

# Particle Swarm Algorithm for Multi-objective Reinforcement Learning

Teresa Becerril Torres<sup>a,\*</sup>, Carlos Ignacio Hernández Castellanos<sup>b,1</sup> and Roberto Santana Hermida<sup>c,1</sup>

<sup>a</sup>Postgraduate Program in Computer Science and Engineering, UNAM

<sup>b</sup>Applied Mathematics and Systems Research Institute, UNAM

<sup>c</sup>Intelligent Systems Group, UPV/EHU

**Abstract.** The application of multi-objective bio-inspired algorithms to multi-objective reinforcement learning problems has gained attention due to their ability to approximate diverse Pareto-optimal solutions. However, the impact of different search mechanisms remains largely unexplored. In this work, we propose adapting the classical Optimized Multi-Objective Particle Swarm Optimization algorithm for these problems. Its simplicity, fast convergence, and population-based nature make it a promising alternative to more expensive strategies. We evaluate the algorithm on continuous problems and compare it with four multi-objective evolutionary algorithms using three performance indicators. Our preliminary results show competitive performance, highlighting the potential of swarm-based algorithms in multi-objective reinforcement learning.

## 1 Introduction

Typically, real-world problems involve optimizing several objectives simultaneously, as well as a series of decisions. Take autonomous driving [11], for example: minimizing travel time and maximizing safety are inherently contradictory. Higher speeds reduce travel time but compromise safety; prioritizing safety typically results in a longer trip duration. These problems require approaches capable of generating solutions that reflect different trade-offs between objectives, known as efficient or Pareto-optimal solutions.

Multi-objective reinforcement learning (MORL) [10] offers a robust framework for addressing these problems, modeling them as multi-objective Markov decision processes (MOMDPs) [10, 27]. Among the approaches to solving MORL problems, multi-objective bio-inspired methods have demonstrated great potential for generating diverse sets of policies that approximate the Pareto front [1, 22]. A particularly attractive approach is Multi-Objective Particle Swarm Optimization (MOPSO), which uses Pareto dominance-based leader selection and external archive to preserve non-dominated solutions [7]. This method offers: (i) fast convergence (beneficial in high-dimensional MORL problems); (ii) population-based search for improved Pareto front approximation and parallel exploration of multiple solutions; (iii) algorithmic simplicity that rivals more expensive methods; and (iv) particle swarm optimization (PSO) has a competitive performance on continuous MORL problems, as shown in recent benchmark study [22]. Despite these strengths, MOPSO remains underexplored in MORL. This integration could leverage its efficiency

and low computational cost for Pareto front approximation in complex environments, providing a practical alternative to both gradient-based and more expensive multi-objective evolutionary algorithms.

In this work, we propose adapting the Optimized Multi-Objective Particle Swarm Optimization (OMOPSO) algorithm [25] for MORL problems. OMOPSO enhances classical MOPSO with an  $\varepsilon$ -archive for non- $\varepsilon$ -dominated solutions, a crowding distance metric for diversity, and a Pareto-dominance-based leader selection strategy. These mechanisms efficiently guide exploration and preserve a well-distributed set of policies.

The key contributions of this paper are as follows:

- **Adaptation of OMOPSO for MORL:** We develop an OMOPSO variant where each particle represents a policy parameterized by the weights of a neural network.
- **Evaluation in Benchmark problems:** We evaluate our method on MuJoCo problems from the MO-Gymnasium suite, featuring continuous dynamics and two to four objectives.
- **Comparison with State-of-the-Art MOEAs:** We compare our approach with four widely used MOEAs, using three performance indicators.

The rest of the paper is organized as follows: Section 2 introduces background on MORL. Section 3 reviews related work. Section 4 presents our algorithm. Section 5 details experiments and results. Section 6 concludes the paper and discusses future work.

## 2 Background

Multi-objective reinforcement learning problems can be modeled as multi-objective Markov decision processes, which are formally defined as follows:

**Definition 1** (Van Moffaet, 2016). *A MOMDP is represented by a 5-tuple  $(S, A, T, R, \gamma)$ , where:*

- $S$  is the finite set of states.
- $A$  is the finite set of actions.
- $T : S \times A \times S \rightarrow [0, 1]$  is the transition function, where  $T(s' | s, a)$  returns the probability (discrete) or probability density function (continuous) of reaching state  $s'$  after action  $a$  in state  $s$ .
- $R : S \times A \times S \rightarrow \mathbb{R}^k$  is the reward function, where  $R(s, a, s')$  returns a  $k$ -dimensional reward vector.
- $\gamma \in [0, 1]$  is the discount factor.

\* Corresponding Author. Email: tbt2415@gmail.com

<sup>1</sup> Equal contribution.

