Combining Automated Optimisation of Hyperparameters and Reward Shape

Julian Dierkes
dierkes@aim.rwth-aachen.de
Chair for AI Methodology
RWTH Aachen University

Emma Cramer
emma.cramer@dsme.rwth-aachen.de
Institute for Data Science in Mechanical Engineering
RWTH Aachen University

Holger H. Hoos
hh@aim.rwth-aachen.de
Chair for AI Methodology
RWTH Aachen University
LIACS, Universiteit Leiden

Sebastian Trimpe
trimpe@dsme.rwth-aachen.de
Institute for Data Science in Mechanical Engineering
RWTH Aachen University

arXiv:2406.18293v2 [cs.LG] 9 Oct 2024

Abstract

There has been significant progress in deep reinforcement learning (RL) in recent years. Nevertheless, finding suitable hyperparameter configurations and reward functions remains challenging even for experts, and performance heavily relies on these design choices. Also, most RL research is conducted on known benchmarks where knowledge about these choices already exists. However, novel practical applications often pose complex tasks for which no prior knowledge about good hyperparameters and reward functions is available, thus necessitating their derivation from scratch. Prior work has examined automatically tuning either hyperparameters or reward functions individually. We demonstrate empirically that an RL algorithm's hyperparameter configurations and reward function are often mutually dependent, meaning neither can be fully optimised without appropriate values for the other. We then propose a methodology for the combined optimisation of hyperparameters and the reward function. Furthermore, we include a variance penalty as an optimisation objective to improve the stability of learned policies. We conducted extensive experiments using Proximal Policy Optimisation and Soft Actor-Critic on four environments. Our results show that combined optimisation significantly improves over baseline performance in half of the environments and achieves competitive performance in the others, with only a minor increase in computational costs. This suggests that combined optimisation should be best practice.

1 Introduction

Deep reinforcement learning (RL) has successfully been applied to various domains, including Silver et al. (2017); Akkaya et al. (2019); Kaufmann et al. (2023); Bi & D'Andrea (2023). Despite successes in these and other challenging applications, configuring RL algorithms remains difficult. This is due to the algorithms typically having several hyperparameter and reward configurations, critically determining learning speed and the general outcome of the training process. For each task, there usually is a final objective one wants to achieve. Defining the RL rewards in terms of this objective is typically insufficient; instead, augmenting the reward with additional intermediate rewards, subgoals, and constraints is necessary for effective training. This augmentation of a reward signal is referred to as reward shaping, and performance and learning speed can crucially depend on it (Ng et al., 1999). Next, RL algorithms require the optimisation of hyperparameters, such as learning rate or discount factor. Effective hyperparameter tuning requires an effective reward signal, and effective

reward shaping depends on good hyperparameter configurations. This circular dependency becomes particularly relevant when applying RL to novel environments beyond commonly used benchmarks, for which neither effective reward shapes nor good hyperparameter settings are available.

In the area of Automatic RL (AutoRL) (Parker-Holder et al., 2022), different data-driven approaches have been developed in recent years to automatically approach hyperparameter optimisation (Parker-Holder et al., 2020; Falkner et al., 2018) and reward shaping (Wang et al., 2022; Zheng et al., 2018). However, these methods approach each problem individually without considering their interdependency. Therefore, they require the availability of high-performing configurations of the non-optimised component. To the best of our knowledge, ours is the first study to thoroughly investigate the effectiveness and broader applicability of jointly optimising hyperparameters and reward shape by using multiple and different environments and systematically evaluating the benefit thus obtained.

We examine the combined optimisation of hyperparameters and reward shape using two state-of-the-art RL algorithms: Proximal Policy Optimisation (PPO) (Schulman et al., 2017) and Soft Actor-Critic (SAC) (Haarnoja et al., 2018). We performed experiments on Gymnasium LunarLander (Towers et al., 2023), Google Brax Ant and Humanoid (Freeman et al., 2021), and Robosuite Wipe (Zhu et al., 2020). The Wipe environment is a robot task representing contact-rich interactions inspired by modern production tasks, which has not been well studied in the literature yet. We compare the combined optimisation results against baselines from the literature and, in particular, to individual optimisation of only hyperparameters and reward shape. We employ the state-of-the-art black-box hyperparameter optimisation algorithm DEHB (Awad et al., 2021) for our experiments, which recently showed to outperform other optimisation methods in RL (Eimer et al., 2023).

Our key contributions can be summarised as follows:

1. We illustrate the advantage of joint optimisation by showing complex dependencies between hyperparameters and reward signals in the LunarLander environment. We use an existing hyperparameter optimisation framework and extend it with additional hyperparameters that control reward shaping. We show that combined optimisation can match the performance of individual optimisation with the same compute budget despite the larger search space; furthermore, we show that it can yield significant improvement in challenging environments, such as Humanoid and Wipe.
2. We demonstrate that including a variance penalty for multi-objective optimisation can obtain hyperparameter settings and reward shapes that substantially improve performance variance of a trained policy while achieving similar or better expected performance.

2 Background

We begin with some background on RL, define the optimisation of hyperparameters and reward shape, and present the selected algorithm applicable to these optimisation problems.

2.1 Reinforcement Learning and Reward-Shaping

In RL, an agent learns to optimise a *task objective* through interaction with an environment (Sutton & Barto, 2018). The environment is represented as a discounted Markov Decision Process (MDP) $\mathcal{M} := (\mathcal{S}, \mathcal{A}, p, r, \rho_0, \gamma)$, with state space $\mathcal{S}$, action space $\mathcal{A}$, an unknown transition probability distribution $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, distribution of the initial state $\rho : \mathcal{S} \to \mathbb{R}$, and discount rate $\gamma \in (0, 1)$. A policy $\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ selects an action with a certain probability for a given state. The agent interacts with the MDP to collect episodes $\tau = (s_0, a_0, r_1, s_1, \ldots, s_T)$, i.e., sequences of states, actions, and rewards over time steps $t = 0, \ldots, T$.

In applications, RL algorithms are very sensitive to the rewards in a given MDP to infer policies achieving the desired objective. To ease the policy search, reward shaping is the practice of designing a reward function $\tilde{r}^{\alpha,w} := \alpha \cdot (r + f^w)$ based on the original reward r of $\mathcal{M}$, where the reward shaping function $f^w : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ denotes the change in reward (Ng et al., 1999) parameterised by reward

weights $w \in \mathbb{R}^n$ and scaled by $\alpha \in \mathbb{R}^+$. The shaped reward essentially yields the modified MDP $\mathcal{M}_{\alpha,w} := (\mathcal{S}, \mathcal{A}, p, \tilde{r}^{\alpha,w}, \rho_0, \gamma)$. The function f^w is commonly designed by identifying key terms or events that should be rewarded or penalised and combining these as a weighted sum.

To obtain policies, there are now two hierarchical objectives for measuring performance. The outer task objective measures success in terms of the overall goal one wants to solve, and the inner objective in terms of maximising the collected shaped rewards when interacting with $\mathcal{M}_{\alpha,w}$. We formalise the overall task objective as o_{goal}, measuring the success of a task in the trajectory τ by assigning it a score $o_{\text{goal}}(\tau) \in \mathbb{R}$. Examples of such goals are achieving a certain objective or minimising time to perform a task. In addition to this outer task objective, we have the inner objective of maximising the expected return of the shaped rewards given by $J(\pi) = \mathbb{E}_{\tau\sim\pi}[\sum_{t=1}^{T} \gamma^t \cdot \tilde{r}_t]$. The common approach of RL is to maximise performance with regard to the task's objective o_{goal} by finding the policy π that maximizes the expected return $J(\pi)$. This typically involves tuning the parameters α and w of the shaped reward to obtain reward signals that facilitate finding policies in RL training that perform well with regard to o_{goal}. The task objective o_{goal} is not used during RL training and only measures success for a full trajectory τ. This allows measuring success much sparser than the shaped reward. Such sparse task objectives are commonly straightforward to define.

2.2 Combined Hyperparameter and Reward Shaping Optimisation

In practical RL applications, both the algorithm's hyperparameters and the environment's reward shape require tuning. For an environment $\mathcal{M}_{\alpha,w}$ with task objective o_{goal}, we can approach the refinement of hyper- and reward parameters as a two-level optimisation process. In the outer loop, an optimisation algorithm selects hyper- and reward parameters for the algorithm and environment. In the inner loop, these parameters are used for RL training, yielding a policy π. This policy is then assessed against a task performance metric $\mathcal{O}_{\text{goal}}$ based on o_{goal}, and its score is returned to the optimisation algorithm to determine the next parameter configuration.

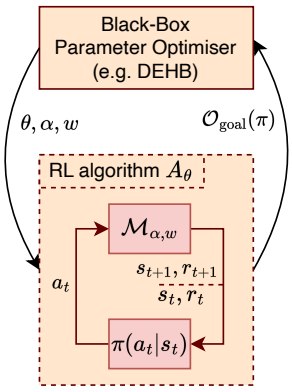


Figure 1: Illustration of the two-level optimisation process. Outer loop: hyper- and reward parameter optimisation; inner loop: RL training. In each iteration, the parameter optimiser chooses parameters and receives their performance measured by $\mathcal{O}_{\text{goal}}(\pi)$.

To evaluate a policy's performance with regard to o_{goal}, different metrics can be used. The single-objective performance metric $\mathcal{O}^{so}_{\text{goal}}(\pi) := \mathbb{E}_{\tau\sim\pi}[o_{\text{goal}}(\tau)]$ is exclusively concerned with optimising the average task score. The multi-objective metric $\mathcal{O}^{mo}_{\text{goal}}(\pi) := \mathbb{E}_{\tau\sim\pi}[o_{\text{goal}}(\tau)] - \sigma_{\tau\sim\pi}[o_{\text{goal}}(\tau)]$ includes an additional variance-based penalty, as described by Garcıa & Fernández (2015), preferring policies with low-performance variance and therefore consistent outcomes. Figure 1 illustrates the two-level optimisation process. To formally introduce the optimisation problems, we adopt the definition of algorithm configuration by Eggensperger et al. (2019) and adapt it to our RL context. Consequently, our focus is on optimising the hyperparameters of the RL algorithm, represented by θ, as well as the reward shaping, represented by α and w.

