

Hindsight Preference Replay Improves Preference-Conditioned Multi-Objective Reinforcement Learning

Jonaid Shianifar^{1[0000–0003–0477–0056]}, Michael Schukat^{1[0000–0002–6908–6100]},
and Karl Mason^{1[0000–0002–8966–9100]}

School of Computer Science, University of Galway, H91 FYH2, Galway, Ireland
 {J.Shianifar1,Michael.Schukat,Karl.Mason}@universityofgalway.ie

Abstract. Multi-objective reinforcement learning (MORL) enables agents to optimize vector-valued rewards while respecting user preferences. CAPQL, a preference-conditioned actor-critic method, achieves this by conditioning on weight vectors \mathbf{w} and restricts data usage to the specific preferences under which it was collected, leaving off-policy data from other preferences unused. We introduce Hindsight Preference Replay (HPR), a simple and general replay augmentation strategy that retroactively relabels stored transitions with alternative preferences. This densifies supervision across the preference simplex without altering the CAPQL architecture or loss functions. Evaluated on six MO-Gymnasium locomotion tasks at a fixed 300,000-step budget using expected utility (EUM), hypervolume (HV), and sparsity, HPR-CAPQL improves HV in five of six environments and EUM in four of six. On `mo-humanoid-v5`, for instance, EUM rises from 323 ± 125 to 1613 ± 464 and HV from 0.52M to 9.63M, with strong statistical support. `mo-halfcheetah-v5` remains a challenging exception where CAPQL attains higher HV at comparable EUM. We report final summaries and Pareto-front visualizations across all tasks.¹

Keywords: Multi-objective reinforcement learning (MORL) · Hindsight Preference Replay (HPR) · Preference-Conditioned Actor-Critic · CAPQL · Replay augmentation · Robotic Control

1 Introduction

Many real-world control tasks, such as robotic locomotion or autonomous driving, involve optimizing multiple conflicting objectives, for example, energy efficiency versus speed, or safety versus exploration. Multi-objective reinforcement learning (MORL) formalizes this by optimizing vector-valued rewards $\mathbf{r}_t \in \mathbb{R}^m$ [12, 15]. However, selecting a single solution requires expressing user preferences over these objectives [6].

¹ This paper has been accepted for presentation at the 33rd Irish Conference on Artificial Intelligence and Cognitive Science (AICS 2025).

One approach to MORL is scalarization, where a weight vector $\mathbf{w} \in \Delta^m$ is used to linearly combine the objectives: $u_{\mathbf{w}}(\mathbf{r}) = \mathbf{w}^\top \mathbf{r}$. Preference-conditioned policies exploit this idea by taking \mathbf{w} as input, enabling a single policy to serve a range of trade-offs depending on user preferences. This setup supports efficient deployment in dynamic or interactive systems where preferences may change over time [7, 8].

CAPQL [7] is a recent preference-conditioned actor-critic method that extends Soft Actor-Critic (SAC) [5] to the MORL setting. CAPQL augments the critic loss with a concavity-based regularizer to improve coverage and training stability. However, like most off-policy algorithms, it suffers from a key inefficiency: data collected under one preference \mathbf{w} is only used to train for that same \mathbf{w} [7].

This results in low sample efficiency, as transitions informative for neighboring preferences are discarded. This inefficiency is especially costly in high-dimensional control tasks, where sample collection is expensive. Inspired by Hindsight Experience Replay (HER) that relabels goals to overcome sparse rewards [1, 13], we propose *Hindsight Preference Replay* (HPR), which retroactively relabels preference vectors, densifying supervision without architectural or loss changes [3]. Empirically, HPR-CAPQL improves hypervolume in five of six benchmarks, with especially large gains on `mo-humanoid-v5`, while `mo-halfcheetah-v5` remains a challenging case where CAPQL covers the front more broadly.

Our contributions are as follows:

- We present HPR, a general-purpose hindsight relabeling mechanism for preference-conditioned MORL.
- We instantiate HPR within CAPQL, creating HPR-CAPQL, with minimal implementation overhead.
- We evaluate HPR-CAPQL on six continuous control tasks from the MO-Gymnasium benchmark suite, showing performance gains in several environments.

