) \right)^2, \tag{15}$$

where $\mathcal{L}_j$ is the sorted list of the jth objective values given all of the policies in the found POF POF, and $\mathcal{L}_j(i)$ is the ith value of $\mathcal{L}_j$. Note that the sparsity is not calculated across the entire objective space, only across the found POF, and as noted by Felten et al. (2024) is somewhat limited on its own. As a result, the sparisty is not included in our main results, but we provide the approximated IQM and boxplots corresponding to the sparsity values for reference.

The final metric is the cardinality of the final POF, which is denoted by $|POF|$. Similarly to the sparsity, the $|POF|$ is rather limited when used on its own, however in combination with the other metrics used it can provide more insight into the results.

For the hypervolume, EUM, and $|POF|$, a higher value is better, and for the sparsity a lower value is better.

D.2 Hyper-parameter Settings

The hyper-parameters for MORL/D and SAC are given in Table 4. These values were selected based on those given by Felten et al. (2024).

D.3 Computing Infrastructure Used

All neural networks were trained using an NVIDIA Volta V100 GPU. The total number of environment steps was used as a stopping criterion, and therefore the total training time varied between approaches. The runtime analysis is provided in Section F.3.

E Parameter Selection

This section gives the results from parameter selection studies which were conducted on the static-τ approach. These were used to determine the number of actions to compare at each timestep and the evaporation rate. Rather than fine-tuning parameters for each environment, which makes it difficult for future comparisons on new test problems, the same parameter values are used for all environments after being compared in the MO-Hopper environment. The MO-Hopper environment was selected because it is a moderately challenging task in that it has a larger action and observation space than the simplest environment (MO-Swimmer), but they are not as large as those of the MO-Ant or MO-Humanoid. Therefore, the MO-Hopper is a suitable environment for determining hyper-parameters which will generalize well across the MO-Gymnasium environments and avoid overfitting the parameters to either simple or complex environments.

Table 4: MORL/D and SAC hyper-parameters

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

E.1 Number of Selected Actions per Step

The number of selected actions per step were selected using the static-τ approach with an evaporation rate of 0.5, and $\{3, 5, 10\}$ selected actions per step were used for comparison on the 2-objective MO-Hopper environment. Table 5 summarizes the performance metric values found for each of the tested number of actions selected per step.

Table 5: Final performance metric values for the various actions selected per step, where the best values are in bold

τ	Hypervolume	EUM	Sparsity	$\lvert POF \rvert$
3	132399.2546	247.85333	157.91959	20
5	130281.10595	244.4278	**129.97767**	21
10	**136242.50312**	**251.60858**	157.16719	**22**

Since selecting 10 actions led to the best hypervolume, EUM, and $|POF|$, it was selected for further comparison by all of the action selection methods introduced in this paper.

E.2 Selection of the Evaporation Rate

Once the number of selected actions per step was determined, the initial evaporation rate for the static-τ approach was selected. Compared values of $\tau = \{0.3, 0.6, 0.7, 0.8\}$ were tested on the 2-objective MO-Hopper environment. For each of these, 10 actions were sampled from the buffer based on the results in the previous sub-section.