In classical reinforcement learning, the agent makes decisions following a policy  $\pi$ . In this context, the policy is parameterized by the weights of the neural network. The formal definition is given below:

**Definition 2** (Hayes et al., 2022). *A parameterized policy  $\pi_\theta : S \times A \rightarrow [0, 1]$  assigns to each state  $s \in S$  a probability distribution over actions  $a \in A$ , where  $\pi_\theta(a | s)$  denotes the probability of selecting action  $a$  given state  $s$ , and  $\theta$  represents the weights of the neural network.*

In MOMDPs, the agent receives a vector reward rather than a scalar, leading to a set of non-dominated policies  $\Pi_\Theta$  rather than an optimal policy [21]. Each policy, parameterized by  $\theta \in \Theta$ , represents a distinct trade-off among objectives. The state-value function of a parameterized policy  $\pi_\theta$  is defined as:

**Definition 3** (Hayes et al., 2022).  *$V^{\pi_\theta}(s) \in \mathbb{R}^k$  specifies the expected discounted cumulative reward vector obtained when starting from state  $s$  and following policy  $\pi_\theta \in \Pi_\Theta$ :*

$$V^{\pi_\theta}(s) = \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid S_t = s, \pi_\theta \right]. \quad (1)$$

Since rewards are vectors, policies are compared using Pareto dominance, which induces a partial order over the policy space [27]. Formally, a policy  $\pi_\theta \in \Pi_\Theta$  dominates another policy  $\pi_{\theta'}$ , denoted by  $\pi_\theta \preceq \pi_{\theta'}$ , if and only if [10]:

$$\begin{aligned} 1. & V_i^{\pi_\theta}(s) \geq V_i^{\pi_{\theta'}}(s), \quad \forall i \in \{1, \dots, k\}, \\ 2. & V_i^{\pi_\theta}(s) > V_i^{\pi_{\theta'}}(s), \quad \exists i \in \{1, \dots, k\}. \end{aligned} \quad (2)$$

The set of all efficient policies is known as the Pareto set. Its image, which represents the objective space, is formally known as the Pareto front (PF).

**Definition 4** (Hayes et al., 2022). *The Pareto set  $P_{\Pi_\Theta}$  is the set of all non-dominated policies in  $\Pi_\Theta$ :*

$$P_{\Pi_\Theta} = \{\pi_\theta \in \Pi_\Theta \mid \nexists \pi_{\theta'} \in \Pi_\Theta : \pi_{\theta'} \preceq \pi_\theta\}. \quad (3)$$

**Definition 5** (Hayes et al., 2022). *The Pareto front  $F(P_{\Pi_\Theta})$  is the set of state-value function values of the non-dominated policies:*

$$F(P_{\Pi_\Theta}) = \{V^{\pi_\theta} \mid \pi_\theta \in P_{\Pi_\Theta}\}. \quad (4)$$

### 3 Related work

This section presents relevant work in two key research areas underlying this proposal: multi-objective reinforcement learning and multi-objective bio-inspired algorithms.

#### 3.1 Multi-Objective Reinforcement Learning

To address multi-objective Markov decision processes, the literature proposes two main strategies. Single-policy methods convert the problem into a scalar problem using scalarization functions, such as the weighted sum method [15], allowing the application of standard reinforcement learning techniques. Examples include EUPG [20] and PA2D-MORL [12], which combine scalarization with deep learning and policy gradients.

In contrast, multi-policy approximates the Pareto front by learning a set of policies that reflect different trade-offs between objectives.

Examples include Pareto Q-learning [28], C-MORL [18], and hybrid approaches such as Borrel et al. [4], which combine Q-learning with the R2 indicator. Predictive strategies, such as Xu et al. [29], use gradient estimation to guide exploration. More recent work by Callaghan et al. [5] extended evolution-guided policy gradient learning [17] to multi-objective scenarios, proposing efficient exploration. Additionally, Hernández and Santana [22] developed a benchmarking framework to assess single- and multi-objective evolutionary algorithms in continuous MORL problems, noting the influence of environment complexity on performance. This work adopts this strategy due to its ability to produce diverse and representative solutions. Furthermore, their results showed that single-objective Particle Swarm Optimization was able to find promising solutions more quickly than other approaches. Thus, it would be interesting to analyze the behavior of a multi-objective version.

#### 3.2 Multi-Objective Bio-Inspired Algorithms

Multi-objective bio-inspired algorithms are metaheuristics designed to explore search spaces with multiple objectives [26]. They are typically grouped into multi-objective evolutionary algorithms (MOEAs) and multi-objective swarm-based algorithms.

MOEAs, inspired by natural evolution, are commonly classified into three categories [24]: Dominance-based, such as NSGA-II [9] and SPEA2 [34], employing Pareto dominance and diversity preservation; Decomposition-based, like MOEA/D [32] and NSGA-III [8], decomposing the problem into scalar subproblems via scalarization; and Indicator-based, such as SMS-EMOA [3] and IBEA [13], evaluating solutions based on quality indicators like hypervolume or inverted generational distance. Reinforcement learning has recently been incorporated into the design of MOEAs, such as the algorithm by Xue et al. [31], which applies Q-learning to adjust weight vectors in MOEA/D. Deep learning-inspired approaches, such as PSMGD [30], utilize stochastic periodic multi-gradient descent to accelerate convergence.

Multi-objective swarm-based algorithms, inspired by collective behavior in nature, such as ants [2] or birds [6]. Multi-Objective Particle Swarm Optimization [6] extends the classical PSO [16], enabling particles to collaboratively explore based on individual and leader experience, guided by Pareto dominance. For example, Nebro et al. [19] propose an auto-configured MOPSO that dynamically tunes parameters to improve efficiency and flexibility. Although significant advances have been made in the field, further analysis of different search mechanisms remains largely unexplored.