2 Background and Literature Review

Modern control problems, such as robotic manipulation and locomotion, inherently require optimizing multiple conflicting objectives, for example, achieving both high efficiency and safety. The field of multi-objective reinforcement learning (MORL) formalizes this by extending classical reinforcement learning to vector-valued rewards $\mathbf{r}_t \in \mathbb{R}^m$, enabling agents to discover policies that provide favorable trade-offs across the Pareto front [4, 8, 12, 16].

A common approach is *scalarization*, where a preference or weight vector $\mathbf{w} \in \Delta^m$ combines objectives into a single utility $u_{\mathbf{w}}(\mathbf{r}) = \mathbf{w}^\top \mathbf{r}$, thereby parameterizing the trade-off space [8]. Recent progress on adaptive scalarization reports improved stability and Pareto coverage in MORL, particularly in continuous-control settings [10]. In parallel, preference-conditioned policies condition both policy and value networks on \mathbf{w} to adapt to user preferences and efficiently cover the Pareto front [4, 7]. From a robotics perspective, deep RL for adaptive robotic arm control underscores the need for sample-efficient replay and safe, budgeted

interaction [9]. Our approach is complementary to these lines: HPR augments replay to reuse trajectories across preferences without modifying the underlying losses or architectures.

Among these methods, *CAPQL* (Concavity-Aware Preference-based Q-Learning) has emerged as a strong baseline in preference-conditioned MORL. CAPQL extends the Soft Actor-Critic (SAC) framework by conditioning both policy and critics on \mathbf{w} , enabling a single policy to generalize across multiple trade-offs. A key innovation in CAPQL is a concavity-based regularizer in the critic loss, which encourages stable training and broad coverage of the Pareto front by penalizing deviations from concave utility approximations. This design improves sample efficiency compared to earlier preference-conditioned approaches; however, CAPQL still updates policies only for the specific preferences under which data were collected, leaving off-policy experience from other preferences unused [7].

A milestone in single-objective, goal-conditioned RL was the introduction of *Hindsight Experience Replay* (HER), which relabels failed transitions with alternative goals to improve learning under sparse rewards [1]. HER established that retroactive relabeling can transform unsuccessful experiences into effective learning signals. Inspired by this success, researchers have explored extending hindsight mechanisms to other conditioning variables such as preferences.

The recently proposed *Hindsight Preference Replay* (HPR) adapts HER’s philosophy to the multi-objective setting by retroactively relabeling the preference vector \mathbf{w} in stored transitions. This enables richer and denser supervision across the preference simplex without any architectural or loss modification, offering large improvements in sample efficiency and Pareto coverage when combined with preference-conditioned methods like CAPQL [7].

Earlier work on multi-objective HER primarily focused on relabeling goals and generating curriculum for complex robotic tasks using evolutionary algorithms [1, 4], but did not address the inefficiency of preference-conditioned experience reuse. Recent strategies in multi-objective learning, such as Ensemble Multi-Objective RL (EMORL) [4], curriculum generation via evolutionary multi-objective optimization [1, 2, 16], and meta-learning for adaptation, have pushed the boundaries of scalable multi-objective RL; however, they do not apply retroactive preference relabeling.

The proposed HPR therefore offers a novel and practical mechanism to improve sample efficiency and generalization in preference-conditioned MORL, building on the proven utility of hindsight-based replay and extending it in previously unexplored directions.

3 Methodology: Hindsight Preference Replay (HPR)

In preference-conditioned MORL, transitions collected under a specific preference vector \mathbf{w} are typically used exclusively to update the policy for that same \mathbf{w} . However, these transitions may also be informative for other preferences,

particularly those nearby in the simplex. This leads to inefficiencies in data usage, limiting gradient diversity and generalization.

To address this, we propose *Hindsight Preference Replay* (HPR), a replay augmentation strategy that retroactively relabels transitions with alternative preference vectors. The key idea is analogous to HER [1], but adapted to preference relabeling rather than goal relabeling.

The pseudocode in Algorithm 1 summarizes the full HPR-CAPQL procedure integrated with the CAPQL-style actor-critic learning. It highlights how transitions are collected and relabeled with multiple alternative preferences, stored in the replay buffer, and how training updates interleave with environment interaction and evaluation. This overview guides the detailed design of the relabeling strategies and integration steps discussed next.

Algorithm 1 HPR-CAPQL: Hindsight Preference Replay for Preference-Conditioned MORL

Require: Environment Env ; preference-conditioned agent (policy/critics conditioned on w); replay buffer \mathcal{B} ; per-step relabels K ; Dirichlet concentration κ ; relabeled minibatch fraction ρ ; (optional) acceptance test.