Definition. Consider an environment $\mathcal{M}_{\alpha,w} := (\mathcal{S}, \mathcal{A}, p, \tilde{r}^{\alpha,w}, \rho_0, \gamma)$ with *reward parameters* consisting of *reward scaling* $\alpha \in A$ and *reward weights* $w \in W$. Further, given an RL algorithm $A_\theta(\mathcal{M}_{\alpha,w})$ parametrised by *hyperparameters* $\theta \in \Theta$. This algorithm interacts with the environment $\mathcal{M}_{\alpha,w}$ and returns a policy π. For *performance metric* $\mathcal{O}_{\text{goal}}(\pi)$, we define the following optimisation problems:

Hyperparameter optimisation: For fixed reward parameters $\hat{\alpha}$ and $\hat{w}$, find $\theta^* \in \Theta$, s.t.

$$\theta^* \in \arg\max_{\theta\in\Theta} \mathcal{O}_{\text{goal}}(A_\theta(\mathcal{M}_{\hat{\alpha},\hat{w}})).$$

Reward parameter optimisation: For fixed hyperparameters $\hat{\theta}$, find $(\alpha^*, w^*) \in A \times W$, s.t.

$$(\alpha^*, w^*) \in \arg\max_{(\alpha,w)\in A\times W} \mathcal{O}_{\text{goal}}(A_{\hat{\theta}}(\mathcal{M}_{\alpha,w})).$$

Combined optimisation: Find $(\theta^*, w^*, \alpha^*) \in \Theta \times W \times A$, s.t.

$$(\theta^*, w^*, \alpha^*) \in \underset{(\theta,\alpha,w)\in\Theta\times A\times W}{\arg\max} \mathcal{O}_{\text{goal}}(A_\theta(\mathcal{M}_{\alpha,w})).$$

2.3 DEHB

Among the many optimisation methods for RL, DEHB has recently demonstrated superior performance (Eimer et al., 2023) and can be utilised for all three optimisation problems introduced in Section 2.2. DEHB is a black-box, multi-fidelity hyperparameter optimisation method combining differential evolution (Storn, 1996) and HyperBand (Li et al., 2018). Its multi-fidelity approach involves running numerous parameter configurations with a limited budget (e.g., a fraction of training steps) and advancing promising configurations to the next higher budget. This strategy allows for efficient exploration of the parameter space by testing a large number of configurations while avoiding wasteful evaluations on suboptimal configurations. The best-performing parameter configuration observed during optimisation is called the *incumbent* configuration.

3 Related Work

Our work aims to optimise RL algorithms by focusing jointly on hyperparameters and reward shapes to consistently obtain policies with high performance. The critical importance of hyperparameter tuning in deep RL is well-established (Andrychowicz et al., 2020; Henderson et al., 2018; Islam* et al., 2017). Similarly, reward shaping is recognised as important for fast and stable training (Ng et al., 1999; Gupta et al., 2022). The development of stable and reliable policies has been explored in risk-averse, multi-objective RL (Garcıa & Fernández, 2015; La & Ghavamzadeh, 2013), employing a straightforward variance-based performance penalty among many possible methods.

For black-box hyperparameter and reward shape optimisation, several methods have already been developed in the framework of AutoRL (Parker-Holder et al., 2020), a data-driven approach for systematically optimising RL algorithms through automated machine learning. However, these methods only target either the hyperparameter or reward-shape optimisation problem. Black-box methods optimising hyperparameters comprise population-based (Jaderberg et al., 2019; Parker-Holder et al., 2020; Wan et al., 2022) and multi-fidelity methods (Falkner et al., 2018; Awad et al., 2021). A recent study (Eimer et al., 2023) highlights the effectiveness of the multi-fidelity DEHB procedure for RL. Black-box methods for optimising reward shapes have been studied using evolutionary methods (Zheng et al., 2018; Faust et al., 2019; Wang et al., 2022). None of the mentioned works for AutoRL consider joint optimisation of hyperparameters and reward parameters.

The differences in performance achieved by separate hyperparameter optimisation and reward weight optimisations have been discussed by Faust et al. (2019), showing that reward parameters alone can improve performance and search efficiency compared to hyperparameter tuning. To the best of our knowledge, no comprehensive investigation has been conducted into whether combined reward and hyperparameter optimisation is generally possible and examined its potential in depth. Moving beyond this, Jaderberg et al. (2019) provided initial evidence that joint optimisation of hyperparameters and reward shape can outperform standard hyperparameter optimisation with manual reward shaping. However, their findings were limited to a single environment and presented as a custom solution, focusing solely on solving one specific environment. Furthermore, they did not thoroughly investigate the effects of a combined optimisation approach.

4 Setup of Experiments

In this section, we describe the setup of our experiments, the results of which will be discussed in Section 5. The experiments detailed in Section 4.1 aim to examine the relationship between specific hyperparameters and reward weights to better understand their interdependencies and the necessity of joint optimisation. Subsequently, the experiments in Section 4.2 empirically investigate

the performance of joint optimisation compared to individual optimisation to analyse differences in performance and cost. We trained PPO and SAC agents in four environments, each with a specific task objective: in Gymnasium's continuous LunarLander (Towers et al., 2023), a probe aims to minimise landing time; in Google Brax Ant and Humanoid (Freeman et al., 2021), a walking robot aims to maximise travel distance; and in Robosuite Wipe (Zhu et al., 2020), a simulated robot arm seeks to maximise the amount of dirt wiped from a table. All environments were chosen for non-trivial reward structures and for posing difficult hyperparameter optimisation problems. Specifically, Humanoid is notoriously difficult to solve, and the Wipe environment has a large reward parameter space that needs to be optimised. The Wipe environment represents a task that has not yet been extensively studied in the literature and is closely related to real-world applications. This allows us to test the applicability of our combined optimisation approach to environments that are less well-established in the field, yet of high practical interest. More information about the environments and their reward structure can be found in Appendix A. For training with LunarLander and Wipe, we employed the stable-baseline's Jax PPO and SAC implementation (Raffin et al., 2021), while for the Google Brax environments, we utilised the Google Brax GPU implementations. Implementation details can be found in the supplementary code repository `https://github.com/ADA-research/combined_hpo_and_reward_shaping`.

4.1 Interdependency of Hyperparameters and Reward Parameters

We conducted an exhaustive landscape analysis for PPO training on LunarLander, exploring pairwise combinations of a hyper- and reward parameter to better understand their interdependencies and substantiate the intuition that both components should be optimised jointly. The parameters not considered in each pair were fixed to their baseline value. A resolution of 100 values per parameter was applied, and the training performance of each pair was measured by the single-objective task performance and averaged over 10 seeds. In terms of hyperparameters, we considered the discount factor γ, generalised advantage estimation λ, and learning rate η, and in terms of reward parameters, the tilting, distance, and velocity weight. A logarithmic grid was applied to the discount factor and learning rate, with points positioned at equidistant logarithms. A uniform grid of equidistant points was applied to all other hyper- and reward parameters. Both choices were also used in our later optimisation experiments.

4.2 Optimisation of Hyperparameters and Reward Parameters

We conducted optimisation experiments to empirically compare the performance of joint optimisation with individual optimisation of hyperparameters and reward parameters; our goal was to understand the practicality of joint optimisation in finding well-performing hyperparameters and reward parameters without requiring any manual tuning.

We used the black-box algorithm DEHB for the three optimisation problems introduced in Section 2.2. The hyperparameter search spaces for PPO and SAC consist of four and seven parameters, respectively, that are commonly optimised and known to impact performance significantly. In particular, learning rate and discount factor were optimised for PPO and SAC. The hyperparameters not included in the search space were fixed at the baseline values of each training. For the reward function, we adjusted the weight parameters of each environment's reward shape. LunarLandar has four reward parameters, Ant and Humanoid three, and Wipe seven. The hyperparameters not optimised in reward-weight-only optimisation were set to the algorithm's training baseline values for the respective environment. The reward parameters not optimised in the case of hyperparameter-only optimisation were set to the default values of the respective environments. In the combined optimisation approaches, all hyperparameters and reward parameters in the search space were optimised from scratch. The search spaces and baseline values for hyperparameters are detailed in Appendix C, while the search spaces and default values for reward parameters are provided in Appendix A. DEHB has been demonstrated to outperform random search for hyperparameter optimisation (Awad et al., 2021). To analyse its effect on the optimisation of reward parameters, we used a random search

approach for the combined optimisation task, where the hyperparameters are optimised with DEHB, but the reward parameters are chosen randomly in each optimisation step.

In our setup, we set the fidelity of DEHB to equal the number of RL training steps. DEHB evaluates parameter configurations during the optimisation using three training step budgets, each increasing by a factor of three, with the largest matching the baseline's training steps. The fitness of each configuration is determined by the average performance metric after training with the designated steps over three random seeds. We performed experiments using the single- and multi-objective task objective performance metrics introduced in Section 2.2. Each optimisation experiment was conducted with five random seeds, and each resulting final incumbent configuration was evaluated by training using ten additional random seeds and evaluating performance on the corresponding task objective.

SAC is particularly sensitive to the scaling of the reward signal, since it influences the agent's exploration behaviour (Haarnoja et al., 2018). The reward scale α has only been optimised as part of the SAC baseline for Humanoid and Ant. Thus, we separately optimised the reward scale α only for Humanoid and Ant and kept the reward scale fixed to $\alpha = 1$ for the other environments. Details on how reward scaling was performed can be found in Appendix E.

The overall optimisation budget for DEHB with PPO and SAC equals 133 and 80 full training step budgets, respectively. Due to its computational demands, for the Wipe environment, we only considered SAC. The wall-clock times for the PPO and SAC optimisations are about 4 h and 60 h for LunarLander, 12 h and 15 h for Ant, 36 h and 60 h for Humanoid, and 120 h for Wipe. An overview of our execution environment and the overall computational cost can be found in Appendix G.

5 Empirical Results

We now present the results from our experiments. First, we show the complex interdependencies between hyperparameters and reward weights. Second, we demonstrate that joint optimisation can match or outperform individual optimisation and produce policies with substantially lower variance.

5.1 Interdependency between Hyperparameters and Reward Parameters

The parameter landscapes for LunarLander with PPO are shown in Figure 2. We observe an interdependency of varying strength between the hyperparameters and reward parameters. In all cases, the behaviour of specific reward parameters changes with different hyperparameter values. The ranges for reward parameters that lead to successful training vary depending on the hyperparameter and exhibit sharp boundaries in some cases. In particular, we observed ranges of reward parameters where performance deteriorates across all possible hyperparameter values.

Regarding the relation between hyperparameters and best-performing reward parameters (indicated by the blue lines in the plot), we observed a strong dependency for the distance weight and weaker dependencies for the velocity weights. In particular, we see a non-convex region of successful training parameters for the distance weight. Furthermore, we see large changes of the distance weight in its optimisation space. The tilting weight shows almost no dependency on the hyperparameters. Finally, our landscapes suggest that optimal value of the tilting weight is mostly near zero, which suggests that it is mostly irrelevant to RL training. In addition, we observed that the incumbent configurations in the joint optimisation experiments for LunarLander, presented in Section 5.2 (indicated by the black dots in the plot), are often not fully located in high-performing regions. We believe this is due to the larger search space during optimisation, which introduces additional dependencies on other parameters that impact performance in these regions. This further highlights the interdependencies of the parameters within the context of the full search space. In Appendix B, we report landscapes showing the optimal hyperparameters with respect to the reward parameters, showing similar dependencies.

Overall, our results indicate that hyperparameters and reward parameters are interdependent and that finding high-performing hyperparameters necessitates well-chosen reward parameters and vice versa. This confirms the intuition this work is based on: optimising the hyperparameters and reward shape should not be considered independently but instead approached jointly.

5.2 Joint Optimisation Performance

Table 1 reports the results of our optimisation experiments. Performance is shown in terms of single-objective task performance and the coefficient of variation (in percent). As outlined in Section 4.2, each experiment consists of five optimisation runs, with the incumbent parameter configuration of each run evaluated through ten RL training runs. The performance results in Table 1 are derived by calculating the median performance for each optimisation run across its ten evaluations and then computing the median of these five values for each experiment. The median coefficients of variation are calculated analogously. We chose the median over the mean to present our results, as it is more robust to outliers. To gain further insights into the statistical differences between the optimisation experiments, we employed linear mixed-effects model analysis (Gelman & Hill, 2006). For each combination of environment and algorithm, we performed pairwise comparisons of the aggregated 50 evaluation runs of the best-performing experiment with those of the related optimisation experiments. The mixed-effects model allows us to test for statistically significant differences in the results of two optimisation experiments, using all available data, while correctly handling the dependencies between optimisation runs. We show the best performance and all results that show no statistically significant differences (at significance level 0.05) to it in boldface. Details on how the test was conducted and its assumptions can be found in Appendix H. Boxplots of the median performance from the five optimisation runs for each experiment as well as boxplots of the 50 aggregated evaluations across all optimisation and training runs are presented in Appendix D.2.