### 4 Adapting OMOPSO to MORL Problems

In this work, we adapt the OMOPSO algorithm [25] to MORL problems by representing each particle in the swarm as a policy parameterized by the weights of a neural network. Unlike gradient-based methods, our method directly optimizes network weights using swarm-based strategies. Each particle corresponds to a policy  $\pi_\theta$ , evaluated in a multi-objective problem by computing the average reward vector over several episodes.

Algorithm 1 presents the general pseudocode of OMOPSO-RL. The algorithm begins by initializing a swarm of  $n$  particles, each representing a neural network policy with weights randomly sampled from  $[-1, 1]$ , zero initial velocity, and its best individual position ( $pBest_i$ ) set to its starting point. Each policy is evaluated over  $n_{eval}$  episodes in the environment to compute an average reward vector. Next, the leader set is initialized with non-dominated particles,

maintaining diversity via crowding distance when its size exceeds  $n$ . Subsequently, the external  $\varepsilon$ -archive is updated by merging it with the leader set, retaining only non- $\varepsilon$ -dominated policies, again using crowding distance to resolve excess entries. Crowding distances are computed to encourage exploration into sparsely populated regions of the objective space.

The algorithm then enters a main loop that runs for  $g_{\max}$  generations. In each generation, each particle is updated: a leader  $L_i$  is chosen using a binary tournament based on the crowding distance. The particle’s velocity is updated using the equation  $v_i = w \cdot v_i + r_1 \cdot (pBest_i - x_i) + r_2 \cdot (L_i - x_i)$ , where  $w \in [0.1, 0.5]$  is the inertia weight,  $c_1, c_2 \in [1.5, 2]$  are cognitive and social coefficients, and  $r_1, r_2 \in [0, 1]$  are random values. Here,  $v_i \in \mathbb{R}^n$  and  $x_i \in \mathbb{R}^n$  are the particle’s velocity and position in generation  $g$ , and  $n$  is the search space dimension. Following this, the position is updated as  $x_i = x_i + v_i$ . Polynomial mutation is applied to the position, the resulting policy is evaluated, and if the new position dominates or is non-dominated relative to the previous  $pBest_i$ , it is updated. After all particles are processed, the leader set and  $\varepsilon$ -archive are refreshed with the newly found non-dominated policies, and crowding distances are recomputed for the next generation. After completing all iterations, the algorithm returns the  $\varepsilon$ -archive as the final approximation of the Pareto front.

#### 4.1 Neural Networks as Policies

Each particle represents a fully connected feed-forward neural network policy. In environments with discrete action spaces, the network outputs a probability distribution over actions using a softmax activation, and the action is selected via stochastic sampling. In continuous action spaces, the network produces outputs with sigmoid activation, which are then linearly scaled to match the environment’s action bounds. Formally, the policy is defined as:

$$\pi_{\theta}(s) = \begin{cases} \text{Categorical}(\text{Softmax}(f_{\theta}(s))) & \text{if } \mathcal{A} \text{ is discrete} \\ a_{low} + (a_{high} - a_{low}) \cdot \sigma(f_{\theta}(s)) & \text{if } \mathcal{A} \text{ is continuous} \end{cases}$$

where  $f_{\theta}(s)$  denotes the forward pass of the neural network parameterized by  $\theta$  and  $\mathcal{A}$  represents the action space of the environment. Here,  $a_{high}$  and  $a_{low}$  are the lower and upper bounds of the continuous action space, respectively. Thus, the resulting policy is stochastic in discrete action spaces and deterministic in continuous ones.

#### 4.2 $\varepsilon$ -Dominated External Archive

The external archive [23, 24], stores policies not  $\varepsilon$ -dominated by any other policy. For each new policy, it is added only if no existing policy  $\varepsilon$ -dominates it. Duplicate policies (identical weight vectors) are ignored. If the archive exceeds its size limit, solutions are selected based on crowding distance to maintain diversity.

## 5 Experiments

This section outlines the objectives of the experimental study, the experimental setup, and a summary of the obtained results. All code and experiments are publicly available<sup>2</sup>.

The general objective of this work is to adapt and analyze a novel multi-objective reinforcement learning algorithm based on OMOPSO. Our specific objectives are: (i) to adapt the OMOPSO algorithm for MORL problems; and (ii) to conduct a comparative

---

#### Algorithm 1 OMOPSO-RL

---

**Require:** Environment  $env$ , evaluation episodes  $n_{eval}$ , swarm size  $n$ , generations  $g_{\max}$ , OMOPSO parameters ( $w, c_1, c_2$ ), mutation probability  $pm$ , distribution index  $\eta$ , network architecture  $\ell$