- 1: Initialize agent and targets; $\mathcal{B} \leftarrow \emptyset$
- 2: **for** episodes until budget exhausted **do**
- 3: Sample behavior preference $w \sim p(w)$ on the simplex; reset Env ; set episode return $G \leftarrow \mathbf{0}$
- 4: **while** not done **do**
- 5: Act $a \sim \pi(\cdot | s, w)$, step env $\rightarrow (r, s', d)$; push (s, a, r, s', d, w) to \mathcal{B}
- 6: $G \leftarrow G + \gamma^t r$ \triangleright discounted vector return for relabeling/eval
- 7: **for** $k = 1..K$ **do** \triangleright Neighborhood relabeling
- 8: $\tilde{w} \sim \text{Dir}(\kappa w)$; **if** accepted **then** push $(s, a, r, s', d, \tilde{w})$ to B
- 9: **end for**
- 10: $s \leftarrow s'$
- 11: **Training step(s):** sample minibatches from B with relabeled fraction ρ ; update CAPQL with scalarized rewards $r_w = w^\top r$
- 12: **end while** \triangleright Return-aligned relabeling (episode end)
- 13: $\hat{w} \leftarrow \Pi_\Delta(\text{softplus}(G))$; for each episode transition push $(s, a, r, s', d, \hat{w})$ if accepted
- 14: **end for**
- 15: **Evaluation:** on a fixed weight grid, log EUM, HV, sparsity, and per-objective means

Given a stored transition tuple $(s, a, \mathbf{r}, s', \mathbf{w})$, HPR replaces the original preference vector \mathbf{w} with a new preference vector $\tilde{\mathbf{w}}$ sampled by return-aligned or neighborhood strategies [3, 14]. The reward vector \mathbf{r} is left unchanged, enabling recomputation of scalarized returns $u_{\tilde{\mathbf{w}}}(\mathbf{r}) = \tilde{\mathbf{w}}^\top \mathbf{r}$ and allowing a single transition to serve multiple scalarization preferences. This approach generalizes the HER

principle [1, 13] into the preference space, enabling richer supervision without modifying CAPQL’s architecture [7].

3.1 Relabeling Strategies

We implement two relabeling strategies in HPR:

Return-Aligned Relabeling. For each trajectory or transition, we compute a discounted cumulative return vector \mathbf{G} . We then define the relabeled preference $\tilde{\mathbf{w}}$ as:

$$\tilde{\mathbf{w}} \propto \text{softplus}(\mathbf{G}) \quad (1)$$

followed by projection onto the simplex. Optionally, a convex combination with the original \mathbf{w} can be applied to control deviation. This heuristic aligns the relabeling with the outcomes the trajectory has already achieved.

Neighborhood Sampling. We draw K new preference vectors $\tilde{\mathbf{w}} \sim \text{Dir}(\kappa\mathbf{w})$ from a Dirichlet distribution centered around the original \mathbf{w} , where κ is a concentration parameter. Larger κ values result in narrower neighborhoods, encouraging local generalization.

3.2 Integration with CAPQL

We integrate HPR into CAPQL by modifying the replay buffer insertion logic and minibatch sampling:

- During data collection, each transition is stored as $(s, a, \mathbf{r}, s', \mathbf{w})$.
- On insertion (or periodically), up to K relabeled versions are added using one of the above strategies, with preferences $\tilde{\mathbf{w}}$.
- During training, minibatches are sampled uniformly, mixing original and relabeled transitions.

Importantly, all CAPQL loss functions and network structures remain unchanged. HPR only alters the diversity of training data in the preference space.

3.3 Practical Settings and Filters

We explore relabel counts $K \in \{0, 1, 2, 4\}$, Dirichlet concentration parameters $\kappa \in \{10, 20, 50\}$, and relabel batch ratios $\rho \in [0.3, 0.7]$.

In settings with strongly orthogonal objectives, naive relabeling can harm learning. In such cases, we apply acceptance filters based on alignment between $\tilde{\mathbf{w}}$ and G , e.g., $\cos(\tilde{\mathbf{w}}, G) \geq \tau$ or $\tilde{\mathbf{w}}^\top G \geq \mathbf{w}^\top G - \varepsilon$, with $\varepsilon \in \{0.0, 0.1\}$ and $\tau \in \{0.7, 0.8\}$. This mitigates regressions in environments where broad relabels destabilize HV (e.g., `mo-halfcheetah-v5`), while preserving the gains observed elsewhere.