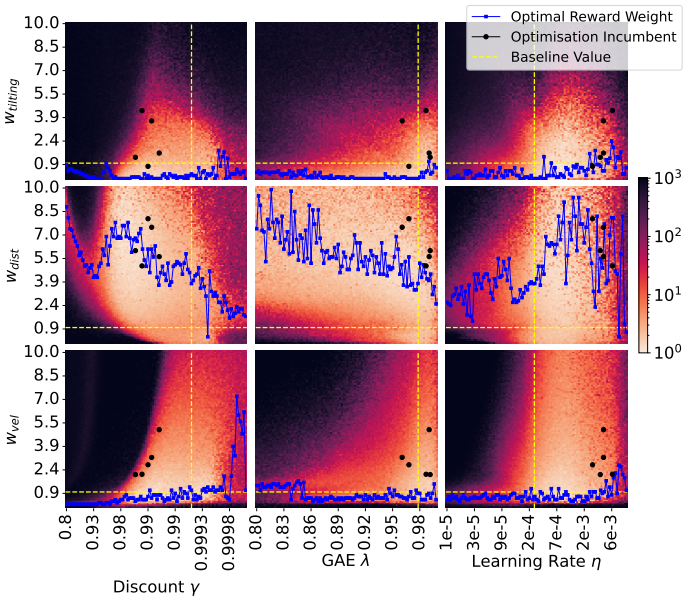


Figure 2: Landscapes depicting the average return on LunarLander for pairwise hyperparameters and reward weights over ten PPO trainings. Lower values (lighter) correspond to faster landing (better performance). Yellow lines mark each parameter's default value. Blue lines denote the best-performing reward weights for each hyperparameter value. The black dots mark the incumbent configurations found in the joint optimisation experiments in Section 5.2.

Our results show that simultaneously optimising hyperparameters and reward parameters consistently matches or outperforms individual optimisation, without depending on tuned baseline parameters for non-optimised components. The only outlier is the single-objective PPO Ant optimisation. Significant performance gains are observed in the complex Humanoid and Wipe environments, while the simpler Ant and LunarLander environments, which are mostly solved using baseline parameter settings, generally show no additional improvements from joint optimisation. However, even if performance only matches the baseline parameters, joint optimisation still offers the advantage of not requiring hand-tuning, while addressing the mutual dependencies of hyperparameter and reward parameters.

Environment	HPO	RPO	Task Performance $\mathbb{E}_{\tau\sim\pi}[o_{\text{goal}}(\tau)]$ $(100 \cdot \text{CV}_{\tau\sim\pi}[o_{\text{goal}}(\tau)])$			
			PPO		SAC	
			Single Obj.	Multi Obj.	Single Obj.	Multi Obj.
Gymnasium LunarLander (minimise)	base		273 (11%)		208 (27%)	
	base	DEHB	287 (31%)	**223 (10%)**	**175 (14%)**	**174 (13%)**
	DEHB	base	265 (27%)	277 (11%)	194 (23%)	186 (15%)
	DEHB	RS	262 (38%)	252 (24%)	**171 (15%)**	193 (18%)
	DEHB (ours)		**234 (25%)**	**227 (15%)**	**177 (23%)**	**182 (21%)**
Google Brax Ant (maximise)	base		6785 (16%)		8054 (28%)	
	base	DEHB	6706 (17%)	6663 (14%)	7927 (32%)	7994 (29%)
	DEHB	base	**8111 (14%)**	7842 (6%)	**8282 (21%)**	**8216 (13%)**
	DEHB	RS	8013 (16%)	-	**8064 (21%)**	-
	DEHB (ours)		8049 (12%)	**7923 (6%)**	**8199 (23%)**	**8169 (18%)**
Google Brax Humanoid (maximise)	base		4196 (<1%)		3273 (11%)	
	base	DEHB	4464 (<1%)	4472 (<1%)	5284 (11%)	5208 (8%)
	DEHB	base	4826 (1%)	4719 (<1%)	4881 (18%)	4466 (15%)
	DEHB	RS	5112 (2%)	-	**5913 (17%)**	-
	DEHB (ours)		**5433 (7%)**	**5485 (1%)**	**6033 (12%)**	**6103 (1%)**
Robosuite Wipe (maximise)	base		-		101 (24%)	
	base	DEHB			108 (24%)	114 (20%)
	DEHB	base			132 (10%)	**131 (11%)**
	DEHB	RS			134 (10%)	-
	DEHB (ours)				**136 (8%)**	**130 (10%)**

Table 1: Median performance for our optimisation experiments. *HPO* and *RPO* show the optimisation method for hyper- and reward parameters: *base* for fixing to baseline values, *DEHB* and *RS* for optimisation with DEHB or random search. Each environment's first row is baseline performance, followed by optimising reward-, hyperparameters, or both. Best performance are highlighted in bold for each environment and column (multiple bold entries mark statistically insignificant differences).

In LunarLander, Ant, and Humanoid, the optimised incumbent of our joint optimisation on the respective default reward function generally achieves performance considered to solve the environment. For Robosuite Wipe, the average objective score comes close to the maximum of 142. Especially in our experiments with LunarLander, Humanoid and Wipe, the policy improvements could be seen not only in the improved average objective score, but also in qualitative improvements in the agents' behaviour (representative videos can be found in the supplementary material). Therefore, combined optimisation shows competitive performance for already well-studied environments as well as the less-studied Wipe environment. We do not observe a clear pattern in Table 1 that indicates whether optimising solely hyperparameters or reward parameters consistently outperforms the other. This underscores the necessity of joint optimisation to automatically determine which component requires more optimisation, especially for novel environments, where prior knowledge about the dynamics is lacking. Unsurprisingly, DEHB outperforms random search in almost all our experiments. We report the best hyperparameter and reward parameter values for each environment and algorithm in Appendix F. Appendix D.1 provides the results of policies obtained for each configuration when evaluated using the default shaped reward function of the respective environment.

In Figure 3, we show median incumbent performance during our SAC experiments at each optimisation time step. The speed of the combined optimisation is comparable to that of the individual optimisation approaches, despite involving much larger search spaces. In all environments except multi-objective Wipe, combined optimisation already matches the performance of the best-performing individual optimisation after roughly a third of the total optimisation steps and continues to improve. For multi-objective Wipe, this is achieved after two-thirds of the total optimisation steps. Similar trends are observed for the PPO results, shown in Appendix F. This indicates that combined

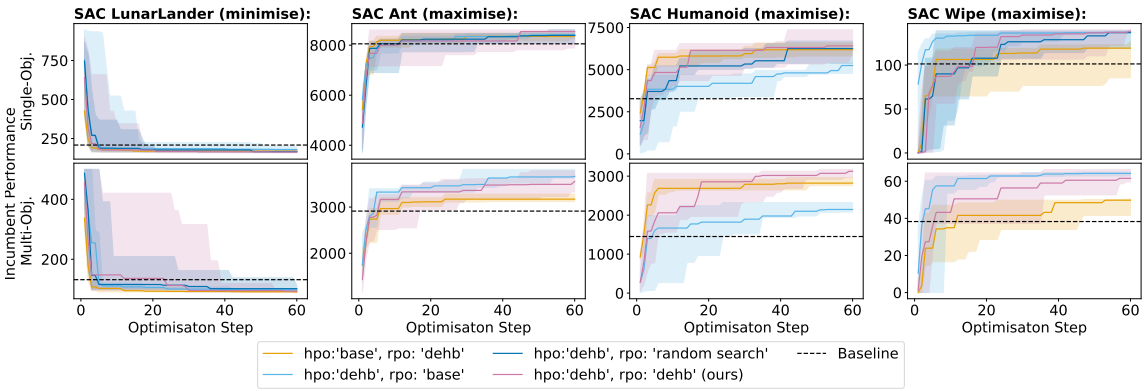


Figure 3: Incumbent performance in terms of median optimisation objective across the five optimisation runs for the SAC experiments at each time step; shaded areas indicate min and max values. The performance drop in the multi-objective experiments is due to the weighted penalty term.

optimisation, despite the larger search space, requires minimal additional computational effort in terms of optimisation time.

5.3 Single- *vs* Multi-objective Optimisation

From Table 1, we conclude that multi-objective optimisation can improve policy stability by including a penalty for large standard deviation in performance. These improvements come with only marginal performance loss and sometimes even achieve slight gains; this is the case for Ant, Humanoid and LunarLander, where, in particular, for Humanoid and PPO LunarLander, improved performance is achieved. Only for Wipe, we observed that stability is not further improved compared to the single-objective training. RL is notoriously sensitive to hyperparameter settings. Therefore, optimising hyperparameters using a variance penalty for newly developed algorithms or novel scenarios can lead to increased stability and thus greatly facilitate research and applications.

6 Conclusions and Future Work

In this work, we have demonstrated the importance of jointly optimising hyperparameters and reward parameters. We illustrated dependencies in a simple environment, highlighting the circular dependency encountered in optimising hyperparameters and reward parameters and underscoring the need for simultaneous optimisation. Our empirical results indicate that this joint optimisation is feasible and can match or surpass the performance of individual optimisation approaches without requiring separate parameter tuning for the non-optimised component. Additionally, we demonstrated that this approach requires minimal extra computational effort and is applicable to practical environments, not yet extensively studied. We conclude that combined optimisation should be the best practice for RL optimisation. While we have focused on optimising specific reward parameters within a predefined reward structure, future work should explore a broader range of reward function combinations. Such an extension could consider further aspects of the reward function, including metrics, exponentiation, or specific functional choices, such as nonlinear transformations.

Our results further indicate that including a variance penalty in a multi-objective optimisation can substantially enhance the performance variance of a given policy, with little or no reductions in performance. This emphasises the value of combined optimisation in achieving a good balance between a high average objective score and achieving this performance consistently. This improvement in stability is often a crucial requirement in reinforcement learning, enhancing reproducibility and the reliability of results in varying environments. Future research should investigate more sophisticated risk-averse metrics and thoroughly assess the trade-off between a policy's performance and stability.

Acknowledgments

The authors would like to thank Theresa Eimer for helpful discussions regarding the search space design, Anna Münz for help with statistical tests, and Anja Jankovic for helpful feedback. We gratefully acknowledge computing resources provided by the NHR Center NHR4CES at RWTH Aachen University (p0021208), funded by the Federal Ministry of Education and Research, and the state governments participating on the basis of the resolutions of the GWK for national high performance computing at universities. This research was supported in part by an Alexander von Humboldt Professorship in AI held by HH and by the "Demonstrations- und Transfernetzwerk KI in der Produktion (ProKI-Netz)" initiative, funded by the German Federal Ministry of Education and Research (BMBF, grant number 02P22A010).

References

Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's rube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Leonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, et al. What matters for on-policy deep actor-critic methods? a large-scale study. In *International Conference on Learning Representations*, 2020.