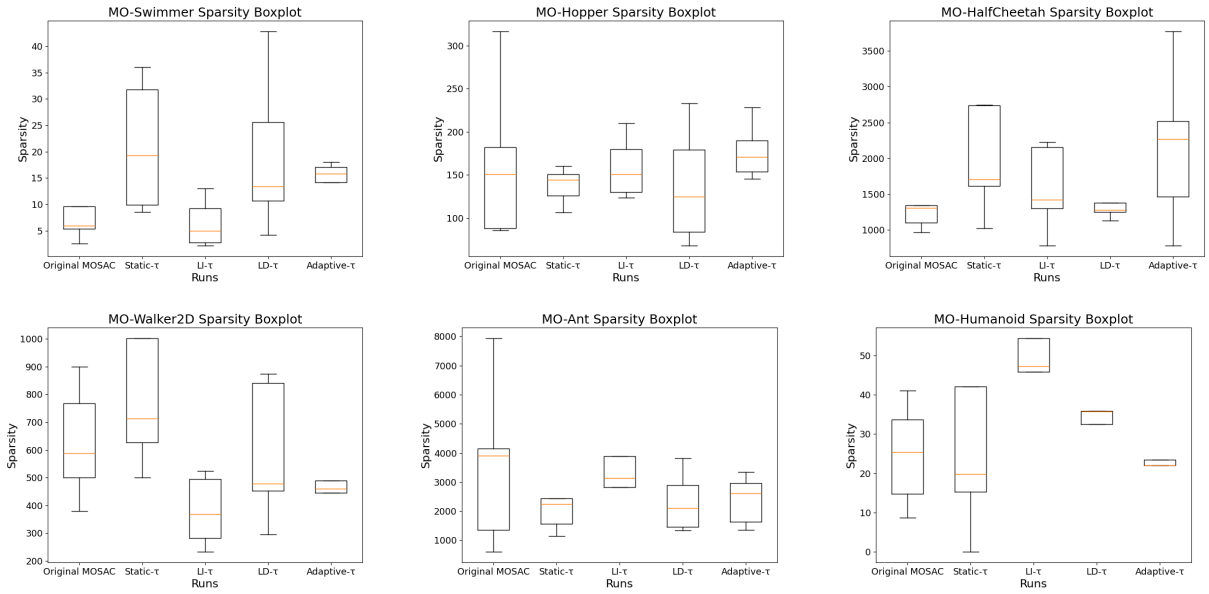


Figure 5: Boxplots of the sparsity for each environment across 5 runs, note that a lower value is better.

Table 6: Final performance metric values for the various evaporation rates, where the best values are in bold

τ	Hypervolume	EUM	Sparsity	$\lvert POF \rvert$
0.3	131946.42046	248.25093	289.76139	19
0.5	**136242.50312**	**251.60858**	157.16719	22
0.6	129814.48533	242.29363	100.38522	30
0.7	136065.49577	248.95066	**83.32114**	**30**
0.8	129092.50078	243.34102	121.73243	20

Since it had the best $|POF|$, as well as nearly the same hypervolume and EUM values, a value of $\tau = 0.7$ was selected for further comparison. Additionally, the same value of $\tau = 0.7$ was used as the initial value for the adaptive-τ approach.

F Additional Experimental Results

This section gives some additional experimental results, including the approximated IQM of the final performance metric values, sparsity values, and runtime analysis.

F.1 IQM of Final Performance Metric Values

F.2 Sparsity Results

To illustrate the variance of the final sparsity in each environment independently, Figure 5 gives boxplots of the final sparsity values for all environments.

Table 7: The approximated IQM of the final performance metric values, where “HV" indicates the hypervolume, “EUM" indicates the expected utility, “$|POF|$" indicates the cardinality of the POF, and “S" indicates the sparsity. For sparsity, a lower value is better and for the remaining performance measures, a higher value is better. Results for each metric are given in the format “value/rank", and the best results are highlighted in bold.

Metric	Original MOSAC	Static τ	LI-τ	LD-τ	Adaptive-τ
MO-Swimmer					
HV	**12598.94130**	12330.83432	12577.09606	12391.02275	12320.05741
EUM	10.12315	9.14701	**10.14271**	9.45210	9.08576
S	6.95625	20.33849	**5.64429**	16.53760	15.69246
$\|POF\|$	**55.66667**	47.33333	54.00000	50.00000	47.66667
MO-Hopper					
HV	130212.86815	132927.64495	134773.62355	**137023.68927**	134538.30155
EUM	245.28546	247.58268	**251.31837**	251.21055	249.85178
S	140.52149	140.40061	153.69331	**129.27369**	171.51256
$\|POF\|$	21.66667	22.66667	21.33333	**29.66667**	23.33333
MO-HalfCheetah					
HV	**25162.84262**	20652.43688	23498.10820	21862.56997	18229.53829
EUM	257.57856	262.16267	**262.17836**	254.42789	259.06863
S	**1252.36411**	2020.23987	1627.96370	1303.42912	2083.62948
$\|POF\|$	**51.66667**	47.66667	50.66667	48.33333	41.33333
MO-Walker2D					
HV	**66280.13958**	62536.33220	65905.88756	64465.93061	62359.67141
EUM	**170.04696**	165.69951	166.44883	167.54495	156.53987
S	618.53918	780.60800	**381.81767**	590.49273	465.21705
$\|POF\|$	32.33333	30.66667	**34.00000**	**34.00000**	32.33333
MO-Ant					
HV	181309.01441	190319.99323	193728.70870	**204786.11131**	204055.66013
EUM	306.25824	327.61432	324.26281	333.74656	**335.34535**
S	3138.09445	**2082.11305**	3284.65249	2151.29348	2401.73219
$\|POF\|$	15.33333	**18.00000**	15.00000	16.66667	15.66667
MO-Humanoid					
HV	282443.00776	276949.40691	283026.68961	283264.25205	**283773.99606**
EUM	433.20737	429.84406	433.92322	434.20929	**436.25185**
S	24.60182	**25.75401**	49.20288	34.71133	22.49014
$\|POF\|$	8.33333	6.33333	5.66667	**8.66667**	8.33333