4 Experimental Setup

4.1 Environments

We evaluate HPR-CAPQL across six continuous control tasks from the MO-Gymnasium benchmark suite [4], an extension of Gymnasium [11] designed for multi-objective reinforcement learning. All environments feature continuous observations and continuous action spaces, returning 2D reward vectors that correspond to conflicting objectives.

The environments are described as follows:

- **mo-hopper-2obj-v5**: A 2D hopping robot optimizing for forward progress and jumping height.
- **mo-walker2d-v5**: A bipedal walker balancing locomotion efficiency and energy cost.
- **mo-halfcheetah-v5**: A planar cheetah optimizing for speed versus energy efficiency.
- **mo-humanoid-v5**: A high-dimensional humanoid robot focusing on fast movement and minimal effort.
- **mo-swimmer-v5**: A simple two-link swimmer navigating through fluid.
- **mo-ant-2obj-v5**: A quadruped ant robot optimizing its movement in both X and Y directions.

Key characteristics of these environments, including observation dimensions, action dimensions, and reward components, are summarized in Table 1.

Table 1. Observation and Action Dimensions and Reward Components of MO-Gymnasium Environments

Environment	Observation Dim.	Action Dim.	Reward Components
mo-hopper-2obj-v5	11	3	Forward velocity, Jump height
mo-walker2d-v5	17	6	Forward velocity, Control cost
mo-halfcheetah-v5	17	6	Forward velocity, Control cost
mo-humanoid-v5	348	17	Forward velocity, Control cost
mo-swimmer-v5	8	2	Forward velocity, Control cost
mo-ant-2obj-v5	105	8	X-velocity, Y-velocity

All environments utilize MuJoCo physics and provide dense, continuous rewards.

4.2 Training Protocol

We compare standard CAPQL against HPR-CAPQL using the same codebase for both methods. Each algorithm is trained for 300,000 environment steps per task, with evaluation every 10,000 steps on a fixed preference grid over the 2D

simplex. We report the mean and standard deviation over five seeds. Metrics include Expected Utility (EUM), Hypervolume (HV), and the sparsity of the nondominated archive; we also plot the final Pareto fronts. HV is computed with a fixed per-environment reference point $\mathbf{r}_{\text{ref}} = [-100, -100]$ shared by all methods and seeds, and we report HV in millions ($\times 10^6$) for readability.

4.3 Hardware Configuration

All experiments were conducted on a high-performance computing server equipped with an AMD Ryzen Threadripper 7960X 24-core processor, two NVIDIA RTX A6000 GPUs, and 128 GB of system RAM. The system ran Ubuntu 22.04.5 LTS. The software stack included Python 3.9.23, PyTorch 2.8.0, Gymnasium 1.1.1, and MO-Gymnasium 1.3.1. This configuration provided ample computational power for efficient training and evaluation of preference-conditioned policies across all benchmark tasks.

5 Results and Discussion

We evaluate CAPQL and HPR-CAPQL for 300,000 environment steps on six MO-Gymnasium tasks. We report Expected Utility (EUM; \uparrow), hypervolume (HV; \uparrow ; Table values in millions; For each environment, we use a fixed $r_{\text{ref}} = [-100, -100]$), and sparsity (archive spacing; \downarrow). Final summaries with Welch tests for HV appear in Table 2. Figures 1 to 3 show EUM/HV/sparsity learning curves, and Figure 4 visualizes final Pareto fronts from the union of non-dominated points across seeds.

5.1 Final Performance Summary

Across tasks, hindsight preference relabeling primarily increases Pareto coverage. HPR-CAPQL attains higher HV in most environments and improves EUM in a majority as well; the aggregate row of Table 2 reflects this pattern. Several HV differences are statistically reliable, notably on `mo-humanoid-v5`, `mo-walker2d-v5`, and `mo-ant-2obj-v5`. Two tasks deviate from this trend. `mo-halfcheetah-v5` favors CAPQL on HV while EUM remains comparable, indicating that additional relabels do not widen the front in that domain. `mo-swimmer-v5` is effectively a draw: both algorithms saturate quickly and the small HV differences are not significant.

