Dynamic Context Scheduling: Learning Beyond the Static Universe

Martin Mráz
University of Freiburg
mrazm@cs.uni-freiburg.de

André Biedenkapp
University of Freiburg
biedenka@cs.uni-freiburg.de

Abstract

We study *dynamic context scheduling* as a training instrument for contextual reinforcement learning. Rather than treating intra-episode context variation as a deployment reality, we treat it as a controlled shaping mechanism. Thereby, context evolves within each training episode according to a predetermined schedule, exposing the policy to a richer and more temporally structured region of the environment parameter space. We introduce DYNAMICCARLENV, a framework that wraps contextual environments with pluggable schedule families, such as sinusoidal offsets or cosine annealing. Across CartPole, BipedalWalker and VehicleRacing with CARL contextualization, we show that dynamic schedules match or outperform static context baselines in the out-of-distribution (OOD) regimes. Interestingly, for the more complex BipedalWalker and VehicleRacing environments we also achieve higher in-distribution (ID) evaluation performance. Preliminary findings indicate that automatic search for multi-stage curricula can successfully discover schedules that improve generalization, performing comparably to extensive grid search over single-stage schedulers.

1 Introduction

Reinforcement learning (RL) achieves strong performance in simulation, yet policies trained under narrow, stationary conditions often fail when environment parameters shift at deployment. Small changes in friction, payload, actuator gain, or gravity can cause large drops in return [see, e.g., Tobin et al., 2017, Zhou et al., 2019, Ding et al., 2020, Zhang et al., 2021, Cobbe et al., 2020, Wang et al., 2021, Benjamins et al., 2023, Kirk et al., 2023, Gumbsch et al., 2024, Prasanna et al., 2024, Suau et al., 2024, Iannotta et al., 2025]. This brittleness reflects a clear mismatch between the single-dynamics regime encountered during training and the potentially different dynamics faced at test-time.

Several established paradigms address this issue. Predominantly, they aim to expose agents to a broader variety of experiences during training, such that transfer to novel experiences does not pose such a drastic, and potentially catastrophic shift at test-time. *Domain randomization* (DR) improves robustness by exposing the agent to many environment instances [Tobin et al., 2017, Packer et al., 2019] in an unstructured manner. DR typically samples related environments from a distribution without explicitly informing learning agents about the changes in environments. Cobbe et al. [2020] proposed a suite of environments that leverage *procedural content generation* (PCG) to vary level structure and visuals of video games. This broadly enables learning of behaviors that are robust to changes in an environment but is highly dependent on the choice of distribution. A too broad distribution might even cause agents to unlearn desirable behavior as sampled environments might require diametrically opposed solutions. Thus, DR and PCG are often coupled with *curriculum learning* techniques to guide learning to more and more complex scenarios [OpenAI et al., 2019, Klink et al., 2020]. *Robust RL* optimizes worst-case objectives [Pinto et al., 2017]. For example, in real-world systems noisy sensor readings are to be expected [Zhang et al., 2020]. To increase robustness of RL policies the robustness objective is typically modeled as a max-min problem. In this

setting the goal is to learn a policy that maximizes the reward under the worst possible adversarial setting [Panaganti et al., 2022]. This style of learning can mitigate worst-case outcomes but largely sacrifices performance in average or best case scenarios as the learned policies act conservatively. *Meta-RL* enables online adaptation to new environments [see, e.g., Duan et al., 2016, Finn et al., 2017, Rakelly et al., 2019, Melo, 2022, Grigsby et al., 2024, Beck et al., 2025, Shala et al., 2025]. Such approaches, however, often require complex architectures and expensive bi-level optimization. Furthermore, Meta-RL often builds on system identification approaches [Yu et al., 2017, Zhou et al., 2019, Evans et al., 2022]. Thereby agents attempt to estimate or recognize environment dynamics from a history of observations. While enabling online adaptation, such approaches require further environment interactions at deployment to continue to learn [Beck et al., 2025, Grooten et al., 2026].

Counter to the prior examples, *Contextual RL* (cRL) aims to explicitly provide agents with the knowledge of how environments are related to each other. To this end, cRL assumes that transitions and rewards depend on explicit context variables c, and the agent learns a policy conditioned on the current context [Hallak et al., 2015, Modi et al., 2018, Benjamins et al., 2023, Zhou et al., 2026]. While naïvely treating context as another observable can aide in learning more general behavior, more dedicated architectures have been explored in which context is injected into the latent-representations [Beukman et al., 2023, Prasanna et al., 2024, Benad et al., 2025, Engwegen et al., 2025].

A key assumption shared by most approaches listed above is that context is treated as a monolithic, static element [Biedenkapp, 2026]. The corresponding context c is sampled once at the beginning of an episode and held fixed until the episode ends. This design choice is convenient for training stability and credit assignment, but it is also highly restrictive. Real-world physics do not reset between timesteps; payloads shift, actuators fatigue, and terrain variations unfold continuously. More importantly for training, per-episode static sampling discards a rich source of structured signal, i.e., the temporal variation of context *within* an episode.