N. Awad, N. Mallik, and F. Hutter. DEHB: Evolutionary hyberband for scalable, robust and efficient hyperparameter optimization. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pp. 2147–2153. IJCAI, 2021.

Thomas Bi and Raffaello D'Andrea. Sample-efficient learning to solve a real-world labyrinth game using data-augmented model-based reinforcement learning, 2023.

Violet A Brown. An introduction to linear mixed-effects modeling in r. *Advances in Methods and Practices in Psychological Science*, 4, 2021.

Katharina Eggensperger, Marius Lindauer, and Frank Hutter. Pitfalls and best practices in algorithm configuration. *Journal of Artificial Intelligence Research*, 64:861–893, 2019.

Theresa Eimer, Marius Lindauer, and Roberta Raileanu. Hyperparameters in reinforcement learning and how to tune them. In *International Conference on Machine Learning*, pp. 9104–9149. PMLR, 2023.

Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*, pp. 1437–1446. PMLR, 2018.

Aleksandra Faust, Anthony Francis, and Dar Mehta. Evolving rewards to automate reinforcement learning. In *6th ICML Workshop on Automated Machine Learning*, 2019.

C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - a differentiable physics engine for large scale rigid body simulation, 2021. URL http://github.com/google/brax.

Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.*, 16:1437–1480, 2015.

Andrew Gelman and Jennifer Hill. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge university press, 2006.

Abhishek Gupta, Aldo Pacchiano, Yuexiang Zhai, Sham Kakade, and Sergey Levine. Unpacking reward shaping: Understanding the benefits of reward engineering on sample complexity. *Advances in Neural Information Processing Systems*, 35:15281–15295, 2022.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 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.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

Riashat Islam*, Peter Henderson*, Maziar Gomrokchi, and Doina Precup. Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. *ICML Workshop on Reproducibility in Machine Learning*, 2017.

Max Jaderberg, Wojciech M. Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castañeda, Charles Beattie, Neil C. Rabinowitz, Ari S. Morcos, Avraham Ruderman, Nicolas Sonnerat, Tim Green, Louise Deason, Joel Z. Leibo, David Silver, Demis Hassabis, Koray Kavukcuoglu, and Thore Graepel. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364:859–865, 2019.

Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620: 982–987, 2023.

Prashanth La and Mohammad Ghavamzadeh. Actor-critic algorithms for risk-sensitive mdps. *Advances in Neural Information Processing Systems*, 26, 2013.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *J. Mach. Learn. Res.*, 18:1–52, 2018.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*, volume 99, pp. 278–287. Citeseer, 1999.

Jack Parker-Holder, Vu Nguyen, and Stephen J Roberts. Provably efficient online hyperparameter optimization with population-based bandits. In *Advances in Neural Information Processing Systems*, volume 33, pp. 17200–17211. Curran Associates, Inc., 2020.

Jack Parker-Holder, Raghu Rajan, Xingyou Song, André Biedenkapp, Yingjie Miao, Theresa Eimer, Baohe Zhang, Vu Nguyen, Roberto Calandra, Aleksandra Faust, et al. Automated reinforcement learning (autorl): A survey and open problems. *Journal of Artificial Intelligence Research*, 74: 517–568, 2022.

Antonin Raffin. Rl baselines3 zoo. https://github.com/DLR-RM/rl-baselines3-zoo, 2020.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *J. Mach. Learn. Res.*, 22:1–8, 2021.

Holger Schielzeth, Niels J Dingemanse, Shinichi Nakagawa, David F Westneat, Hassen Allegue, Céline Teplitsky, Denis Réale, Ned A Dochtermann, László Zsolt Garamszegi, and Yimen G Araya-Ajoy. Robustness of linear mixed-effects models to violations of distributional assumptions. *Methods in Ecology and Evolution*, 11:1141–1152, 2020.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 2017.

Rainer Storn. On the usage of differential evolution for function optimization. In *Proceedings of North American Fuzzy Information Processing*, pp. 519–523. IEEE, 1996.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018.

Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, 2023.

Xingchen Wan, Cong Lu, Jack Parker-Holder, Philip J. Ball, Vu Nguyen, Binxin Ru, and Michael Osborne. Bayesian generational population-based training. In *Proceedings of the First International Conference on Automated Machine Learning*, volume 188 of *Proceedings of Machine Learning Research*, pp. 14/1–27. PMLR, 2022.

Ludi Wang, Zhaolei Wang, and Qinghai Gong. Bi-level optimization method for automatic reward shaping of reinforcement learning. In *31st International Conference on Artificial Neural Networks*, pp. 382–393. Springer, 2022.

Zeyu Zheng, Junhyuk Oh, and Satinder Singh. On learning intrinsic rewards for policy gradient methods. *Advances in Neural Information Processing Systems*, 2018.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, and Yifeng Zhu. Robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020.

Reward Weight	LunarLander		Ant		Humanoid		Wipe	
	Default Value	Search Space	Default Value	Search Range	Default Value	Search Space	Default Value	Search Space
w_{dist}	100	$[0, 1000]$	1	$[0, 10]$	1.25	$[0, 10]$	5	$[0, 10]$
$w_{\text{dist_th}}$	-		-		-		5	$[0, 10]$
w_{vel}	100	$[0, 1000]$	-		-		0	$[0, 1]$
w_{tilting}	100	$[0, 1000]$	-		-		-	
w_{contact}	10	$[0, 100]$	-		-		0.01	$[0, 1]$
w_{healthy}	-		1	$[0, 10]$	5	$[0, 10]$	-	
$w_{\text{unhealthy}}$	-		-		-		-10	$[-100, 0]$
w_{force}	-		0.5	$[0, 1]$	0.1	$[0, 1]$	0.05	$[0, 1]$
w_{wiped}	-		-		-		50	$[0, 100]$

Table 2: Default reward weights for each environment and the corresponding search spaces of our optimisation experiments.

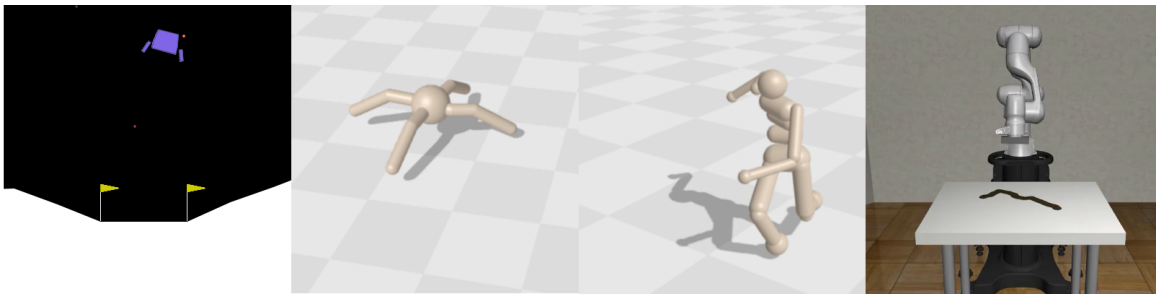


Figure 4: Illustrations from left to right of the environments Gymnasium LunarLander, Google Brax Ant and Humanoid, and Robosuite Wipe.

A Environments and Reward Parameter Search Spaces

In this section, we give a detailed overview of the environments used in our experiments and the reward parameters we are optimising in each environment's reward function. The default values and respective search spaces of the reward parameters can be found in Table 2. We opted not to optimise the terminal rewards r for the LunarLander and Wipe environments, as these constitute the sparse rewards that are addressed through the optimisation of the reward shape.

For mapping a reward weight w_i to its search space, we always used the mapping $w_i \mapsto [0, 10^n]$, where $n \in \mathbb{N}_0$ is the smallest integer such that $w_i < 10^n$; we chose this approach, since it preserves the general magnitude of the reward parameters, while relying less on their initial ratios. We believe that practitioners typically have a rough idea about the importance of different components but find it difficult to obtain the exact ratios between them.

A.1 Gymnasium LunarLander:

The objective of the environment is to navigate a probe to a designated landing platform safely. We considered the environment's variant with continuous control inputs. In the shaped reward, positive rewards are given for moving closer to the landing platform and negative rewards for moving further away. A positive reward is granted for making successful contact with the platform using the probe's legs. Negative rewards are imposed for high velocities and tilting the probe excessively. Fuel consumption by the probe's engine, when activated, results in negative rewards. However, we considered fuel consumption as a constant physical attribute of the probe and did not consider it in our optimisations. The overall shaped reward function is given by

$$\tilde{r}^{\alpha,w} := \alpha \cdot (w_{\text{dist}} \cdot \tilde{r}_{\text{dist}} + w_{\text{contact}} \cdot \tilde{r}_{\text{contact}} - w_{\text{vel}} \cdot \tilde{r}_{\text{vel}} - w_{\text{tilting}} \cdot \tilde{r}_{\text{tilting}} - \tilde{r}_{\text{fuel}} + r_{terminal}),$$

with $r_{terminal}$ being the environment's sparse reward signal of successful landing or crashing.

A.2 Google Brax Ant and Humanoid:

The task in both environments is to train a robot to walk forward in a specified direction. In Ant, the robot is designed to resemble a four-legged ant and is human-like in Humanoid. The environment's rewards consist of positive rewards for staying healthy (being able to continue walking) and a reward for the distance travelled in each timestep. A negative reward for exercising large forces on the robot's joints is obtained. The overall shaped reward function is given by

$$\tilde{r}^{\alpha,w} := \alpha \cdot (w_{\text{dist}} \cdot \tilde{r}_{\text{dist}} + w_{\text{healthy}} \cdot \tilde{r}_{\text{healthy}} - w_{\text{force}} \cdot \tilde{r}_{\text{force}}).$$

A.3 Robosuite Wipe:

The task is to wipe a table of dirt pegs with a simulated robot arm equipped with a sponge. Positive rewards are obtained for the sponge's distance to the dirt pegs, having contact with the table and exercising appropriate pressure on the table. Negative rewards are obtained for applying excessive force on the table, large accelerations while moving or arm-limit collisions resulting in an unhealthy state. The overall shaped reward function is given by

$$\begin{aligned}\tilde{r}^{\alpha,w} := \alpha \cdot (&w_{\text{wiped}} \cdot \tilde{r}_{\text{wiped}} + w_{\text{dist}} \cdot (1 - \tanh(w_{\text{dist_th}} \cdot \tilde{r}_{\text{dist}})) + w_{\text{contact}} \cdot \tilde{r}_{\text{contact}} \\ &- w_{\text{force}} \cdot \tilde{r}_{\text{force}} - w_{\text{vel}} \cdot \tilde{r}_{\text{vel}} + w_{\text{unhealthy}} \cdot \tilde{r}_{\text{unhealthy}} + r_{terminal}),\end{aligned}$$

with $r_{terminal}$ being the environment's sparse reward for wiping the table clean or not.

B Interdependecy between Hyperparameters and reward parameters

In Figure 5, we present the same landscapes as in Figure 2 but mark the best-performing hyperparameter value for each reward weight (shown as the blue lines).