5.2 Learning Behavior Over Time

Figures 1 to 3 plot EUM, HV, and sparsity over the 300,000-step horizon and show how the final differences in Table 2 emerge during training. On `mo-hopper-2obj-v5` and `mo-ant-2obj-v5`, HPR-CAPQL accelerates early, establishing EUM and HV margins that persist through the end of training. On

Table 2. Final performance after 300,000 steps (mean \pm std over seeds). HV shown in millions ($\times 10^6$) using the fixed r_{ref} . Significance markers: * for $p < 0.05$, ** for $p < 0.01$, and *** for $p < 0.001$.

Environment	Algorithm	EUM \uparrow	Sparsity \downarrow	HV \uparrow	HV p -value
mo-hopper-2obj	CAPQL	2370.5 \pm 634.7	84.6\pm76.8	4.7534 \pm 2.39	0.0749
	HPR-CAPQL	2942.7\pm143.8	295.9 \pm 81.5	7.2026\pm1.03	
mo-walker2d	CAPQL	1056.4 \pm 273.3	2294.8 \pm 230.0	1.3276 \pm 0.52	0.0079 **
	HPR-CAPQL	1320.7\pm207.6	513.5\pm361.5	2.9855\pm0.91	
mo-halfcheetah	CAPQL	1616.7\pm202.2	416.3\pm116.6	8.7208\pm1.48	0.0350 *
	HPR-CAPQL	1603.0 \pm 319.7	535.9 \pm 183.1	5.9927 \pm 1.97	
mo-humanoid	CAPQL	323.1 \pm 125.4	120.0\pm44.3	0.5173 \pm 0.27	0.0009 ***
	HPR-CAPQL	1613.3\pm464.3	486.5 \pm 88.1	9.6332\pm2.94	
mo-ant-2obj	CAPQL	1823.3 \pm 191.2	265.9\pm58.2	3.2311 \pm 0.31	0.0060 **
	HPR-CAPQL	2060.7\pm157.1	653.6 \pm 356.2	4.5428\pm0.68	
mo-swimmer	CAPQL	22.2\pm1.7	10.3 \pm 7.1	0.0054 \pm 0.0030	0.2926
	HPR-CAPQL	22.1 \pm 1.9	5.9\pm1.3	0.0104\pm0.0090	
HPR-CAPQL better (of 6)		4	2	5	—

mo-humanoid-v5, gains arrive later but become pronounced: once the policy stabilizes, relabeled preferences unlock new parts of the frontier and HV separates sharply. mo-walker2d-v5 shows the same qualitative pattern with narrower uncertainty. In contrast, mo-halfcheetah-v5 exhibits a plateau for HPR-CAPQL while CAPQL continues to expand HV, and mo-swimmer-v5 reaches a quick equilibrium for both methods. Variance bands widen in the tasks where coverage expands the most, reflecting a broader exploration of the preference simplex rather than instability of the learner.

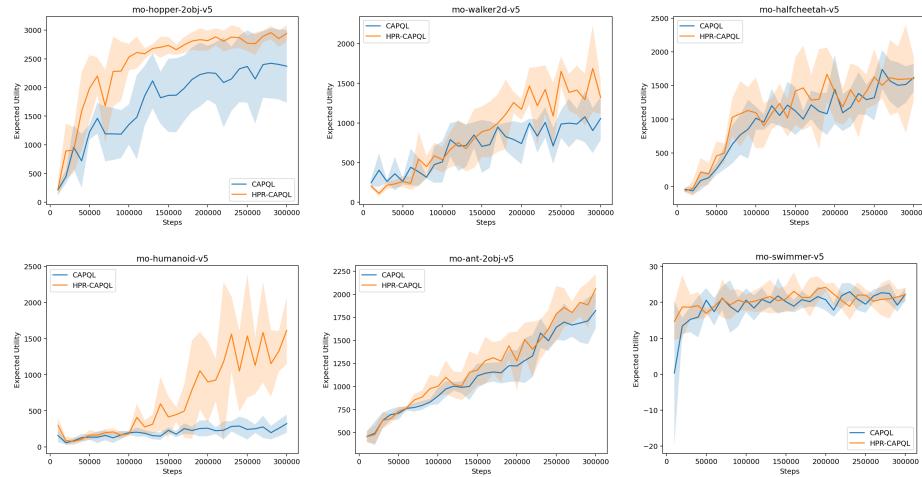


Fig. 1. EUM learning curves up to 300,000 steps (mean \pm std over five seeds).