We propose to exploit intra-episode context variation not as a model of deployment conditions but as a *training instrument*. Concretely, we define a family of *dynamic context schedules* that govern how the context c evolves within each training episode, including sinusoidal sweeps, piecewise constant regimes, linear drift, random walks, cosine annealing, and hybrid compositions, and study their effect on policy robustness and generalization. However, our evaluation protocol always considers policies under static context. Thus, the temporal variation serves the training-time purposes (i) *regularization*; (ii) and *exploration*. The former prevents policies from overfitting while the latter ensures that agents experience a broader, temporally structured set of state-context pairs. This training strategy is related to the *dynamic contextual MDP* (dcMDP) formalism of Tennenholtz et al. [2023], which allows context to evolve exogenously via a process $\Omega(c_{t+1} \mid c_t)$. Here, we instantiate Ω with deterministic, parametric schedules rather than a learned or stochastic model, keeping the framework simple and interpretable while retaining full cMDP semantics. Thereby, context remains outside the agent's control, transitions and rewards depend on c_t, and the agent may or may not observe c_t.

Our work provides the following contributions:

1. We provide a novel training paradigm for (contextual) reinforcement learning to facilitate better generalizability of learned policies;
2. We empirically evaluate a broad suite of scheduling families to study how and which dynamic context changes facilitate better generalization;
3. We present an open-source extension of the CARL [Benjamins et al., 2023] benchmark to facilitate easy use of dynamic context schedules;
4. We conduct a state-space coverage analysis, revealing the counter-intuitive result that dynamic schedules improve generalization without expanding the agent's state-space footprint.

2 Related Work

A contextual Markov Decision Process (cMDP) [Hallak et al., 2015] augments a standard MDP [Bellman, 1957] with the notion of context. An MDP $M = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \rho)$ entails a state space $\mathcal{S}$, an action space $\mathcal{A}$, transition dynamics $\mathcal{T} \colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, a reward function $\mathcal{R} \colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and an initial state distribution ρ. Contextual MDPs introduce the notion of context $c \in \mathcal{C}$ that parametrizes transition function $\mathcal{T}_c$, the reward function $\mathcal{R}_c$ as well as the initial state distribution ρ_c while leaving the state and action spaces unchanged. Context spaces can be either discrete or described by a distribution $p_{\mathcal{C}}$ [Benjamins et al., 2023]. Consequently, a cMDP $\mathcal{M}$ represents a

family of related MDPs $\mathcal{M} = \{M_c\}_{c \sim \mathcal{C}}$ and can be seen as a sub-class of partially observable MDPs [Kirk et al., 2023]. Tennenholtz et al. [2023] presents a special case of cMDPs where contexts are history-dependent and are allowed to evolve over time, which they dubbed dynamic cMDPs (dcMDPs). Recently Biedenkapp [2026] proposes a novel taxonomy of context that distinguishes between allogenic (environment-imposed) and autogenic (agent-driven) contexts. They further discuss how context might evolve over time. The notion of autogenic context thus relates to the dcMDP setting, as autogenic contexts may be directly influenced by an agent's behaviour (e.g. battery power), whereas allogenic context is independent of an agent's decisions.

Kirk et al. [2023] highlight the utility of the cMDP setting for assessing the zero-shot generalizability of learned policies and propose a novel evaluation protocol. Benjamins et al. [2023] implement this protocol in their study on the effect of context on training various deep RL agents on their novel CARL benchmark. CARL extends common RL benchmarks and environments [e.g., Brockman et al., 2016, Tassa et al., 2018, Freeman et al., 2021] with physical contexts, such as gravity, friction or masses of robots. Their study shows that agents that are simply trained on a distribution of contexts without having explicit access to the true context value tend to learn robust behavior but do not necessarily solve every environment optimally. On the other hand, context aware agents that naïvely concatenate the context to the state-observation, might be able to perfectly adapt to the changes in environments but may require changes to the RL pipeline (such as choice of hyperparameters [Eimer et al., 2021]) to be able to do so. Beyond naïve concatenation, multiple works explore how to employ hypernetworks to facilitate better adaptability of learned policy by learning adapter modules or the weights of a policy directly [Beukman et al., 2023, Benad et al., 2025, Engwegen et al., 2025]. Opposite to these lines of work, Prasanna et al. [2024], Gumbsch et al. [2024] try to exploit contextual information by injecting contextual information into latent representations of a world model.

Most commonly in contextual RL research, context is treated as a monolithic, static quantity and assumed to be mostly static throughout an episode [Biedenkapp, 2026]. To the best of our knowledge, few works explore dynamic changes of context. Gumbsch et al. [2024] for example, aims to learn when shifts in context, such as the opening or closing of a door, occur. Whenever works aim to learn to estimate context on-the-fly, however, policies might operate under the assumption that context changes between states [Kumar et al., 2021, Ren et al., 2023, Ndir et al., 2024]. Relatedly, Chandak et al. [2020] aims to learn policies that are not only working well with the context, but simultaneously aim to estimate how the MDP changes between episodes, such that the policy is setup well for solving this future task as well. Counter to these approaches, our work proposes to leverage the fact that most of the training occurs in simulation and that it is possible to explicitly adapt context throughout an episode with the goal to push the generalization capabilities of RL agents.