F.3 Runtime Comparison

Figure 6 gives graphs of the median runtimes of each algorithm on each problem. From these graphs, it is clear that including the additional action selections at each step does lead to an increase of roughly 100% in the runtime compared to the original MOSAC approach. The introduced static-τ, LI-τ, LD-τ, and adaptive-τ approaches all had similar runtimes to each other. The increase in runtime is caused by a number of reasons: the number of actions which are selected at each timestep, the number of actions which are maintained in the archive, and the implementation of the algorithms themselves. For the experimentation which was done for this paper, since all tests were run using a GPU, certain calculations were done in parallel to improve the runtime of the proposed algorithms, however it may be possible to further optimize the code.

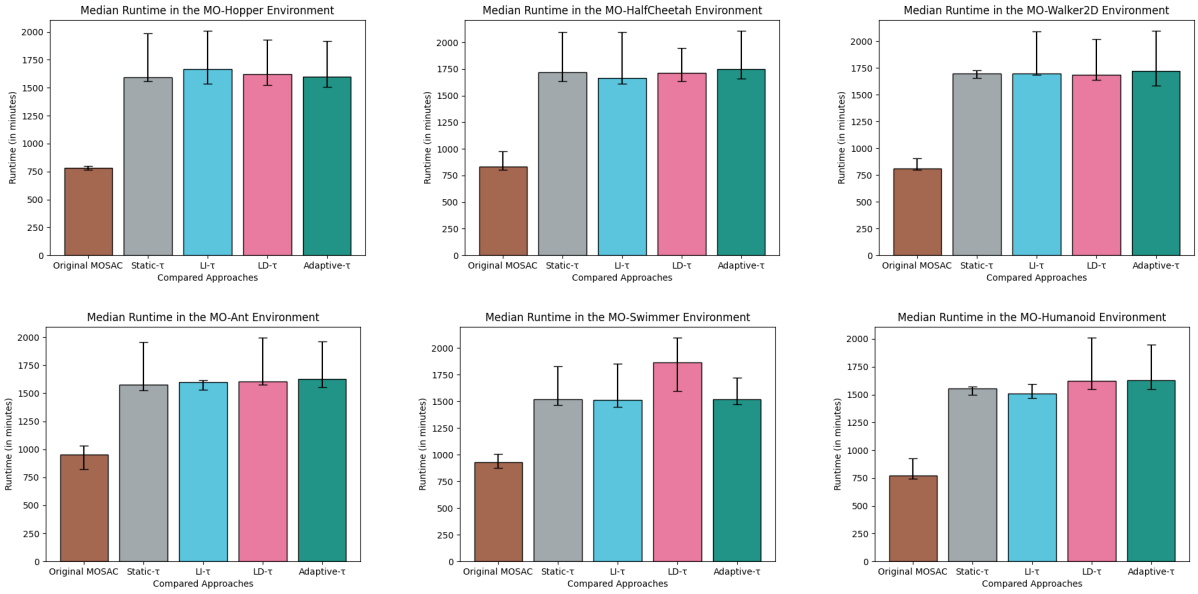


Figure 6: The median runtimes on each problem, the error bars indicate the minimum and maximum run times of each approach.

References

Alegre, L. N., Bazzan, A. L. C., Roijers, D. M., Nowé, A., and da Silva, B. C. (2023). Sample-efficient multi-objective learning via generalized policy improvement prioritization. In *Proceedings of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023)*.

Amirabadi, R. K. and Fard, O. S. (2024). Combining hybrid metaheuristic algorithms and reinforcement learning to improve the optimal control of nonlinear continuous-time systems with input constraints. *Computers and Electrical Engineering*, 116:109179.