**Ensure:** External archive of non-dominated policies:

```

1: swarm  $\leftarrow$  initialize_swarm( $n, \ell, env$ )
2: leaders  $\leftarrow$  update_leaders(swarm)
3: epsilon_archive  $\leftarrow$  update_epsilon_archive(leaders)
4: crow_dist  $\leftarrow$  calculate_crowding_distance(leaders)
5: for  $g = 1$  to  $g_{\max}$  do
6:   for each particle  $i = 1$  to  $n$  do
7:      $L_i \leftarrow$  select_leader(crow_dist)
8:      $r_1, r_2 \sim \mathcal{U}(0, 1)$ 
9:      $v_i \leftarrow wv_i + c_1r_1(pBest_i - x_i) + c_2r_2(L_i - x_i)$ 
10:     $x_i \leftarrow x_i + v_i$ 
11:     $x_i \leftarrow$  polynomial_mutation( $x_i, pm, \eta$ )
12:    fitness  $\leftarrow$  evaluate_policy( $x_i, n_{eval}, env$ )
13:     $pBest_i \leftarrow$  update_pBest(fitness,  $pBest_i$ )
14:  end for
15:  leaders  $\leftarrow$  update_leaders(swarm)
16:  epsilon_archive  $\leftarrow$  update_epsilon_archive(leaders)
17:  crow_dist  $\leftarrow$  calculate_crowding_distance(leaders)
18: end for
19: return epsilon_archive

```

---

study with state-of-the-art multi-objective evolutionary algorithms to analyze the effect of OMOPSO search operators.

To evaluate OMOPSO-RL, we compared it against four established algorithms from the Pymoo<sup>3</sup> framework: NSGA-II [9], NSGA-III [8], MOEA/D [32], and SMS-EMOA [3]. The algorithms were tested on MuJoCo environments from the MO-Gymnasium<sup>4</sup> benchmark suite: mo-reacher-v5, mo-hopper-v5, mo-walker2d-v5, mo-ant-v5, mo-humanoid-v5, mo-halfcheetah-v5, and mo-swimmer-v5, all of which are stochastic. Performance was measured using three standard multi-objective indicators: Hypervolume (HV) [33], Inverted Generational Distance Plus (IGD+) [14], and Generational Distance Plus (GD+) [14]. From the union of all non-dominated solutions generated by all algorithms, we constructed an approximate reference PF. This front was used to calculate IGD+ and GD+.

The experiments used the following parameters: a population size of  $n = 20$ ,  $g_{\max} = 100$  generations, and  $n_{eval} = 5$  evaluation episodes. The policy network consisted of an input layer (whose size is defined by the environment’s observation space), three hidden layers with four neurons each, and an output layer (matching the environment’s action space). In this work the decision variables ranges from 142 to 1,521, which depend on the environment. For OMOPSO-RL, the inertia weight decreased linearly from  $w = 0.5$  to  $w = 0.1$ , while the cognitive and social coefficients were set to  $c_1 = 1.5$  and  $c_2 = 1.5$ , respectively. The mutation probability was  $pm = 0.1$ , with a distribution index of  $\eta = 15$ .

Each algorithm was run over 30 independent trials per environment to capture stochastic variability. Performance was analyzed using descriptive statistics and pairwise Wilcoxon rank-sum tests ( $\alpha = 0.05$ ) to assess statistical significance between OMOPSO-RL and each MOEA. The Friedman test with Nemenyi post-hoc comparisons was used to analyze critical differences and evaluate rankings by performance indicator and overall rankings.

Experiments were conducted using Python 3.11.13 on Google Compute Engine with an NVIDIA Tesla T4 GPU (CUDA 12.4), 12.7 GB RAM (CPU), 15.0 GB (GPU), and 235.7 GB disk space.

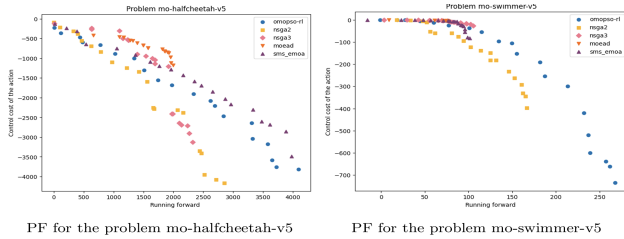
<sup>2</sup> <https://github.com/Terebece/OMOPSO-RL.git>

<sup>3</sup> <https://pymoo.org/>

<sup>4</sup> <https://mo-gymnasium.farama.org/environments/mujoco/>

## 5.1 Pareto Front for the MORL Problems

We present Pareto fronts obtained in the two-objective problems mo-halfcheetah-v5 and mo-swimmer-v5, selected for their clear visualization and intuitive interpretation of algorithmic behavior. The mo-halfcheetah-v5 models a 2-dimensional cheetah-like agent that aims to achieve fast and energy-efficient locomotion. In contrast, mo-swimmer-v5 presents a snake-like agent that focuses on coordinating its segments for effective propulsion. Figure 1 shows the PFs