3 Dynamic Context Scheduling

We introduce a training framework that replaces the standard static per-episode context with a parametric intra-episode schedule. The context c_t evolves according to a chosen schedule family throughout each training episode, while evaluation always uses a predefined set of static contexts so that the policy is judged on its generalization to fixed dynamics, not on its ability to track change.

Dynamic Contextual Environments We open-source our DYNAMICCARLENV as a lightweight wrapper that sits on top of any CARL environment [Benjamins et al., 2023]. We intercept each environment step to (i) advance a parametric context schedule, (ii) pass the updated c_t to the underlying simulator, and (iii) deliver the chosen context signal to the policy observation. The transition $s_t \xrightarrow{a_t, c_t} s_{t+1}$ is therefore governed by the live scheduled context rather than the static episode draw. Context is read from and written to the simulator via user-supplied getter/setter callables, which isolates the scheduling logic from environment internals and makes the wrapper applicable to any CARL-compatible physics backend.

We study two context observability modes (Table 1). *Live* mode allows the context to evolve within an episode and can be explicitly observed by an agent, similar to a dcMDP transition model. *None* mode recovers domain randomization with dynamic changes within an episode. However *None* does

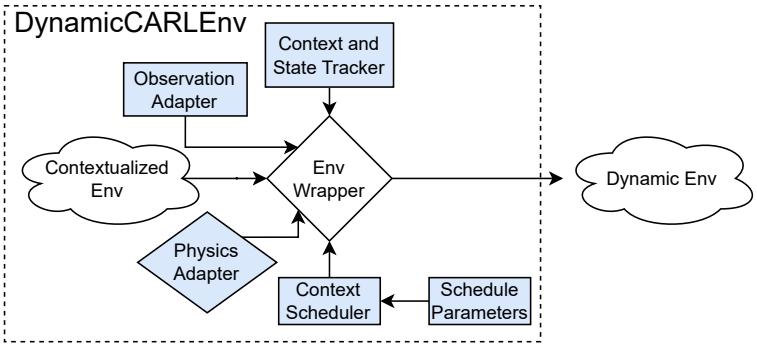


Figure 1: Dynamic context wrapper around CARLEnv. Blue components are user-controlled. The *Context Scheduler* produces the per-step context c_t given the *Schedule Parameters* (e.g., amplitude, period, drift rate, dwell times, change points). The wrapper applies c_t through a *Physics Adapter*, controls what the policy observes through an *Observation Adapter* (none, c_0, or c_t), and logs $(s_t, a_t, r_{t+1}, c_t, c_{\text{vis}})$ via the *Context and State Tracker*. The base contextualized environment still receives c_0 at reset.

Table 1: Context observability modes.

Mode	Observation	Policy input
None	s_t	State only; dynamics change silently.
Live	$[s_t,\ c_t^*]$	State concatenated with current context. *Context normalized; see text.

not make the context observable and lets us isolate the effect of temporal structure from explicit context conditioning.[1]

When context is observable by an agent, we normalize c_t to lie in $[-1, +1]$ before appending it to the state. We motivate this choice as some context might be obtainable with special sensors. Such sensors would need to be calibrated and provide sensor-limit bounds. Crucially, these bounds are set to a physically plausible range that extends *beyond* the evaluation contexts, rather than being fit to the training or evaluation splits. This ensures that OOD contexts remain well within the normalized range at test time, so the agent is never exposed to saturated inputs. Further, this ensures that the evaluation grid is not inadvertently encoded into the policy's input representation.

Schedule Families We implement six canonical schedule families, each inducing qualitatively different temporal structure on the context trajectory within an episode. Three additional composite families (Sinusoidal Jump, Ornstein–Uhlenbeck, Phased OU) are described in Appendix A.

- **Sinusoidal.** $c_t = c_0 + A\sin(\omega t)$, where c_0 is the episode-initial context drawn from the training pool, A is the amplitude, and $\omega = 2\pi/T$ the angular frequency. The direction is randomized per episode; values are reflected at context bounds. Repeatedly sweeps the policy across a wide context range within a single episode.
- **Cosine Annealing.** $c_t = c_{\text{end}} + \frac{1}{2}(c_{\text{start}} - c_{\text{end}})(1 + \cos(\pi t/T_0))$, with optional periodic restarts. Produces a smooth monotone drift per cycle; with restarts the policy must repeatedly re-adapt from diverse starting points, inspired by [Loshchilov and Hutter, 2017].
- **Continuous Incrementer.** $c_{t+1} = c_t \pm \delta$, reflected at bounds. The drift direction is fixed per episode (randomized at reset). Requires generalization across the full context range as a single monotone sweep per episode.
- **Piecewise Constant.** Context is held fixed within segments of random duration and jumps abruptly to a new value sampled from a discrete set at each change point. Simulates discrete regime shifts (e.g., sudden payload drops or actuator failure).
- **Random Walk.** $c_{t+1} = \text{clip}(c_t + \epsilon_t,\ c_{\text{min}},\ c_{\text{max}}), \epsilon_t \sim \mathcal{N}(0, \sigma^2)$. Bounded stochastic drift; tests robustness under slow, persistent perturbations.