Castelletti, A., Pianosi, F., and Restelli, M. (2013). A multiobjective reinforcement learning approach to water resources systems operation: Pareto frontier approximation in a single run. *Water Resources Research*, 49(6):3476–3486.

Chen, L., Zhu, G., Liang, W., and Wang, Y. (2023). Multi-objective reinforcement learning approach for trip recommendation. *Expert Systems with Applications*, 226:120145.

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

Felten, F., Alegre, L. N., Nowé, A., Bazzan, A. L. C., Talbi, E. G., Danoy, G., and Silva, B. C. da. (2023). A toolkit for reliable benchmarking and research in multi-objective reinforcement learning. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS 2023)*.

Felten, F., Danoy, G., Talbi, E.-G., and Bouvry, P. (2022). Metaheuristics-based exploration strategies for multi-objective reinforcement learning. In *Proceedings of the 14th International Conference on Agents and Artificial Intelligence (ICAART 2022)*, volume 2, pages 662–673.

Felten, F., Talbi, E.-G., and Danoy, G. (2024). Multi-objective reinforcement learning based on decomposition: A taxonomy and framework. *Journal of Artificial Intelligence Research*, 79:679–723.

Gambardella, L. M. and Dorigo, M. (1995). Ant-q: A reinforcement learning approach to the traveling salesman problem. In Prieditis, A. and Russell, S., editors, *Machine Learning Proceedings 1995*, pages 252–260. Morgan Kaufmann, San Francisco (CA).

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., and Levine, S. (2019). Soft actor-critic algorithms and applications.

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

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

Huynh, T. N. Y., Nguyen, A. T., Ahn, Y., Oo, B. L., and Lim, B. T. (2025). Multi objectives reinforcement learning for smart buildings: A systematic review of algorithms, applications and future perspectives. *Energy and Buildings*, 345:116045.

Keller, A. A. (2019). *Multi-Objective Optimization in Theory and Practice II: Metaheuristic Algorithms*. Bentham Science Publishers Ltd., Publisher Address.

Kharitonov, A., Abani, J. I., Nahhas, A., and Turowski, K. (2025). Literature survey on combining machine learning and metaheuristics for decision-making. *Procedia Computer Science*, 253:199–208. 6th International Conference on Industry 4.0 and Smart Manufacturing.

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

Lu, H., Herman, D., and Yu, Y. (2023). Multi-objective reinforcement learning: Convexity, stationarity and pareto optimality. In *The Eleventh International Conference on Learning Representations (ICLR 2023)*.

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

Reymond, M., Bargiacchi, E., and Nowé, A. (2022). Pareto conditioned networks. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022)*, pages 1110–1118.

Roijers, D., Steckelmacher, D., and Nowe, A. (2018). Multi-objective reinforcement learning for the expected utility of the return. In *Adaptive Learning Agents (ALA) workshop at AAMAS/ICML/IJCAI 2018*.

Saiki, T. and Arai, S. (2023). Flexible traffic signal control via multi-objective reinforcement learning. *IEEE Access*, 11:75875–75883.

Valencia, D., Jia, J., Li, R., Hayashi, A., Lecchi, M., Terezakis, R., Gee, T., Liarokapis, M., MacDonald, B. A., and Williams, H. (2023). Comparison of model-based and model-free reinforcement learning for real-world dexterous robotic manipulation tasks. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 871–878.

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

Venske, S. M. S., de Almeida, C. P., and Delgado, M. R. (2024). Metaheuristics and machine learning: an approach with reinforcement learning assisting neural architecture search. *Journal of Heuristics*, 30:199–224.

Xu, J., Tian, Y., Ma, P., Rus, D., Sueda, S., and Matusik, W. (2020). Prediction-guided multi-objective reinforcement learning for continuous robot control. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, pages 10607–10616.

Yang, R., Sun, X., and Narasimhan, K. (2019). A generalized algorithm for multi-objective reinforcment learning and policy adaptation. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, pages 14636–14647.

Zintgraf, L. M., Kanters, T. V., Roijers, D. M., Oliehoek, F. A., and Beau, P. (2015). Quality assessment of MORL algorithms: A utility-based approach. In *Benelearn 2015: Proceedings of the 24th Annual Machine Learning Conference of Belgium and the Netherlands*.

Zitzler, E. (1999). *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology Zurich (ETH).