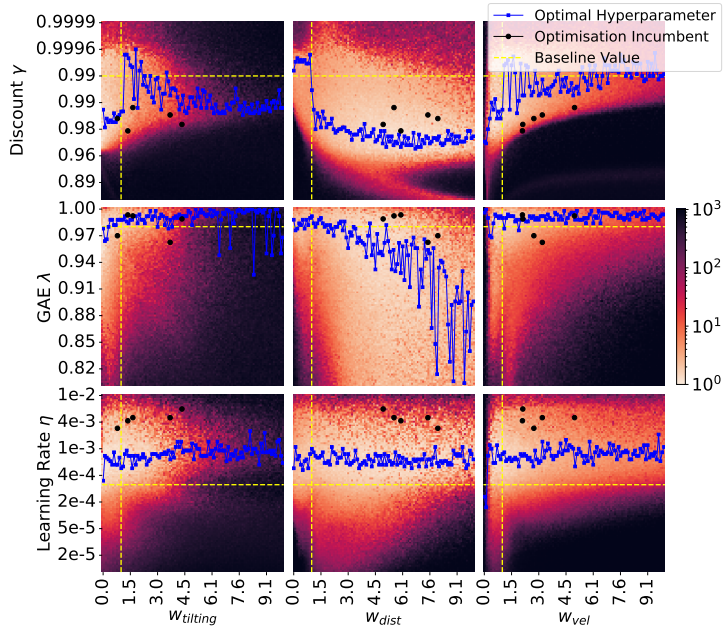


Figure 5: Landscapes depicting the average return on Gymnasium LunarLander for pairwise hyper- and reward parameters over ten PPO trainings. Lower values (lighter) correspond to faster landing time and, thus, better performance. The yellow lines mark the default values for each parameter. The blue line denotes the best-performing hyperparameter value for each specific reward value. The black dots mark the incumbent configurations found in the joint optimisation experiments in Section 5.2

Hyperparameter	Baseline Values			Search Space	
	LunarLander	Ant	Humanoid	Range	Log-Scale
learning rate	$3 \cdot 10^{-4}$			$[1e^{-6}, 0.01]$	Yes
discounting	0.999	0.97		$[0.001, 0.02]$	
gae	0.98	0.95		$[0.8, 1.0]$	No
clipping rate	0.2	0.3		$[0.1, 0.4]$	
entropy coef	0.01	0.001		$[0.0, 0.1]$	
value coef	0.5			$[0.3, 0.7]$	
batch size	64	2048	1024	$\{b/2, b, 2b\}$	
training steps	$1e^6$	5242880		-	
episode length	1000				
num envs	1	256	64		
unroll length	1024	5	10		
num minibatches	-	32			
epochs	4		8		

Table 3: PPO baseline parameters for each environment. Due to differences in the implementation of stable baselines 3 JAX and Google Brax PPO, there is no hyperparameter for the number of minibatches in the case of LunarLander. The search space for the batch size is always a categorical choice over the power of two below and above the baseline value.

Hyperparameter	Baseline Values				Search Space	
	LunarLander	Ant	Humanoid	Wipe	Range	Log-Scale
learning rate	$3 \cdot 10^{-4}$				$[1e^{-6}, 0.01]$	Yes
discounting	0.99	0.95	0.99		$[0.001, 0.02]$	
tau	0.01	0.005	0.005		$[0.001, 0.1]$	
batch size	256	512	1024	256	$\{b/2, b, 2b\}$	No
training steps	500000	5242880	5242880	1250000	-	
episode length	1000			500		
num envs	1	256	64	5		
training freq	1					
gradient steps	1	64	8	2		
min replay buffer size	10000	8192		100		
max replay buffer size	10^6	1038576		10^6		

Table 4: SAC baseline parameters for each environment. The search space for the batch size is always a categorical choice over the power of two below and above the baseline value.

We can again observe strong dependencies and large changes in the best-performing hyperparameters for the discount factor and the generalised advantage estimate regarding the distance reward parameter. Only for the learning rate, we see almost no dependency on the reward parameters.

C Hyperparameter Baselines and Search Spaces

In Table 3 and Table 4, we present the baselines and search spaces for PPO and SAC, respectively. We reproduced all baselines in our setup and, in some cases, made slight modifications to improve their performance when possible. The baselines for continuous LunarLander PPO and SAC have both been obtained from the stable-baselines 3 Zoo (Raffin, 2020). The Google Brax Ant and Humanoid baselines are obtained from Brax's GitHub repository. For PPO, hyperparameters have been shared via a Google Colab notebook in the Google Brax GitHub repository. For SAC, we utilise the performance results of a published hyperparameter sweep. In the case of Humanoid, we made small adjustments to the reported best-performing parameters in order to obtain the best results in our setup. Our adjustments align with the considered search space of their hyperparameter sweep. We utilised the published baseline in Zhu et al. (2020) for Robosuite Wipe with small adjustments to the training frequency and gradient steps to obtain better performance in our setup.

Environment	HPO	RPO	Default Shaped Reward Return			
			PPO		SAC	
			Single Obj.	Multi Obj.	Single Obj.	Multi Obj.
Gymnasium LunarLander (maximise)	base		274 (8%)		276 (7%)	
	base	DEHB	266 (10%) ($p = 2.3 \cdot 10^{-04}$)	**283 (7%)**	286 (7%) ($p = 3.4 \cdot 10^{-02}$)	**287 (7%)**
	DEHB	base	275 (7%) ($p = 5.7 \cdot 10^{-04}$)	272 (7%) ($p = 3.1 \cdot 10^{-04}$)	280 (7%) ($p = 1.0 \cdot 10^{-20}$)	283 (7%) ($p = 5.2 \cdot 10^{-09}$)
	DEHB	RS	275 (8%) ($p = 1.7 \cdot 10^{-02}$)	276 (8%) ($p = 1.7 \cdot 10^{-03}$)	**287 (7%)**	283 (7%) ($p = 7.0 \cdot 10^{-07}$)
	DEHB (ours)		**234 (25%)**	**283 (7%)**	**287 (7%)**	285 (7%) ($p = 2.2 \cdot 10^{-02}$)
Google Brax Ant (maximise)	base		7293 (17%)		8065 (30%)	
	base	DEHB	7235 (18%) ($p = 3.1 \cdot 10^{-114}$)	7236 (17%) ($p = 3.4 \cdot 10^{-52}$)	7600 (34%) ($p = 5.9 \cdot 10^{-19}$)	7814 (32%) ($p = 1.0 \cdot 10^{-03}$)
	DEHB	base	**8379 (17%)**	**8127 (10%)**	**8169 (25%)**	**8037 (19%)**
	DEHB	RS	8063 (18%) ($p = 3.6 \cdot 10^{-09}$)	-	7636 (24%) ($p = 2.2 \cdot 10^{-10}$)	-
	DEHB (ours)		8254 (16%) ($p = 4.0 \cdot 10^{-03}$)	8124 (9%) ($p = 3.9 \cdot 10^{-02}$)	7717 (28%) ($p = 1.0 \cdot 10^{-09}$)	7866 (19%) ($p = 2.1 \cdot 10^{-03}$)
Google Brax Humanoid (maximise)	base		10016 (<1%)		3273 (11%)	
	base	DEHB	10256 (<1%) ($p = 6.6 \cdot 10^{-24}$)	10439 (<1%) ($p = 5.9 \cdot 10^{-23}$)	10509 (11%) ($p = 2.3 \cdot 10^{-07}$)	10695 (11%) ($p = 6.4 \cdot 10^{-13}$)
	DEHB	base	10850 (2%) ($p = 1.1 \cdot 10^{-07}$)	10726 (<1%) ($p = 2.1 \cdot 10^{-22}$)	10317 (19%) ($p = 7.5 \cdot 10^{-17}$)	9763 (17%) ($p = 2.6 \cdot 10^{-64}$)
	DEHB	RS	11204 (<1%) ($p = 6.3 \cdot 10^{-03}$)	-	**11746 (17%)** ($p = 4.3 \cdot 10^{-01}$)	-
	DEHB (ours)		**11599 (7%)**	**11562 (<1%)**	**12141 (11%)**	**12292 (8%)**
Robosuite Wipe (maximise)	base		-		108 (38%)	
	base	DEHB			77 (57%) ($p = 1.7 \cdot 10^{-98}$)	78 (58%) ($p = 1.3 \cdot 10^{-159}$)
	DEHB	base			**134 (20%)**	**131 (20%)** ($p = 9.7 \cdot 10^{-1}$)
	DEHB	RS			126 (21%) ($p = 4.7 \cdot 10^{-11}$)	-
	DEHB (ours)				127 (25%) ($p = 2.3 \cdot 10^{-23}$)	**132 (20%)**

Table 5: Results of the trained policies for each optimisation experiment evaluated on each environment's default shaped reward function and the coefficients of variations in parenthesis. Columns *HPO* and *RPO* indicate the respective optimisation methods: *base* for baseline values, *DEHB* and *RS* for optimisation with DEHB or random search. Hence, the first row of each environment is the baseline performance, followed by rows optimising reward parameters, hyperparameters, or both. Each experiment's performance is computed similarly as in Table 1. Performances without significant statistical differences to the best-performing optimisation experiment, are highlighted in bold for each environment. We reported the test's p-values of the comparison in each cell.

D Additional Optimisation Results and Visualisations

In the following sections, we present additional plots and tables on the performance distributions of our different optimisation experiments.

D.1 Default Shaped Reward Function Evaluation

In Table 5, we present the returns of the obtained policies for each optimisation experiment, evaluated on the corresponding environment's default-shaped reward function. Our observations indicate that for LunarLander and Humanoid, the combined optimisation consistently matches or outperforms the best performance, except for multi-objective SAC training for LunarLander. This suggests that the benefits on the task performance effectively transfer to the default shaped reward function, even though the policies were not specifically optimised for it.

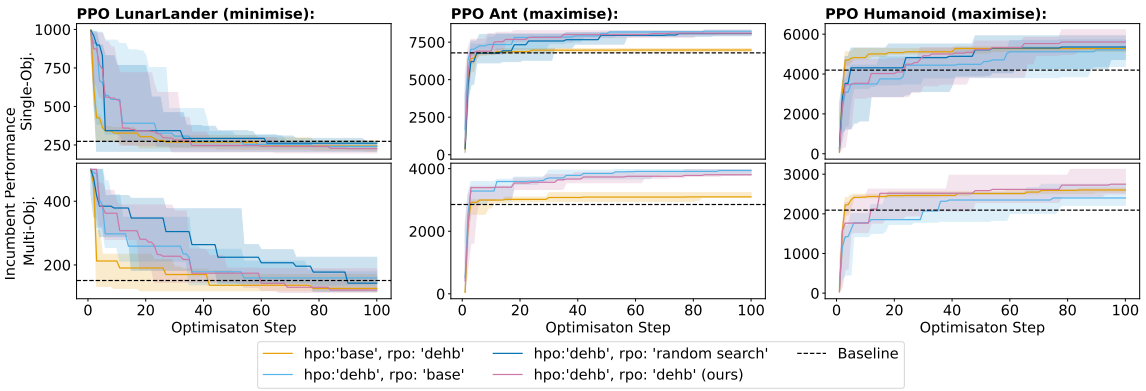


Figure 6: The median optimisation objective's incumbent performance across the five optimisation runs for the PPO experiments at each time step. The min and max are given as error bars.

In the case of Ant, the performance is slightly lower than that achieved through hyperparameter-only optimisation, yet qualitative analysis shows that the environment is still clearly solved. For the Robosuite Wipe environment, however, the combined optimisation performs significantly worse than the hyperparameter-only optimisation, which starkly contrasts with the evaluation of the environment's task performance.