[1]Note, in settings where only static contexts have been considered, *None* is commonly referred to as "hidden" and live as "concatenation" or "naïve".

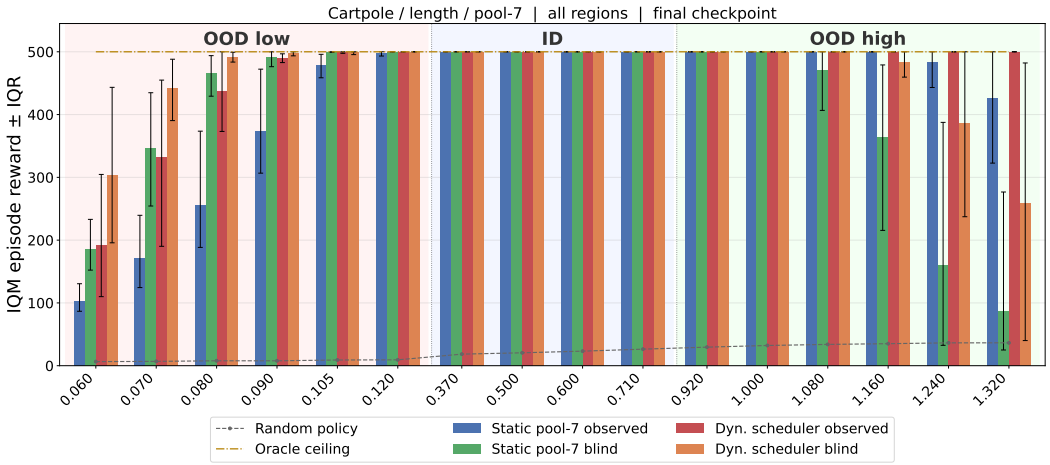


Figure 2: IQM episode reward (bars: Q1–Q3) at the final checkpoint across all 16 evaluation contexts for CartPole. Dashed lines show the random and oracle policy baselines. For dynamic schedulers we show the two best representatives by combined IQM: one observing only states, one also observing the context value.

- **Stochastic Jumps.** Large discrete displacements of magnitude $\Delta \sim \mathrm{Uniform}(m_{\mathrm{lo}}, m_{\mathrm{hi}})$ occur at random timesteps, with direction sampled independently. Unlike piecewise constant, the magnitude and timing are fully random, producing heavy-tailed context trajectories.

Composite families can be formed by combining some of the above. For example, a *sinusoidal jump* layers a sinusoidal oscillation on top of an intermittently jumping baseline; a *Levy walk* combines stochastic jumps with additive Gaussian noise to obtain heavy-tailed drift [Zaburdaev et al., 2015]. Amplitude and step-size parameters are expressed as fractions of the full context range so that a single hyperparameter setting transfers across context variables with different physical scales.

Appendix A shows trajectory examples for all schedule families included in our experiments.

Context Pool Design Each training episode draws its initial context c_0 from a finite *context pool*. We construct pools hierarchically by recursive midpoint insertion: *Pool-1* contains the single midpoint of the context range; *Pool-3* adds the two range extremes; *Pool-5* inserts midpoints between each adjacent pair; and so on. This construction keeps the global range constant across pool sizes, isolating the effect of context density from boundary effects. The episode-initial context is resampled from the pool at the start of each episode, providing inter-episode diversity independently of intra-episode schedule variation.

Multi-Stage Curriculum Scheduling A single schedule family applied for the full training run may not be optimal. In the early phases the agent might benefit from aggressive exploration to seed the policy with a broad behavioral repertoire, while late training calls for a more focused refinement pass. We therefore support *multi-stage* schedules that divide training into K fixed-length stages, each governed by an independently configured scheduler from any of the families above. At each stage boundary the scheduler is re-initialized without interrupting policy training, allowing exploration pressure to be varied throughout the learning lifecycle.

4 Experiments

We empirically investigate whether dynamic context schedules improve policy robustness and generalization compared to static context training. Our core ablation on CartPole systematically compares schedule families, pool sizes, and observability modes. We then validate the findings on the more demanding BipedalWalker and VehicleRacing environments. We additionally analyze the state-space coverage induced by different schedules or other multi-context configurations and explore automated multi-stage curriculum search via Optuna [Akiba et al., 2019].

4.1 Experimental Setup

Environments. We empirically evaluate the impact of dynamic context variation during training on three contextualized environments from CARL [Benjamins et al., 2023]. We use our dynamic context scheduler as introduced in Section 3. We thoroughly focus our experiment on CartPole, varying *pole length* as the context. This axis is challenging at both OOD extremes: short poles and long poles each present distinct failure modes (see Figure 2), unlike contexts such as gravity or force magnitude where only one extreme is typically difficult. Its fast simulation enables a thorough ablation across scheduler families and configurations. We further validate the methodology on CARL BipedalWalker, where we vary a single *payload x-axis offset* by attaching a rigid payload to the torso (see Figure 6a). Finally, we extend the evaluation to CarRacing, a vision-based environment with pixel observations and more complex dynamics, testing whether the methodology scales to image-based settings. CarRacing is derived from CARL VehicleRacing: we fix the vehicle type to a single car and replace the original discrete vehicle-type context with a continuous payload offset to support our scheduling framework, hence the rename. We control the offset along the car's longitudinal and lateral axes (see Figure 6b).[2].