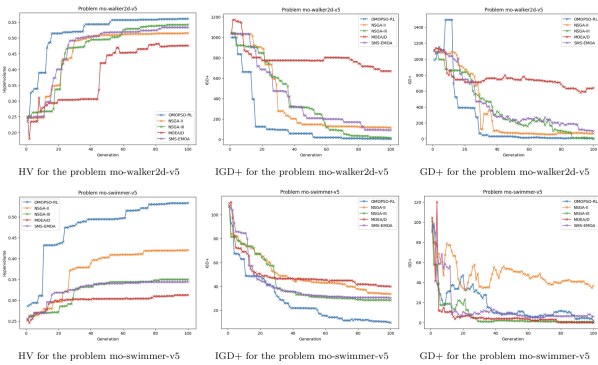


**Figure 1.** PFs for the problems mo-halfcheetah-v5 and mo-swimmer-v5.

of the algorithms in mo-halfcheetah-v5 (left) and mo-swimmer-v5 (right), highlighting OMOPSO-RL (blue circles) and SMS-EMOA (purple triangles). In mo-halfcheetah-v5, both algorithms produce well-distributed PFs with high running forward rewards, but the front of SMS-EMOA dominates that of OMOPSO-RL. In mo-swimmer-v5, OMOPSO-RL yields a more diverse front balancing rewards and costs, whereas SMS-EMOA focuses on low-cost solutions with limited rewards. Overall, OMOPSO-RL produces broader PFs in some environments, while SMS-EMOA can achieve superior dominance in others.

## 5.2 Evolution Graphs of Performance Indicators

Figure 2 presents the evolution of the HV, IGD+, and GD+ metrics for the mo-walker2d-v5 (top row) and mo-swimmer-v5 (bottom row) problems. In mo-walker2d-v5, OMOPSO-RL significantly outperformed the other algorithms from the early generations on all three metrics. It maintained its dominance in HV and IGD+, while in GD+, it was almost caught by NSGA-III towards the end. MOEA/D consistently showed the worst performance in all cases. In



**Figure 2.** Evolution of the indicators for the problems mo-walker2d-v5 and mo-swimmer-v5.

mo-swimmer-v5, OMOPSO-RL significantly outperformed the other early-generation algorithms in HV and IGD+, while in GD+ it was outperformed by most algorithms but improved in later generations. MOEA/D consistently showed the worst performance in HV and IGD+, while NSGA-II showed the worst performance in GD+.

## 5.3 Descriptive statistics

Table 1 shows the performance comparison results for algorithms in MuJoCo environments. Focusing on OMOPSO-RL, the analysis by performance indicator is as follows:

OMOPSO-RL achieves the highest HV on mo-walker2d-v5 (0.541) and mo-swimmer-v5 (0.502), demonstrating its effectiveness in environments with relatively simple or moderately structured locomotion. However, it performs poorly in mo-humanoid-v5, where MOEA/D (0.249) and NSGA-III (0.243) outperform it, with OMOPSO-RL scoring only 0.138. The humanoid’s more structured locomotion and unstable dynamics make OMOPSO-RL’s global exploration insufficient for fine-grained convergence. Similarly, in mo-halfcheetah-v5, SMS-EMOA (0.469) surpasses OMOPSO-RL, reflecting a better balance between diversity and convergence. Overall, OMOPSO-RL excels in specific environments but struggles with highly structured and complex locomotion, whereas NSGA-II and SMS-EMOA show more consistent HV performance, and MOEA/D has the weakest.

OMOPSO-RL achieves the best IGD+ in mo-walker2d-v5 (10.871) and mo-swimmer-v5 (8.076), indicating effective convergence in medium-complexity problems where exploration directs the population toward the reference front. However, OMOPSO-RL records the worst IGD+ in mo-reacher-v5 (2.139) and performs poorly in mo-ant-v5 (156.322), highlighting limited robustness in high-precision or highly coordinated environments. SMS-EMOA consistently achieves the best IGD+ (e.g., mo-reacher-v5: 1.398), demonstrating stronger convergence, while MOEA/D shows inconsistent performance, from worst in mo-walker2d-v5 to second-best in mo-humanoid-v5. Overall, OMOPSO-RL offers diversity and competitive convergence in medium-complexity problems but is outperformed by SMS-EMOA and NSGA-II in most cases.

OMOPSO-RL shows moderate proximity to the reference front, achieving the best GD+ in mo-walker2d-v5 (47.229) and the second-best in mo-halfcheetah-v5 (111.760), suggesting it can approach the Pareto front in structured but not overly complex locomotion. However, it records the worst GD+ in mo-reacher-v5 (3.563) and mo-ant-v5 (210.874), reflecting limited accuracy in environments requiring precise control or high coordination. NSGA-III generally achieves the best GD+, while MOEA/D excels in mo-halfcheetah-v5 (41.323) and mo-swimmer-v5 (1.979), indicating strong convergence. Overall, OMOPSO-RL’s higher GD+ values suggest a prioritization of diversity over convergence, while NSGA-II performs worst in most cases.

## 5.4 General Analysis

To comprehensively evaluate OMOPSO-RL’s performance, we performed pairwise statistical tests and a critical difference analysis across all environments by performance indicator.