Further analysis reveals that this discrepancy is due to the default shaped reward function's inadequate representation of the environment's task objective. Specifically, policies that do not completely clean the table but maintain contact with it until the end of an episode can accumulate a higher overall return compared to those that quickly complete the cleaning task. Consequently, the policies resulting from combined optimisation, which prioritise rapid table cleaning, achieve lower returns despite better performance in wiping the table.

D.2 Optimisation Performance Boxplots

In addition to the results presented in Table 1, we provide an overview of the full dataset as boxplots. Figures 7 and 8 display boxplots for the median performances of each experiment's five optimisation runs. For each experiment's optimisation run, we calculate the median performance across its ten evaluation trainings and present boxplots for the resulting five values per experiment. Figures 9 and 10 showcase boxplots for the combined 50 evaluation training performances, obtained by aggregating all ten evaluation training performances for each of the five optimisation runs per experiment. In each boxplot, the baseline's median performance is marked with a red line.

Consistent with our analysis in Section 4.2, we observe that combined optimisation can match or surpass the individual optimisations of hyperparameters and reward parameters. Furthermore, multi-objective optimisation substantially enhances stability with minimal or no reductions in performance.

D.3 Incumbent Performance during Optimisation

Figure 6 depicts the median incumbent performance during each PPO optimisation experiment. We note that the optimisation steps necessary to surpass the baselines generally occur much earlier than the complete duration of the optimisation, similar to the findings of Eimer et al. (2023). However, we also observe that continuous improvement is still achieved after exceeding the baseline performance. Moreover, the combined optimisation does not seem to be significantly slower than optimising hyperparameters or reward parameters alone, suggesting that the combined optimisation can enhance results without additional costs, similar to the SAC results presented in Figure 3.

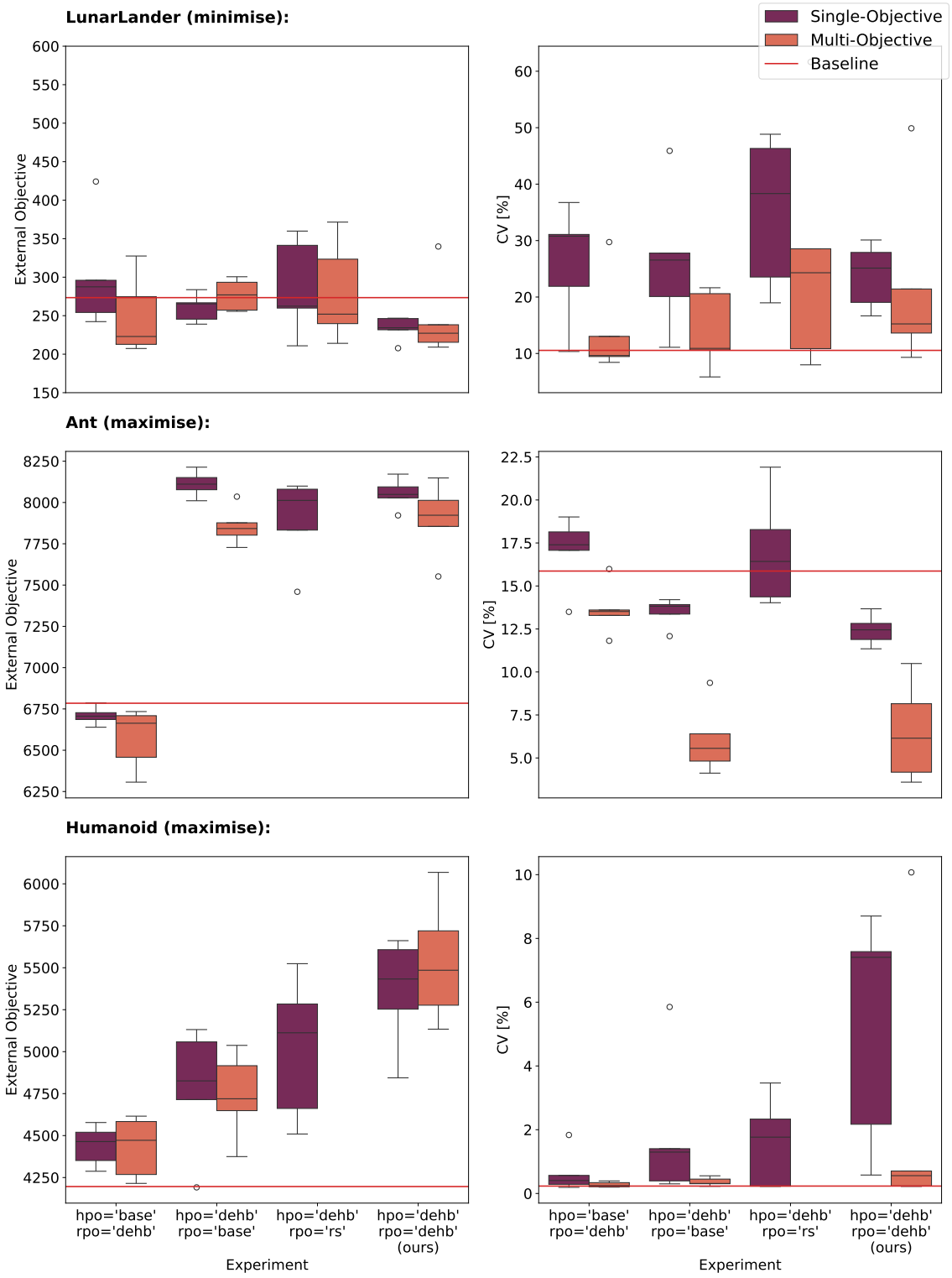


Figure 7: Boxplots for the PPO optimisation of the five median performances of each experiment's optimisation runs. We denote by DEHB, rs and base if hyperparameters (hpo) or reward parameters (rpo) were optimised with DEHB, random search or fixed to their baseline values. The red line denotes the baseline performance.

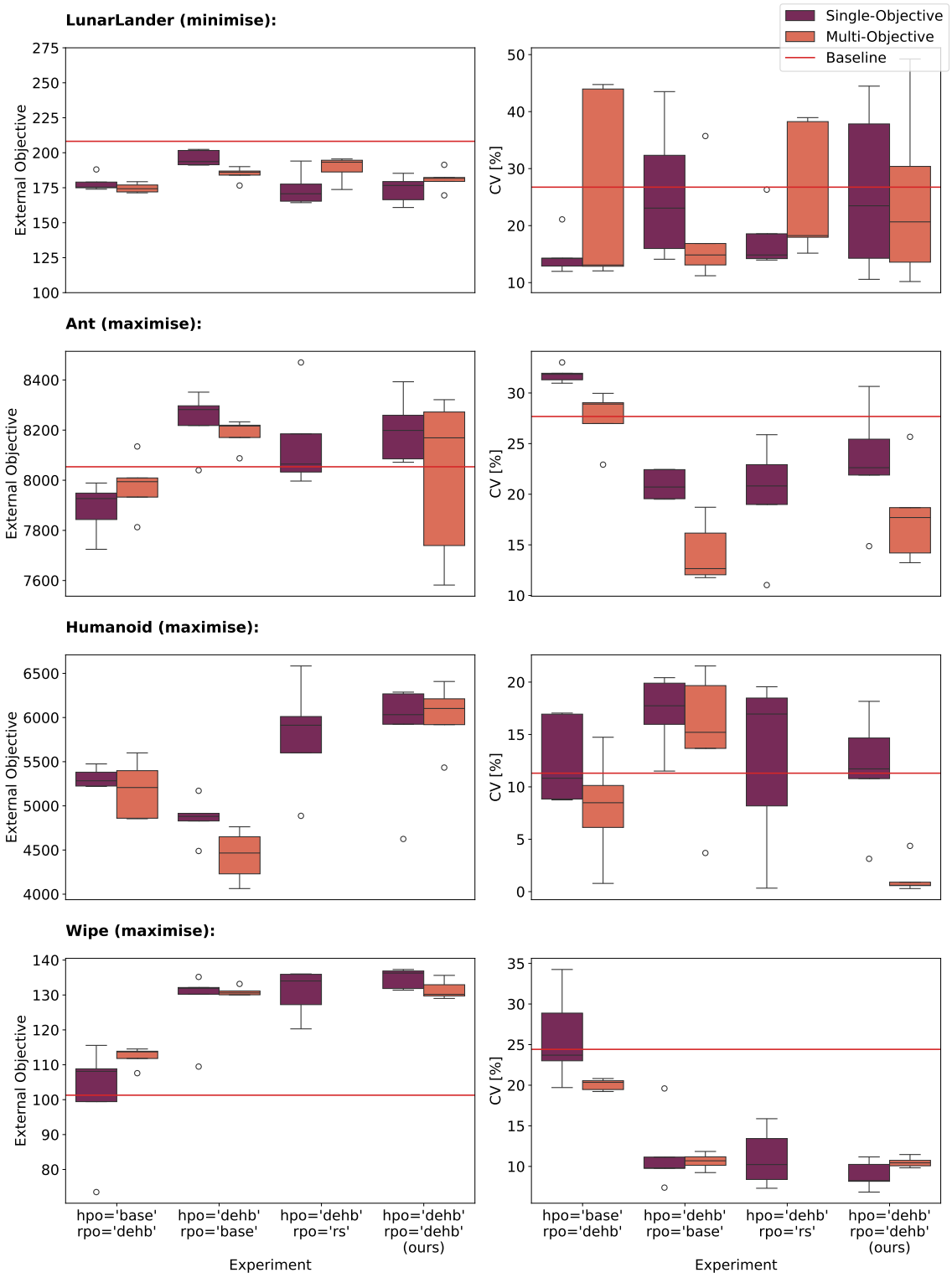


Figure 8: Boxplots for the SAC optimisation of the five median performances of each experiment's optimisation runs. We denote by DEHB, rs and base if hyperparameters (hpo) or reward parameters (rpo) were optimised with DEHB, random search or fixed to their baseline values. The red line denotes the baseline performance.