Evaluation protocol. Following Kirk et al. [2023], Benjamins et al. [2023], we split evaluation contexts into three regimes: *in-distribution* (ID, within the training context pool boundaries), *OOD-low* (below the minimum training context value), and *OOD-high* (above the maximum training context value). Full context ranges and eval grids are listed in Appendix C. All agents are trained with PPO [Schulman et al., 2017] as implemented in Stable-Baselines3 [Raffin et al., 2021] and evaluated deterministically over 30 episodes per evaluation context at the final training checkpoint. Following Agarwal et al. [2021], we report the interquartile mean (IQM) of episode rewards aggregated across evaluation contexts, which provides a robust estimate of central tendency in the presence of outlier seeds. Due to computational restrictions we report experiments using 10 seeds for CartPole, 8 for BipedalWalker and 5 for the CarRacing experiments. Full hyperparameter settings are provided in Appendix B. Finally, when context is observable by the agent, we normalize the context input to stabilize PPO training across both static and dynamic conditions (see Appendix D.3.6).

4.2 Scheduler Search Results

Table 2 reports the combined IQM at the last training checkpoint for all three environments. We opt to report final scores throughout to preserve evaluation fairness: selecting the best-achieved checkpoint would require knowledge of generalisation performance during training, which is information we do not assume access to in practice. Best-anytime results are reported in Appendix D for reference.

Across all environments and observability modes, the best dynamic scheduler outperforms its static counterpart (Table 2). The gains are modest but consistent for CartPole ($+39$ IQM observed, $+53$ blind), where the ceiling effect of the task limits headroom. Walker shows the most dramatic improvement under the observed condition ($+120$ IQM). The static observed baseline collapses to an IQM of only 31.0, likely due to overfitting to the three fixed training contexts when the context value is directly visible, while dynamic schedules force broader generalization. CarRacing presents the sharpest contrast between observability modes as the best blind dynamic scheduler reaches 592.3 ($+489$ IQM over static blind), while both observed conditions result in negative IQM, representing failed policies. We report the lateral axis (COM_Y) as the primary CarRacing result, as it was

[2]We provide an interactive notebook in which users can drive the car themselves under both static and dynamic payload contexts; see `https://github.com/mrazmartin/dynamicCARL`

Table 2: Combined IQM score at the **last checkpoint**, computed from the per-seed `avg_combined` metric (average over all ID and OOD contexts). Values: IQM [Q1, Q3]. Bold marks the best scheduler per environment.

Condition	CartPole	Walker	CarRacing (lateral)
Static observed	429.0 $[401.5,\ 451.4]$	31.0 $[-34.9,\ 63.8]$	-85.9 $[-93.1,\ -76.2]$
Static blind	410.0 $[383.1,\ 436.5]$	91.9 $[60.6,\ 113.0]$	103.6 $[-59.0,\ 378.9]$
Best sched. observed	$\mathbf{468.2}$ $[454.7,\ 484.3]$	$\mathbf{151.4}$ $[122.2,\ 185.3]$	-35.2 $[-67.3,\ -7.4]$
Best sched. blind	462.9 $[430.9,\ 494.1]$	131.1 $[94.9,\ 167.6]$	$\mathbf{592.3}$ $[549.1,\ 625.7]$

Table 3: Multi-stage Optuna retrain results at the **last checkpoint**. Walker values are evaluated at the matched 1.5 M-step budget (half the full single-scheduler training run); single-scheduler baselines are re-evaluated at the same budget for a fair comparison. Values: IQM [Q1, Q3]. Top-3 retrained trials shown per environment.

Condition	CartPole	Walker (1.5 M)
Static observed	429.0 [401.5, 451.4]	77.6 [54.2, 107.8]
Static blind	410.0 [383.1, 436.5]	108.0 [72.3, 127.5]
Best sched. observed	**468.2** [454.7, 484.3]	119.3 [95.7, 136.6]
Best sched. blind	462.9 [430.9, 494.1]	114.8 [88.6, 131.9]
Optuna #1 (T168 / T118)	459.3 [445.9, 480.9]	**125.0** [72.5, 161.7]
Optuna #2 (T218 / T117)	459.3 [441.1, 470.0]	115.6 [96.0, 157.4]
Optuna #3 (T145 / T066)	455.5 [427.1, 483.5]	104.1 [55.9, 124.7]

the main focus of our scheduler search; longitudinal (COM_X) results are provided in Table 17. The best-anytime results in Appendix D.2 show that observed policies do learn during training but ultimately degrade, suggesting late-stage instability rather than a fundamental inability to train on this environment. This reversal, where blind outperforms observed by a wide margin in CarRacing but not in simpler environments, is consistent with the context-observability analysis in Appendix D.1.