We performed pairwise comparisons using the Wilcoxon rank-sum test. Table 1 indicates whether OMOPSO-RL’s performance was statistically better ( $\uparrow$ ), worse ( $\downarrow$ ), or non-significant ( $\leftrightarrow$ ) compared to the MOEAs. OMOPSO-RL achieved significantly better HV performance in mo-walker2d-v5 and mo-swimmer-v5, outperforming all MOEAs. However, it performed worse than NSGA-II and MOEA/D on the mo-ant-v5 and mo-humanoid-v5 environments, highlighting its limitations in more complex problems. For IGD+, OMOPSO-RL showed significantly worse performance in mo-reacher-v5 and mo-humanoid-v5, while it performed best in mo-walker2d-v5 and mo-swimmer-v5. For GD+, OMOPSO-RL underperformed in most en-

**Table 1.** Performance comparison for algorithms in MuJoCo environments. Best result in black and second in gray.

Environment	OMOPSO-RL	NSGA-II	NSGA-III	MOEA/D	SMS-EMOA
<b>Hypervolume</b>					
mo-reacher-v5	0.192 ± 0.005	0.190 ± 0.008 ↔	0.171 ± 0.002 ↑	0.165 ± 0.004 ↑	<b>0.195 ± 0.006 ↓</b>
mo-hopper-v5	0.395 ± 0.001	<b>0.408 ± 0.001 ↓</b>	0.353 ± 0.000 ↑	0.390 ± 0.000 ↑	0.407 ± 0.001 ↓
mo-walker2d-v5	<b>0.541 ± 0.007</b>	0.457 ± 0.002 ↑	0.511 ± 0.010 ↑	0.379 ± 0.090 ↑	0.480 ± 0.010 ↑
mo-ant-v5	0.277 ± 0.009	<b>0.325 ± 0.004 ↓</b>	0.297 ± 0.006 ↓	0.247 ± 0.005 ↑	0.310 ± 0.004 ↓
mo-humanoid-v5	0.138 ± 0.041	0.243 ± 0.060 ↓	0.229 ± 0.069 ↓	<b>0.249 ± 0.072 ↓</b>	0.141 ± 0.049 ↔
mo-halfcheetah-v5	0.387 ± 0.005	0.377 ± 0.001 ↑	0.375 ± 0.002 ↑	0.351 ± 0.001 ↑	<b>0.469 ± 0.002 ↓</b>
mo-swimmer-v5	<b>0.502 ± 0.001</b>	0.414 ± 0.003 ↑	0.355 ± 0.004 ↑	0.322 ± 0.002 ↑	0.350 ± 0.002 ↑
<b>IGD+</b>					
mo-reacher-v5	2.139 ± 0.352	1.923 ± 0.255 ↓	1.580 ± 0.088 ↓	2.006 ± 0.152 ↔	<b>1.398 ± 0.129 ↓</b>
mo-hopper-v5	17.938 ± 0.321	8.804 ± 0.966 ↓	133.141 ± 0.237 ↑	18.475 ± 0.170 ↑	<b>7.864 ± 0.724 ↓</b>
mo-walker2d-v5	<b>10.871 ± 7.560</b>	124.175 ± 2.252 ↑	41.937 ± 9.624 ↑	413.808 ± 269.170 ↑	105.374 ± 15.754 ↑
mo-ant-v5	156.322 ± 14.404	132.472 ± 11.552 ↓	126.026 ± 22.131 ↓	200.652 ± 22.959 ↑	<b>58.327 ± 6.036 ↓</b>
mo-humanoid-v5	412.410 ± 107.472	293.540 ± 89.884 ↓	<b>215.982 ± 118.950 ↓</b>	290.982 ± 105.797 ↓	430.007 ± 131.208 ↔
mo-halfcheetah-v5	335.006 ± 22.120	642.419 ± 5.469 ↑	413.204 ± 9.344 ↑	447.322 ± 5.443 ↑	<b>181.705 ± 4.343 ↓</b>
mo-swimmer-v5	<b>8.076 ± 0.804</b>	36.294 ± 1.906 ↑	40.778 ± 1.464 ↑	53.648 ± 1.212 ↑	42.600 ± 0.631 ↑
<b>GD+</b>					
mo-reacher-v5	3.563 ± 1.055	2.883 ± 0.845 ↓	<b>0.240 ± 0.058 ↓</b>	0.383 ± 0.078 ↓	1.101 ± 0.427 ↓
mo-hopper-v5	64.070 ± 23.376	19.651 ± 1.380 ↓	76.553 ± 0.085 ↑	70.253 ± 40.484 ↑	<b>9.352 ± 0.856 ↓</b>
mo-walker2d-v5	<b>47.229 ± 51.499</b>	167.460 ± 92.977 ↑	90.501 ± 70.649 ↑	748.608 ± 56.411 ↑	163.153 ± 28.325 ↑
mo-ant-v5	210.874 ± 39.481	143.679 ± 29.985 ↓	61.156 ± 18.003 ↓	<b>51.253 ± 19.996 ↓</b>	109.000 ± 8.712 ↓
mo-humanoid-v5	412.410 ± 107.472	354.987 ± 74.833 ↓	<b>292.770 ± 81.063 ↓</b>	390.772 ± 56.189 ↔	430.007 ± 131.208 ↔
mo-halfcheetah-v5	111.760 ± 4.631	721.649 ± 6.577 ↑	442.255 ± 26.083 ↑	<b>41.323 ± 3.482 ↓</b>	140.415 ± 19.895 ↑
mo-swimmer-v5	4.204 ± 0.580	38.051 ± 2.097 ↑	2.593 ± 0.437 ↓	<b>1.979 ± 0.552 ↓</b>	7.139 ± 0.624 ↑