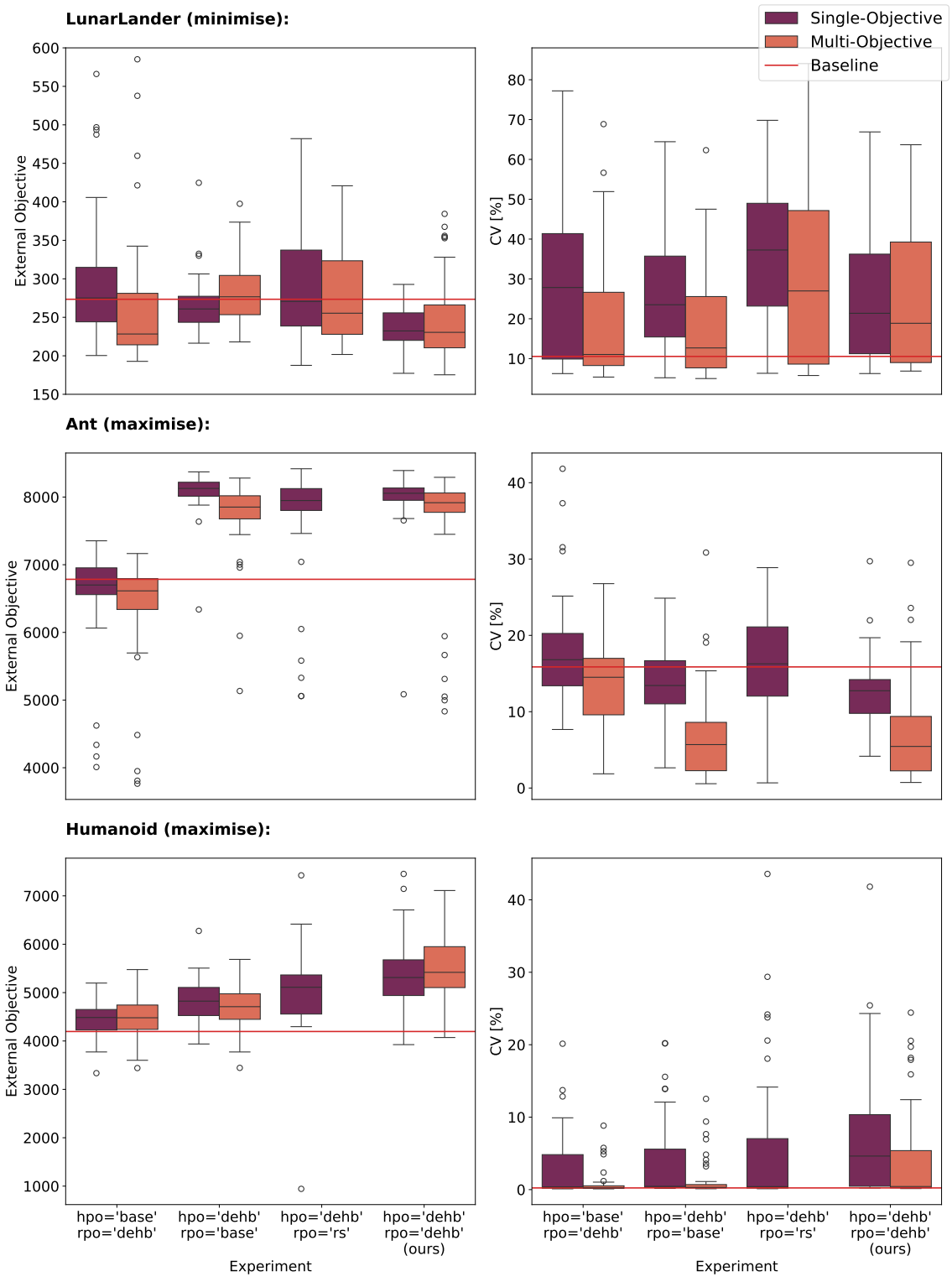


Figure 9: Boxplots for the PPO optimisation experiments over all 50 evaluation trainings of each experiment. We denote by DEHB, rs and base if hyperparameters (hpo) or reward parameters (rpo) were optimised with DEHB, random search or fixed to their baseline values. The red line denotes the baseline performance.

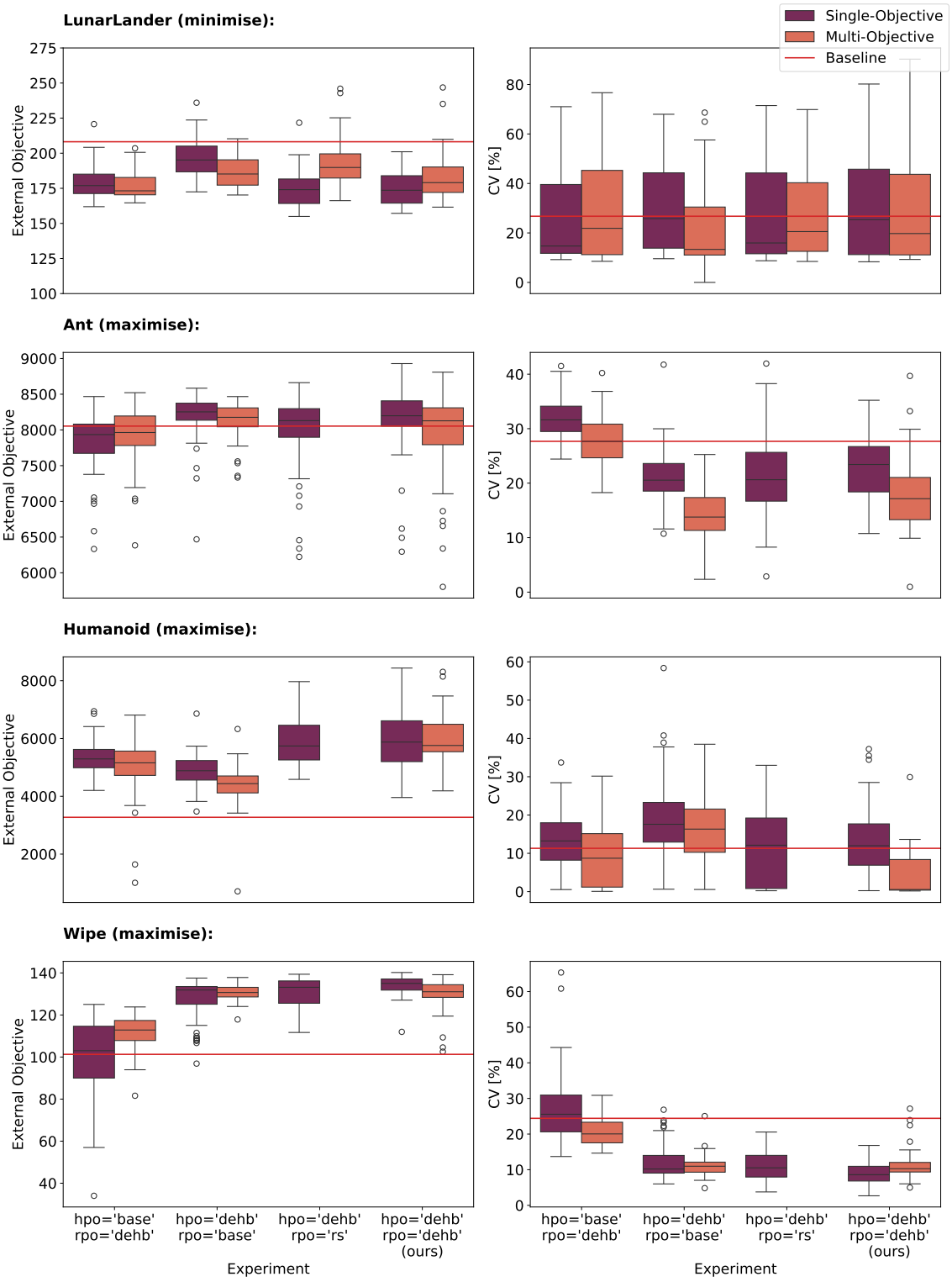


Figure 10: Boxplots for the SAC optimisation experiments over all 50 evaluation trainings of each experiment. We denote by DEHB, rs and base if hyperparameters (hpo) or reward parameters (rpo) were optimised with DEHB, random search or fixed to their baseline values. The red line denotes the baseline performance.

Reward Weight	Explicit Scaling		Implicit Scaling	
	Ant	Humanoid	Ant	Humanoid
w_{dist}	$[0, 10]$		$[0, 25]$	$[0, 63.5]$
$w_{healthy}$	$[0, 10]$		$[0, 25]$	$[0, 63.5]$
w_{force}	$[0, 1]$		$[0, 2.5]$	$[0, 6.35]$
α	$[0, 10]$		1	1

Table 6: The different search spaces of explicit and implicit reward scaling. In the case of explicit reward scaling, the reward weights are normalised to one and scaled by $\|\hat{w}_{ant}\|_1 = 2.5$ and $\|\hat{w}_{humanoid}\|_1 = 6.35$ before applying the reward scale α for Ant and Humanoid, respectively. Thereby, the search spaces in the explicit and implicit scaling have the same upper and lower bounds.

Environment	HPO	RPO	Task Performance	
			Implicit Scaling	Explicit Scaling
Google Brax Ant (maximise)	base	DEHB	7741 (28%) ($p = 2.8 \cdot 10^{-03}$)	**7927 (32%)**
	DEHB		7625 (23%) ($p = 1.0 \cdot 10^{-12}$)	**8199 (23%)**
Google Brax Humanoid (maximise)	base	DEHB	**5124 (12%)** ($p = 4.0 \cdot 10^{-01}$)	**5284 (11%)**
	DEHB		**5846 (12%)** ($p = 8.8 \cdot 10^{-01}$)	**6033 (12%)**

Table 7: Results for the different single-objective optimisation experiments with explicit and implicit reward scaling. Performances in each row without significant statistical differences to the best scaling experiment, as determined by a linear mixed-effects model analysis, are highlighted in bold. We reported the test's p-values of the comparison in each cell.

E Reward Scaling

We examined two alternative methods to perform reward scaling for the Google Brax experiments denoted as explicit and implicit scaling:

Explicit reward scaling aims to disentangle effects between the chosen scaling α and the reward weights w during the optimisation by normalising the reward weights. Formally, the optimiser's selected weights w are normalised by

$$w' = \|\hat{w}\|_1 \cdot \frac{w}{\|w\|_1},$$

where $\hat{w}$ are the default reward weights of the given environment. The resulting reward function $\tilde{r}^{\alpha,w'}$ was then used for RL training as described in Section 5.2. In contrast, *implicit reward scaling* is done by keeping the reward scale fixed as $\alpha = 1$ and instead optimising reward parameters with search spaces multiplied by the upper bound of the explicit scaling. The detailed search spaces are given in Table 6.

In Table 7, we report the results of the two scaling approaches. Explicit scaling statistically significantly outperforms implicit scaling for Ant, whereas, for Humanoid, the median performance is better but not significant. Overall, this hints to explicit scaling being a better choice in cases where scaling matters to the algorithm. Due to its better performance, we used the explicit scaling method for performing experiments in Section 5.2.

F Best performing Hyperparameter and Reward Weight Configurations per Algorithm and Environment

We present the best-performing configurations we found for each training algorithm and environment based on median external objective performance across ten evaluation trainings. Table 8 and Table

Hyperparameter	LunarLander	Ant	Humanoid
learning rate	$3e^{-4}$	0.00112	0.000359
discounting	0.999	0.964	0.962
gae lambda	0.98	0.8378	0.966
clipping rate	0.2	0.276	0.156
entropy coef	0.01	0.345	0.00657
value coef	0.5	0.469	0.35
batch size	64	1024	512
Reward Parameter			
w_{dist}	3.078	1.0	9.28
w_{vel}	0.989	-	-
$w_{tilting}$	0.222		
w_{leg}	4.53		
$w_{healthy}$	-	1.0	2.82
w_{force}		0.5	0.938
α	1.0	1.0	1.0
Performance			
external objective	207 (4%)	8213 (0.1%)	6068 (0.001%)
default shaped return	286 (2%)	8378 (0.2%)	12440 (0.003%)

Table 8: Best performing configurations obtained by our optimisations for PPO training selected by their external objective performance. For each configuration, we report the achieved performance measured by the external objective and the default-shaped reward function with the coefficients of variations in parenthesis.