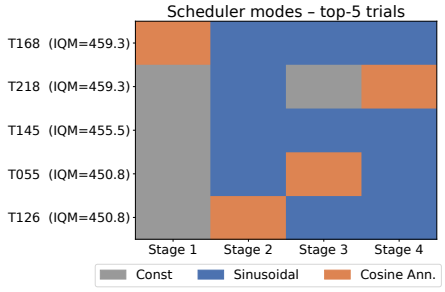
Figure 2 shows per-context IQM episode return for CartPole across all 16 evaluation contexts, separated out into OOD-low, ID, and OOD-high regions. The regional breakdown reveals that aggregate IQM improvements can mask meaningfully different dynamics across regimes. In the ID and OOD-low regions, dynamic and static schedulers perform comparably, with dynamic schedulers providing modest but consistent gains. The OOD-high region tells a different story. Static baselines tend to exhibit a characteristic generalization drop around the midpoint of training (see Figure 37 in the Appendix), where OOD-high performance peaks early before degrading and struggles to recover as the policy specializes to the training distribution. Dynamic schedulers substantially limit this effect. Continuously varying the context during training prevents a policy from overfitting to a fixed regime, so several families either never exhibit the drop or recover more quickly. Looking at only the best-anytime scores would obscure the degradation that static training accumulates over time. Analogous regional figures for Walker and CarRacing appear in Appendix D; these show the same behaviour to be much less pronounced. Results per scheduler families are detailed in Appendix F.

4.3 Multi-Stage Schedulers

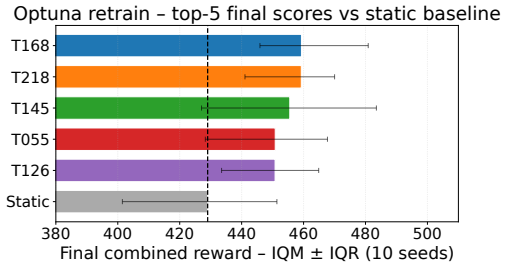
We use Optuna [Akiba et al., 2019] with a TPE sampler to automatically search for effective multi-stage schedule curricula on CartPole and BipedalWalker. Each trial independently selects a scheduler mode from {constant, sinusoidal, cosine annealing} and its associated parameters for each stage. The trial objective is the mean across seeds of each seed's best combined evaluation score observed at any stage boundary during training. For CartPole we search over $K = 4$ stages (30 k / 30 k / 45 k / 45 k steps, 150 k total) on the pool-7 distribution across 240 trials with 4 seeds each; the stage lengths were chosen to align with the OOD-high generalization loss pattern observed in the single-stage experiments, where performance peaks early and degrades thereafter. For BipedalWalker we use $K = 3$ equal stages (500 k each, 1.5 M total) on the pool-3 distribution across 120 trials with 3 seeds each; the 1.5 M budget is half the full single-stage run to keep search tractable, and $K = 3$ equal stages was a pragmatic choice to keep the search space manageable.

The top-20 CartPole configurations are retrained with 10 seeds to obtain reliable IQM estimates. For BipedalWalker, 16 top configurations that clearly outperformed the static baselines during the 3-seed search are retrained with 8 seeds each. All comparisons are at the last training checkpoint; single-scheduler baselines are evaluated at the same training budget to ensure a fair comparison.

Table 3 summarises the last-checkpoint IQM for the top-3 retrained trials alongside the static and best single-stage baselines; best-anytime results are in Appendix E.1. On both environments the top retrained configurations largely outperform the static baseline. However, the best multi-stage trials do not consistently exceed the best single-stage dynamic schedulers found by the grid search over the full set of schedule families.



(a) Scheduler mode per stage for the top-5 retrained CartPole trials, sorted by retrain IQM. Grey: constant; blue: sinusoidal; orange: cosine annealing.



(b) Final-checkpoint IQM $\pm$ IQR for the top-5 retrained CartPole trials (10 seeds). The dashed line marks the static pool-7 baseline IQM.

Figure 3: CartPole Optuna multi-stage retrain results.

This comparison comes with important caveats: the Optuna search is restricted to two active scheduler families (sinusoidal and cosine annealing) and the context-observed mode only, whereas the single-stage grid search covers the full diversity of families and both modes. The 3–4 seeds used to score each trial during search are insufficient to reliably rank configurations: after retraining with more seeds, rankings shift substantially and some configurations that appeared promising fall below the static baseline (Figures 28 and 31). Given these constraints, we cannot conclude whether the multi-stage structure itself is beneficial beyond what a single well-chosen stage already provides.

Nevertheless, inspecting the top CartPole configurations (Figure 3a) reveals a consistent structural tendency: the first stage is often idle while an active stage follows later, suggesting a *warm-up then diversify* pattern. Sinusoidal schedules dominate the active stages, and the number of active stages is essentially uncorrelated with final performance, indicating that a single well-placed active stage captures most of the benefit. Equivalent plots for BipedalWalker are provided in Appendix E.3 (Figures 29a and 29b).

The total number of Optuna trials is comparable to the size of the single-stage grid search over schedule families and their hyperparameters, so both searches operate on a roughly equal compute budget. For BipedalWalker, the per-trial training cost further restricted the search to a subset of the context range and a training run of 1.5 M steps rather than the full 3 M used in the single-stage ablation, which additionally limits what can be concluded from those results specifically.