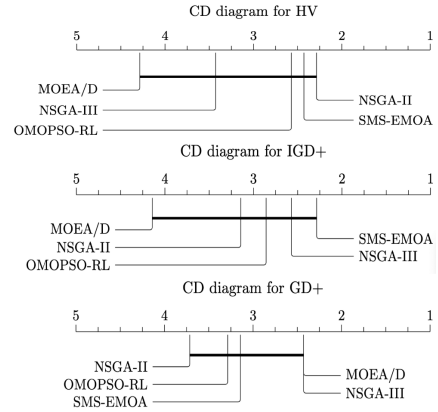
vironments, but achieved significantly better results in mo-walker2d-v5 and mo-halfcheetah-v5. These findings suggest that OMOPSO-RL generally promotes diversity (high HV) but struggles with convergence.

In terms of HV, OMOPSO-RL achieved an average rank of 2.571, behind NSGA-II (2.286) and SMS-EMOA (2.429). The Friedman test resulted in a statistic of  $\chi^2 = 8.0000$  with a  $p = 0.0916$  and a critical difference (CD) of 2.3056, indicating that there are no statistically significant differences among the algorithms. For the IGD+, OMOPSO-RL ranked third with an average rank of 2.857, behind SMS-EMOA (2.286) and NSGA-III (2.571), which suggests some limitations in convergence. Similarly to the HV results, we found no significant differences for IGD+, with  $\chi^2 = 5.7143$ ,  $p = 0.2215$ , and  $CD = 2.3056$ . OMOPSO-RL performed the weakest on GD+, with an average rank of 3.286, while NSGA-III and MOEA/D shared the best rank of 2.429. However, the Friedman test again indicated non-significance, with  $\chi^2 = 3.5429$ ,  $p = 0.4714$ , and  $CD = 2.3056$ . Figure 3 illustrates the critical difference diagrams for all three indicators. When aggregating the ranks across metrics, SMS-EMOA showed the best overall performance with a rank of 2.619, followed by NSGA-III (2.810), OMOPSO-RL (2.905), NSGA-II (3.048), and MOEA/D (3.619), with  $CD = 1.3311$ .

In summary, OMOPSO-RL excels at generating diverse policies, as reflected by its high HV values. However, its relatively high IGD+ and GD+ indicate limitations in convergence, particularly in problems involving more complex and structured locomotion. In mo-walker2d-v5 and mo-swimmer-v5, it achieves competitive results across all three metrics, demonstrating effectiveness in specific scenarios. These findings suggest that enhancing convergence mechanisms could improve generalization and robustness across a wider range of MORL problems.

## 6 Conclusions and Future Work

This work presented an adaptation of the OMOPSO algorithm for MORL problems, resulting in OMOPSO-RL. The experimental results support the following conclusions: (i) OMOPSO-RL is a viable MORL approach, extending particle swarm search to learn multi-objective policies, as shown in mo-walker2d-v5 and mo-swimmer-v5; (ii) it shows high diversity, consistently ranking third in hyper-



**Figure 3.** Critical difference diagrams for HV, IGD+, and GD+ indicators.

volume; however, it struggles with convergence, ranking third in IGD+ and fourth in GD+; (iii) its environment-dependent behavior excels in structured locomotion problems, while it underperforms in more complex environments such as mo-ant-v5 or mo-humanoid-v5, where other algorithms perform better; (iv) OMOPSO-RL is competitive, but not dominant. While it occasionally outperformed other algorithms, it did not show statistically significant superiority and obtained a moderate overall ranking. However, it provides a solid foundation for MORL.

### 6.1 Future Work

Based on these findings, we propose several improvements for OMOPSO-RL: (i) further analysis of the results to identify its strengths and weaknesses; (ii) an improved external archive for a good distribution of the policies both in decision and objective space; (iii) integration of local search strategy to enhance convergence; (iv) conduct a sensitivity analysis of hyperparameters to better understand their impact on performance across different environments; and (v) a comparative study against other MOEAs designed for MORL to compare the performance of OMOPSO-RL. These enhancements aim to strengthen the convergence and robustness of OMOPSO-RL in various MORL problems.