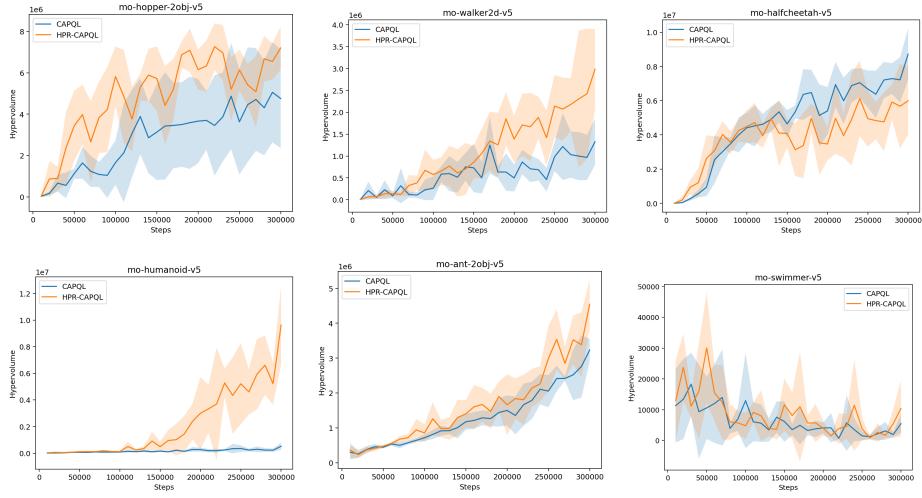


Fig. 2. Hypervolume (HV) learning curves up to 300,000 steps (mean \pm std over five seeds).

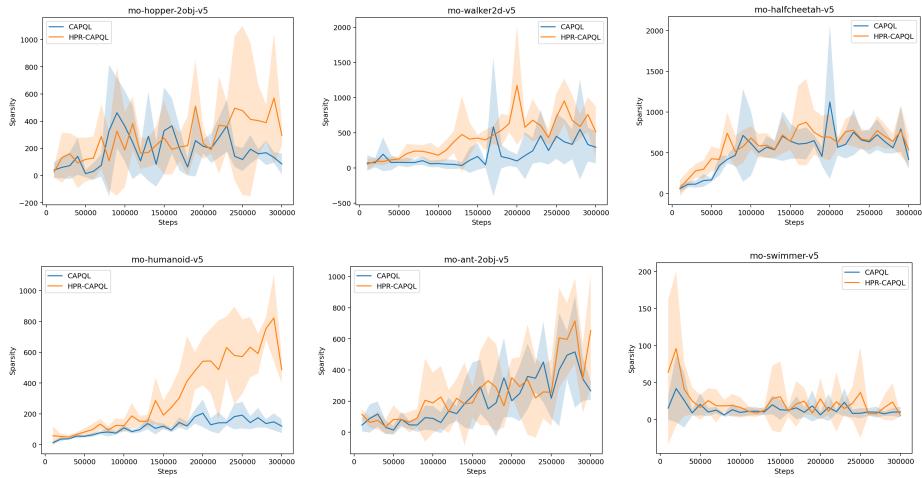


Fig. 3. Sparsity (lower is better) up to 300,000 steps (mean \pm std over five seeds).

5.3 Pareto-Front Comparisons

Figure 4 shows the final non-dominated Pareto fronts (union over seeds at 300,000 steps), revealing where each method extends coverage and how these geometric

differences drive the EUM and HV gaps. On `mo-humanoid-v5`, `mo-walker2d-v5`, `mo-hopper-2obj-v5`, and `mo-ant-2obj-v5`, HPR-CAPQL pushes outward along large portions of the trade-off surface, occupying regions that CAPQL rarely reaches. This outward shift explains both the hypervolume gains and the higher average scalarized utility across preferences. In `mo-halfcheetah-v5`, CAPQL extends further toward an extreme of one objective, which is sufficient to dominate HV despite similar EUM. In `mo-swimmer-v5`, the fronts largely overlap, aligning with the inconclusive learning curves.