4.4 State Space Coverage

A natural hypothesis is that dynamic schedulers improve OOD robustness by steering the agent through a broader region of the combined state–context space during training. We test this with a discretised coverage diagnostic on both CartPole and BipedalWalker.

For CartPole, we run a dedicated coverage diagnostic, pooling pool-3 and pool-7 conditions across 20 seeds each, and discretise the four observation dimensions into 12 bins each ($12^4 = 20{,}736$ hypercells), measuring the fraction of cells visited over 150 k training steps. Figure 4 shows final IQM score and 4D coverage per scheduler. While evaluation scores vary visibly across conditions, with Lévy walk and sudden jump outperforming the static baseline, coverage does not follow the same pattern: all conditions cluster between 6.54% and 6.83%. A one-way Analysis of Variance (ANOVA) calculated via SciPy [Virtanen et al., 2020] confirmed that these coverage differences are not statistically significant ($p > 0.05$). This holds even for the parallel condition that doubles the effective sampling rate, ruling out that raw state throughput is the missing ingredient (see Appendix D.3.4 for the full parallel vs. sequential analysis). We repeat the analysis on BipedalWalker using four independent projections of the 24-dimensional observation space (joint angles, posture, and per-leg phase portraits) and find the same null result across all projections (ANOVA $p > 0.05$ in every case; Appendix D.3.5).

Taken together, these results suggest that dynamic scheduling improves robustness by *restructuring* the training signal within the visited state space, i.e., exposing the policy to richer temporal sequences

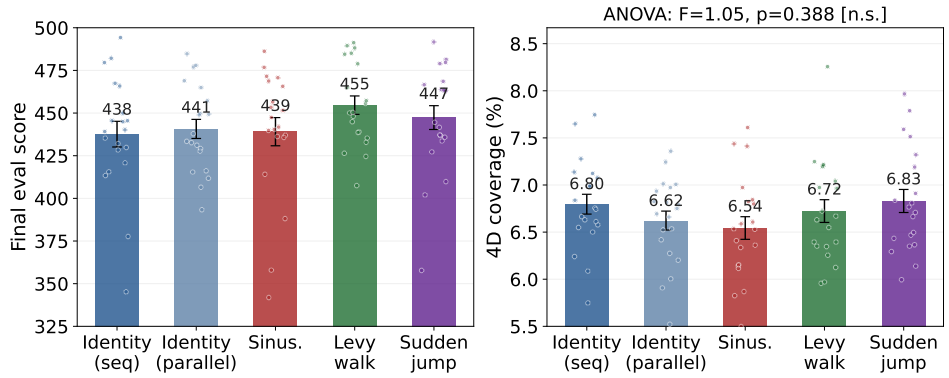


Figure 4: IQM score (left) and 4D state-space coverage (right) per scheduler for CartPole pole-length at pool sizes 3 and 7 (20 seeds each). Points show individual seeds; bars show mean ± SEM. ANOVA on coverage: $p > 0.05$.

of (s_t, c_t) pairs within each episode rather than by expanding the set of states visited. The pool-size scaling study in Appendix D.3.3 further shows that coverage is flat regardless of how many training contexts are used, and that virtually all cells are discovered within the first 50 k training steps for any scheduler, after which the policy converges to a narrow behavioral manifold.

4.5 Experiments Summary

Across CartPole, BipedalWalker, and CarRacing, dynamic context schedules consistently match or outperform static context baselines, with the largest gains on BipedalWalker (observed) and CarRacing (blind). Multi-stage curriculum search via Optuna reliably improves over the static baseline but does not consistently exceed the best single-stage scheduler found by grid search. State-space coverage analysis on CartPole and BipedalWalker shows that dynamic and static schedulers visit indistinguishable fractions of the observation space; the benefit of dynamic scheduling therefore derives from the temporal structure of the training signal rather than from broader state exploration.

5 Discussion and Conclusion

We proposed dynamic context scheduling as a training instrument to improve zero-shot generalization and out-of-distribution (OOD) robustness in reinforcement learning. By continuously evolving physical parameters within training episodes, agents learn behaviors resilient to temporal shifts, consistently matching or outperforming static baselines across CartPole, BipedalWalker, and CarRacing. Crucially, our coverage analysis revealed that these gains do not stem from broader state-space exploration. Instead, dynamic scheduling restructures the temporal sequence of the training signal, acting as a powerful regularizer that prevents convergence to narrow, over-specialized behavioral manifolds. We also found that context observability requires careful consideration: while explicit observation benefits simpler tasks, "blind" dynamic scheduling proves superior in complex environments like CarRacing.

Our approach is currently limited by the introduction of schedule hyperparameters and the need for manually defined normalization bounds in observable modes. While automated multi-stage search via Optuna mitigates manual tuning, it remains computationally expensive. Future work will explore integrating dynamic schedules with automated curriculum learning [Portelas et al., 2019] to adapt temporal structures to an agent's real-time progress. Furthermore, rather than naïvely appending observable context to the state, we plan to investigate more advanced integration methods, such as injecting contextual information directly into the latent representations of world models [e.g., Prasanna et al., 2024, Gumbsch et al., 2024].