## References

- [1] O. S. Ajani, D. F. Ivan, D. Darlan, P. Suganthan, K. Gao, and R. Mallipeddi. Deep reinforcement learning as multiobjective optimization benchmarks: Problem formulation and performance assessment. *Swarm and Evolutionary Computation*, 90:101692, 2024.
- [2] D. Angus and C. Woodward. Multiple objective ant colony optimisation. *Swarm intelligence*, 3:69–85, 2009.
- [3] N. Beume, B. Naujoks, and M. Emmerich. Sms-emoa: Multiobjective selection based on dominated hypervolume. *European journal of operational research*, 181(3):1653–1669, 2007.
- [4] S. M. Borrel Miller and C. I. Hernández Castellanos. A R2 based multi-objective reinforcement learning algorithm. In *International Conference on Learning and Intelligent Optimization*, pages 285–289. Springer, 2024.
- [5] A. Callaghan, K. Mason, and P. Mannion. Extending evolution-guided policy gradient learning into the multi-objective domain. *Neurocomputing*, 636:129991, 2025.
- [6] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga. Handling multiple objectives with particle swarm optimization. *IEEE Transactions on evolutionary computation*, 8(3):256–279, 2004.
- [7] C. C. Coello and M. S. Lechuga. MOPSO: A proposal for multiple objective particle swarm optimization. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, volume 2, pages 1051–1056. IEEE, 2002.
- [8] K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. *IEEE transactions on evolutionary computation*, 18(4):577–601, 2013.
- [9] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.
- [10] C. F. Hayes, R. Rădulescu, E. Bargiacchi, J. Källström, M. Macfarlane, M. Reymond, T. Verstraeten, L. M. Zintgraf, R. Dazeley, F. Heintz, et al. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36(1):26, 2022.
- [11] C. I. Hernandez Castellanos, S. Ober-Blöbaum, and S. Peitz. Explicit multiobjective model predictive control for nonlinear systems under uncertainty. *International Journal of Robust and Nonlinear Control*, 30(17):7593–7618, 2020.
- [12] T. Hu and B. Luo. Pa2d-morl: Pareto ascent directional decomposition based multi-objective reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 12547–12555, 2024.
- [13] H. Ishibuchi, N. Tsukamoto, Y. Sakane, and Y. Nojima. Indicator-based evolutionary algorithm with hypervolume approximation by achievement scalarizing functions. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 527–534, 2010.
- [14] H. Ishibuchi, H. Masuda, Y. Tanigaki, and Y. Nojima. Modified distance calculation in generational distance and inverted generational distance. In *International conference on evolutionary multi-criterion optimization*, pages 110–125. Springer, 2015.
- [15] J. Karlsson. *Learning to solve multiple goals*. University of Rochester, 1997.
- [16] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE, 1995.
- [17] S. Khadka and K. Tumer. Evolution-guided policy gradient in reinforcement learning. *Advances in Neural Information Processing Systems*, 31, 2018.
- [18] R. Liu, Y. Pan, L. Xu, L. Song, P. You, Y. Chen, and J. Bian. C-morl: Multi-objective reinforcement learning through efficient discovery of pareto front. *arXiv preprint arXiv:2410.02236*, 2024.
- [19] A. J. Nebro, M. López-Ibáñez, J. García-Nieto, and C. A. Coello Coello. On the automatic design of multi-objective particle swarm optimizers: experimentation and analysis. *Swarm intelligence*, 18(2):105–139, 2024.
- [20] M. Reymond, C. F. Hayes, D. Steckelmacher, D. M. Roijers, and A. Nowé. Actor-critic multi-objective reinforcement learning for non-linear utility functions. *Autonomous Agents and Multi-Agent Systems*, 37(2):23, 2023.
- [21] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48:67–113, 2013.
- [22] R. Santana Hermida and C. I. Hernández Castellanos. Benchmarking moeas for solving continuous multi-objective rl problems. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '25 Companion, page 415–418. Association for Computing Machinery, 2025.
- [23] O. Schuetze, C. Hernandez, E.-G. Talbi, J.-Q. Sun, Y. Naranjani, and F.-R. Xiong. Archivers for the representation of the set of approximate solutions for mops. *Journal of Heuristics*, 25(1):71–105, 2019.
- [24] O. Schütze and C. Hernández. *Archiving strategies for evolutionary multi-objective optimization algorithms*. Springer, 2021.
- [25] M. R. Sierra and C. A. Coello Coello. Improving pso-based multi-objective optimization using crowding, mutation and epsilon-dominance. In *International conference on evolutionary multi-criterion optimization*, pages 505–519. Springer, 2005.
- [26] E.-G. Talbi. *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009.
- [27] K. Van Moffaert. *Multi-criteria reinforcement learning for sequential decision making problems*. PhD thesis, Ph. D. thesis, Vrije Universiteit Brussel, 2016.
- [28] K. Van Moffaert and A. Nowé. Multi-objective reinforcement learning using sets of pareto dominating policies. *The Journal of Machine Learning Research*, 15(1):3483–3512, 2014.
- [29] J. Xu, Y. Tian, P. Ma, D. Rus, S. Sueda, and W. Matusik. Prediction-guided multi-objective reinforcement learning for continuous robot control. In *International conference on machine learning*, pages 10607–10616. PMLR, 2020.
- [30] M. Xu, P. Ju, J. Liu, and H. Yang. Psmgd: Periodic stochastic multi-gradient descent for fast multi-objective optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 21770–21778, 2025.
- [31] F. Xue, Y. Chen, T. Dong, P. Wang, and W. Fan. Moea/d with adaptive weight vector adjustment and parameter selection based on q-learning. *Applied Intelligence*, 55(6):399, 2025.
- [32] Q. Zhang and H. Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731, 2007.
- [33] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE transactions on Evolutionary Computation*, 3(4):257–271, 1999.
- [34] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. *TIK report*, 103, 2001.