Hyperparameter	LunarLander	Ant	Humanoid	Wipe
learning rate	0.000998	0.000703	0.00078	0.00031
discounting	0.988	0.982	0.968	0.83
tau	0.0771	0.00444	0.0297	0.00601
batch size	256	512	1024	256
Reward Parameter				
w_{dist}	5.901	2.253	7.61	7.745
w_{dist_th}	-	-	-	2.60
w_{vel}	2.691			0.888
$w_{tilting}$	1.102			-
w_{leg}	5.465			
$w_{healthy}$	-	0.136	2.99	
w_{force}		0.109	0.916	0.030
w_{wiped}		-	-	82.3
$w_{contact}$				0.594
$w_{collision}$				-83.0
α	1.0	7.44	7.40	1.0
Performance				
external objective	160 (6%)	8469 (0.2%)	6583 (0.1%)	137 (5%)
default shaped return	288 (2%)	7903 (0.3%)	12670 (0.07%)	126 (14%)

Table 9: Best performing configurations obtained by our optimisations for SAC training selected by their external objective performance. For each configuration, we report the achieved performance measured by the external objective and the default-shaped reward function with the coefficients of variations in parenthesis.

9 display the parameters for PPO and SAC, respectively, and additionally report the performance on the external objective and default shaped reward. We hope the configurations can help as baselines and for future research. The hyperparameter configuration and reward parameters must be used together in training for each environment to achieve the best performance.

G Execution Environment

All experiments were conducted on a high-performance cluster running the Rocky Linux operating system, release 8.9. The Gymnasium LunarLander and Robosuite Wipe optimisations were executed on CPU nodes, while the Google Brax optimisations utilised GPU nodes.

The CPU-based optimisations were carried out on nodes equipped with Intel Xeon Platinum 8160 2.1 GHz processors, each equipped with 24 cores each and 33 792 KB cache, with approximately 3.75 GB of RAM per core. The GPU-based optimisations utilised NVIDIA Volta 100 GPUs (V100-SXM2) with 16 GB of memory.

During the optimisation process, the LunarLander and Wipe environments, on average, employed 25 and 32 CPU cores in parallel, respectively. For LunarLander, each RL training run used 4 cores, while Wipe training utilised 5 cores per run. During optimisation, the Google Brax environments required an average of 6 parallel GPUs, with each RL training run conducted on a single GPU.

H Linear Mixed Effects Regression Analysis

To thoroughly analyse the differences between our optimisation experiments, we employed a linear mixed effects regression with a Wald test to analyse the difference in performance between experiments. We conducted the test based on the introduction of Brown (2021), using the Wald test as a commonly used approximation of the likelihood-ratio test. The linear mixed-effects model analysis enables us to compare the performance of two optimisation experiments across their respective 50 evaluation runs while accounting for the dependencies induced by the seed of an optimisation run to which an evaluation belongs. Therefore, we can compare the full extent of our data and avoid collapsing it by summarising each optimisation run's performance by the median performance of its 10 evaluation trainings.

For each environment and algorithm, we always pick the best-performing optimisation experiment based on its median performance presented in Table 1. We then compare this optimisation experiment pairwise to the other corresponding optimisation experiments and test whether the performance is statistically significantly different. Hence, the value to be predicted by the linear mixed-effects model is evaluation performance, with the fixed effect being the two compared experiments. The different evaluations are grouped by their corresponding optimisation seed as the model's random effect. Using the Wald test, we then check if removing the fixed experiment effect from the model would substantially harm the prediction performance of the model. Hence, small p-values of the test indicate that the model with the fixed experiment effect provides a better fit, and therefore, the experiments' performances are statistically significantly different. We applied a commonly used significance level of 0.05 to test for significance.

For preprocessing the 100 evaluation data points of two experiments, we normalised mean performance to 0 and standard deviation to 1. Afterwards, we fit a mixed-effects model on the data and remove all points as outliers with residuals deviating more than two times the standard deviation from the mean. We then fit a model on the cleaned data and perform the Wald test to check for significance in the fixed effect, hence the difference between the two experiments.

The assumptions underlying the test are (a) independence of the random effects and (b) homoskedasticity of the residuals of the fitted linear mixed-effects model. The normality of the residuals is an assumption of minor importance, as mixed-effect models have been shown to be robust to violations of this distributional assumption (Schielzeth et al., 2020); Gelman & Hill (2006) even suggest not to test for normality of the residuals. The independence of the optimisation runs as the random effect is ensured by using different random seeds. To check the homoskedasticity assumption, we performed White's Lagrange multiplier test on the residuals. The null hypothesis of White's test is homoskedasticity, and hence, large p-values suggest that the assumption of homoskedasticity is fulfilled. Further, we can reasonably assume that the evaluations of our experiments follow a normal distribution, and we further tested normality of the residuals using the Shapiro-Wilk test.

Environment	HPO	RPO	Test Statistics			
			PPO		SAC	
			Single Obj.	Multi Obj.	Single Obj.	Multi Obj.
Gymnasium LunarLander (minimise)	base	DEHB	$p = 8.1 \cdot 10^{-06}$ $p_{hw} = 0.006$ $\lvert x_{out} \rvert = 4$	**best**	$p = 1.6 \cdot 10^{-01}$ $p_{hw} = 0.526$ $\lvert x_{out} \rvert = 2$	**best**
	DEHB	base	$p = 4.5 \cdot 10^{-08}$ $p_{hw} = 0.623$ $\lvert x_{out} \rvert = 2$	$p = 5.1 \cdot 10^{-06}$ $p_{hw} = 0.485$ $\lvert x_{out} \rvert = 5$	$p = 3.4 \cdot 10^{-15}$ $p_{hw} = 0.185$ $\lvert x_{out} \rvert = 2$	$p = 2.1 \cdot 10^{-05}$ $p_{hw} = 0.498$ $\lvert x_{out} \rvert = 2$
	DEHB	RS	$p = 2.9 \cdot 10^{-07}$ $p_{hw} = 0.106$ $\lvert x_{out} \rvert = 5$	$p = 8.0 \cdot 10^{-04}$ $p_{hw} = 0.254$ $\lvert x_{out} \rvert = 4$	**best**	$p = 1.1 \cdot 10^{-08}$ $p_{hw} = 0.276$ $\lvert x_{out} \rvert = 5$
	DEHB (ours)		**best**	$p = 9.2 \cdot 10^{-01}$ $p_{hw} = 0.994$ $\lvert x_{out} \rvert = 4$	$p = 3.4 \cdot 10^{-01}$ $p_{hw} = 0.612$ $\lvert x_{out} \rvert = 4$	$p = 7.8 \cdot 10^{-02}$ $p_{hw} = 0.612$ $\lvert x_{out} \rvert = 4$
Google Brax Ant (maximise)	base	DEHB	$p = 4.0 \cdot 10{-}243$ $p_{hw} = 0.004$ $\lvert x_{out} \rvert = 5$	$p = 4.0 \cdot 10^{-80}$ $p_{hw} = 0.937$ $\lvert x_{out} \rvert = 9$	$p = 1.8 \cdot 10^{-08}$ $p_{hw} = 0.079$ $\lvert x_{out} \rvert = 6$	$p = 3.4 \cdot 10^{-03}$ $p_{hw} = 0.191$ $\lvert x_{out} \rvert = 6$
	DEHB	base	**best**	$p = 9.3 \cdot 10^{-03}$ $p_{hw} = 0.056$ $\lvert x_{out} \rvert = 8$	**best**	**best**
	DEHB	RS	$p = 6.8 \cdot 10^{-04}$ $p_{hw} = 0.051$ $\lvert x_{out} \rvert = 6$	-	$p = 7.7 \cdot 10^{-02}$ $p_{hw} = 0.109$ $\lvert x_{out} \rvert = 5$	-
	DEHB (ours)		$p = 1.7 \cdot 10^{-02}$ $p_{hw} = 0.894$ $\lvert x_{out} \rvert = 2$	**best**	$p = 7.7 \cdot 10^{-01}$ $p_{hw} = 0.978$ $\lvert x_{out} \rvert = 6$	$p = 6.4 \cdot 10^{-01}$ $p_{hw} = 0.170$ $\lvert x_{out} \rvert = 5$
Google Brax Humanoid (maximise)	base	DEHB	$p = 1.1 \cdot 10^{-27}$ $p_{hw} = 0.123$ $\lvert x_{out} \rvert = 7$	$p = 2.2 \cdot 10^{-26}$ $p_{hw} = 0.797$ $\lvert x_{out} \rvert = 6$	$p = 1.0 \cdot 10^{-04}$ $p_{hw} = 0.293$ $\lvert x_{out} \rvert = 6$	$p = 3.5 \cdot 10^{-06}$ $p_{hw} = 0.773$ $\lvert x_{out} \rvert = 4$
	DEHB	base	$p = 3.8 \cdot 10^{-10}$ $p_{hw} = 0.086$ $\lvert x_{out} \rvert = 7$	$p = 4.8 \cdot 10^{-23}$ $p_{hw} = 0.780$ $\lvert x_{out} \rvert = 8$	$p = 1.6 \cdot 10^{-14}$ $p_{hw} = 0.025$ $\lvert x_{out} \rvert = 7$	$p = 3.1 \cdot 10^{-51}$ $p_{hw} = 0.018$ $\lvert x_{out} \rvert = 6$
	DEHB	RS	$p = 1.5 \cdot 10^{-02}$ $p_{hw} = 0.094$ $\lvert x_{out} \rvert = 4$	-	$p = 9.9 \cdot 10^{-01}$ $p_{hw} = 0.132$ $\lvert x_{out} \rvert = 5$	-
	DEHB (ours)		**best**	**best**	**best**	**best**
Robosuite Wipe (maximise)	base	DEHB	-		$p = 2.1 \cdot 10^{-45}$ $p_{hw} = 0.001$ $\lvert x_{out} \rvert = 3$	$p = 3.5 \cdot 10^{-82}$ $p_{hw} = 0.003$ $\lvert x_{out} \rvert = 5$
	DEHB	base			$p = 4.8 \cdot 10^{-13}$ $p_{hw} = 0.278$ $\lvert x_{out} \rvert = 3$	**best**
	DEHB	RS			$p = 4.4 \cdot 10^{-06}$ $p_{hw} = 0.022$ $\lvert x_{out} \rvert = 5$	-
	DEHB (ours)				**best**	$p = 3 \cdot 10^{-1}$ $p_{hw} = 0.296$ $\lvert x_{out} \rvert = 4$

Table 10: We show the p-values for significance in the difference of each experiment to the best-performing optimisation corresponding to the results presented in Table 1. Additionally, p_{wh} gives the p-values of White's test for heteroskedasticity and $|x_{out}|$ the number of removed outliers from the 100 data points for each comparison. We require homoskedasticity of the residuals and hence values of $p_{wh} > 0.05$. Highlighted cells do not perform significantly differently from the best corresponding optimisation experiment.

In Table 10, we present the p-values of the Wald and White tests, as well as the number of outliers removed for the pairwise comparisons detailed in Table 1. The values correspond to comparisons against the cell identified as the best performer. Entries that do not exhibit a significant difference from the best-performing cells are highlighted. The assumption of homoskedasticity is satisfied in the majority of cases, as indicated by p-values greater than 0.05. This is particularly true in scenarios where the performance of the experiment is not significantly different from that of the best

optimisation. Moreover, the p-values from the Wald test are generally well above the significance threshold of 0.05 when there is no significant difference. In our comparisons, the normality assumption was met in 18 out of 35 cases. Given the mixed-effect models' resilience to deviations from the normality assumption and the large number of cases where this assumption was satisfied, we conclude that our test results are reliable for comparing the outcomes of our experiments.