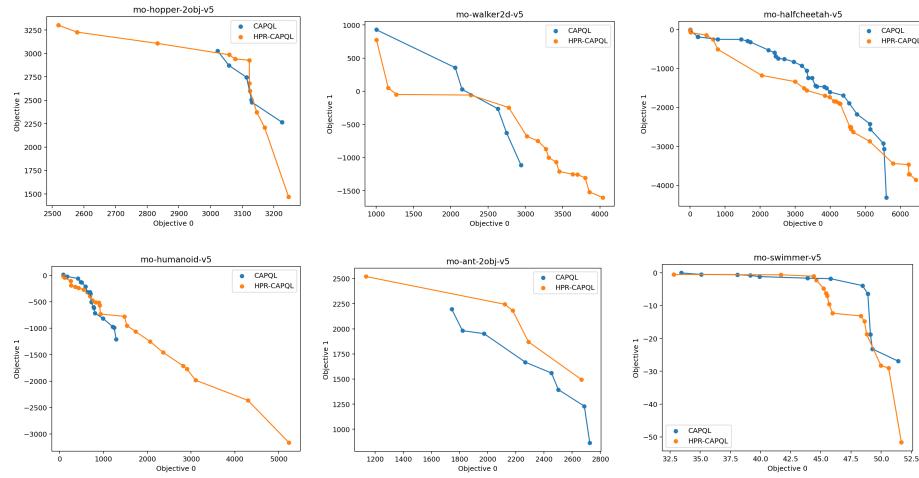


Fig. 4. Final Pareto fronts (non-dominated only) after 300,000 steps across all environments. Each panel shows the union of non-dominated solutions across five seeds; legend: CAPQL vs. HPR-CAPQL

5.4 Coverage–Density Trade-off

Where HPR-CAPQL expands the frontier, sparsity tends to increase: the archive becomes more widely spaced while reaching new territory. This is a natural consequence of a fixed training budget; growing the frontier and densifying it are different objectives. A short densification phase near the end of training, reducing the relabel ratio or temporarily disabling HPR, tightens spacing without relinquishing the newly discovered regions. When dense coverage is paramount, sampling hindsight preferences closer to the uniform simplex or adding small jitter around current preferences further improves spacing.

5.5 Environment-Specific Remarks

On `mo-humanoid-v5`, HPR-CAPQL provides the clearest improvement: the frontier enlarges substantially and both EUM and HV rise throughout the latter half

of training. `mo-walker2d-v5` exhibits steady gains with moderate variance; the final fronts extend outward with only modest loss of density. `mo-hopper-2obj-v5` shows strong early advantages that persist, with the HPR front lying above the baseline over most of the trade-off range. `mo-ant-2obj-v5` mirrors this behavior but illustrates the coverage–density trade-off most clearly. `mo-halfcheetah-v5` favors CAPQL on HV; here, aggressive preference relabeling appears mismatched with the task’s reward geometry. `mo-swimmer-v5` is simple enough that both methods rapidly converge to similar fronts.

6 Conclusion

We presented Hindsight Preference Replay (HPR), a simple replay augmentation that retroactively relabels stored transitions with alternative preference vectors and slots into CAPQL without architectural or loss changes. Under a fixed 300,000-step budget across six MO-Gymnasium tasks, HPR-CAPQL consistently broadens the learned Pareto fronts and often increases preference-averaged utility. In aggregate, hypervolume (HV) is higher in five of six environments and Expected Utility (EUM) in four of six, while CAPQL retains an advantage on `mo-halfcheetah-v5`.

A few highlights illustrate the magnitude and reliability of the effects. On `mo-humanoid-v5`, HPR-CAPQL delivers large gains (EUM 1613 ± 464 vs. 323 ± 125 ; HV $9.63M$ vs. $0.52M$) with strong statistical support ($p = 0.0009$). `mo-walker2d-v5` shows a similar pattern (HV $2.99M$ vs. $1.33M$; $p = 0.0079$; EUM 1321 ± 208 vs. 1056 ± 273), and `mo-ant-2obj-v5` also favors HPR-CAPQL on both metrics (HV $4.54M$ vs. $3.23M$; $p = 0.0060$; EUM 2061 ± 157 vs. 1823 ± 191). Two domains temper the story: `mo-hopper-2obj-v5` trends toward higher HV for HPR-CAPQL but is not significant at 5% ($p = 0.0749$), and `mo-halfcheetah-v5` significantly favors CAPQL on HV (HV $8.72M$ vs. $5.99M$; $p = 0.035$) with similar EUM across methods. `mo-swimmer-v5` is effectively a draw (small, noisy HV differences; $p = 0.2926$).

A consistent pattern is a coverage–density trade-off. When HPR broadens the Pareto frontier, the nondominated archive often becomes more widely spaced, especially where HV gains are largest. This is expected under a fixed budget, since expanding the frontier and densifying it are different goals. A short densification phase near the end, reducing the relabel ratio or briefly disabling HPR, tightens spacing without losing newly reached regions. For tasks such as `mo-halfcheetah-v5`, conservative relabeling and return-alignment filters further curb variance and harmful updates.

These results support HPR as a strong default for preference-conditioned MORL. It reuses off-policy data across preferences and yields broader Pareto coverage and higher preference-averaged performance with minimal engineering. Future work includes many-objective settings, adaptive preference distributions, learned acceptance criteria, and larger seed budgets.

Acknowledgements

This work is supported by the University of Galway College of Science and Engineering Postgraduate Scholarship.

References

1. Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., Zaremba, W.: Hindsight experience replay. *Advances in neural information processing systems* **30** (2017)
2. Auger, A., Bader, J., Brockhoff, D., Zitzler, E.: Theory of the hypervolume indicator: optimal μ -distributions and the choice of the reference point. In: *Proceedings of the tenth ACM SIGEVO workshop on Foundations of genetic algorithms*. pp. 87–102 (2009)
3. Fan, M., Zhou, J., Zhang, Y., Wu, Y., Chen, J., Sartoretti, G.A.: Preference-driven multi-objective combinatorial optimization with conditional computation. arXiv preprint arXiv:2506.08898 (2025)
4. Felten, F., Alegre, L.N., Nowe, A., Bazzan, A., Talbi, E.G., Danoy, G., C da Silva, B.: A toolkit for reliable benchmarking and research in multi-objective reinforcement learning. *Advances in Neural Information Processing Systems* **36**, 23671–23700 (2023)
5. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: *International conference on machine learning*. pp. 1861–1870. Pmlr (2018)
6. Hayes, C.F., Rădulescu, R., Bargiacchi, E., Källström, J., Macfarlane, M., Reymond, M., Verstraeten, T., Zintgraf, L.M., Dazeley, R., Heintz, F., et al.: A practical guide to multi-objective reinforcement learning and planning. arXiv preprint arXiv:2103.09568 (2021)
7. Lu, H., Herman, D., Yu, Y.: Multi-objective reinforcement learning: Convexity, stationarity and pareto optimality. In: *The Eleventh International Conference on Learning Representations* (2023)
8. Ruijters, D.M., Vamplew, P., Whiteson, S., Dazeley, R.: A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research* **48**, 67–113 (2013)
9. Shianifar, J., Schukat, M., Mason, K.: Optimizing deep reinforcement learning for adaptive robotic arm control. In: *International Conference on Practical Applications of Agents and Multi-Agent Systems*. pp. 293–304. Springer (2024)
10. Shianifar, J., Schukat, M., Mason, K.: Adaptive scalarization in multi-objective reinforcement learning for enhanced robotic arm control. *Neurocomputing* p. 132205 (2025)
11. Towers, M., Kwiatkowski, A., Terry, J., Balis, J.U., De Cola, G., Deleu, T., Goulão, M., Kallinteris, A., Krimmel, M., KG, A., et al.: Gymnasium: A standard interface for reinforcement learning environments. arXiv preprint arXiv:2407.17032 (2024)
12. Van Moffaert, K., Nowé, A.: Multi-objective reinforcement learning using sets of pareto dominating policies. *The Journal of Machine Learning Research* **15**(1), 3483–3512 (2014)
13. Wan, T., Xu, N.: Advances in experience replay. arXiv preprint arXiv:1805.05536 (2018)

14. Yang, Y., Zhou, T., Pechenizkiy, M., Fang, M.: Preference controllable reinforcement learning with advanced multi-objective optimization. In: Forty-second International Conference on Machine Learning (ICML) (2024)
15. Zhang, L., Qi, Z., Shi, Y.: Multi-objective reinforcement learning—concept, approaches and applications. *Procedia Computer Science* **221**, 526–532 (2023)
16. Zitzler, E., Brockhoff, D., Thiele, L.: The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration. In: International Conference on Evolutionary Multi-Criterion Optimization. pp. 862–876. Springer (2007)