Acknowledgments

The authors are funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 572775489. The authors acknowledge support by the state of Baden-Württemberg through bwHPC and the German Research Foundation (DFG) through grant INST 35/1597-1 FUGG.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C. Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *CoRR*, abs/2108.13264, 2021. URL `https://arxiv.org/abs/2108.13264`.

T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In A. Teredesai, V. Kumar, Y. Li, R. Rosales, E. Terzi, and G. Karypis, editors, *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, (KDD 2019)*, pages 2623–2631. ACM, 2019.

J. Beck, R. Vuorio, E. Z. Liu, Z. Xiong, L. M. Zintgraf, C. Finn, and S. Whiteson. A tutorial on meta-reinforcement learning. *Found. Trends Mach. Learn.*, 18(2-3):224–384, 2025. doi: 10.1561/2200000080.

R. Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, pages 679–684, 1957.

J. Benad, F. Röder, M. V. Butz, and M. Eppe. Shared dynamic model aligned hypernetworks for contextual reinforcement learning. In *Eighteenth European Workshop on Reinforcement Learning*, 2025. URL `https://openreview.net/forum?id=6gdvQqkFKT`.

C. Benjamins, T. Eimer, F. Schubert, A. Mohan, S. Döhler, A. Biedenkapp, B. Rosenhan, F. Hutter, and M. Lindauer. Contextualize me – the case for context in reinforcement learning. *Transactions on Machine Learning Research*, 2023.

M. Beukman, D. Jarvis, R. Klein, S. James, and B. Rosman. Dynamics generalisation in reinforcement learning via adaptive context-aware policies. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Proceedings of the 36th International Conference on Advances in Neural Information Processing Systems (NeurIPS'23)*. Curran Associates, 2023.

A. Biedenkapp. Contextual intelligence: The next leap for reinforcement learning: Blue sky ideas track. In *Proceedings of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026)*. IFAAMAS, May 2026. doi: 10.65109/QNKH4630. URL `https://doi.org/10.65109/QNKH4630`.

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI gym. *arXiv:1606.01540 [cs.LG]*, 2016.

Y. Chandak, G. Theocharous, S. Shankar, M. White, S. Mahadevan, and P. S. Thomas. Optimizing for the future in non-stationary MDPs. In Daume III and Singh [2020], pages 1414–1425.

K. Cobbe, C. Hesse, J. Hilton, and J. Schulman. Leveraging procedural generation to benchmark reinforcement learning. In Daume III and Singh [2020].

H. Daume III and A. Singh, editors. *Proceedings of the 37th International Conference on Machine Learning (ICML'20)*, volume 98, 2020. Proceedings of Machine Learning Research.

C. Ding, L. Zhou, Y. Li, and X. Rong. Locomotion control of quadruped robots with online center of mass adaptation and payload identification. *IEEE Access*, 8:224578–224587, 2020. doi: 10.1109/ACCESS.2020.3044933.

Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel. RL^2: Fast reinforcement learning via slow reinforcement learning. *arXiv:1611.02779 [cs.AI]*, 2016.

T. Eimer, C. Benjamins, and M. Lindauer. Hyperparameters in contextual rl are highly situational. In *Ecological Theory of RL Workshop NeurIPS*, December 2021.

L. Engwegen, D. Brinks, and W. Boehmer. Modular recurrence in contextual MDPs for universal morphology control. In *Eighteenth European Workshop on Reinforcement Learning*, 2025. URL https://openreview.net/forum?id=0fn0ii1njp.

B. Evans, A. Thankaraj, and L. Pinto. Context is everything: Implicit identification for dynamics adaptation. In *Proceedings of the International Conference on Robotics and Automation, (ICRA 2022)*, pages 2642–2648. IEEE, 2022. doi: 10.1109/ICRA46639.2022.9812119.

C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Precup and Teh [2017], pages 1126–1135.

C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch, and O. Bachem. Brax - A differentiable physics engine for large scale rigid body simulation. In J. Vanschoren and S.-K. Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS DBT'21)*, 2021.

Jake Grigsby, Linxi Fan, and Yuke Zhu. AMAGO: Scalable in-context reinforcement learning for adaptive agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=M6XWoEdmwf.

B. Grooten, P. MacAlpine, K. Subramanian, P. R. Wurman, and P. Stone. Out-of-distribution generalization with a sparc: Racing 100 unseen vehicles with a single policy. In *Proceedings of the Fourtieth AAAI Conference on Artificial Intelligence*. AAAI Press, 2026.

C. Gumbsch, N. Sajid, G. Martius, and M. V. Butz. Learning hierarchical world models with adaptive temporal abstractions from discrete latent dynamics. In *The Twelfth International Conference on Learning Representations (ICLR'24)*. ICLR, 2024. URL https://openreview.net/forum?id=TjCDNssXKU.

A. Hallak, D. Di Castro, and S. Mannor. Contextual markov decision processes. *arXiv:1502.02259 [stat.ML]*, 2015.

M. Iannotta, Y. Yang, J. A. Stork, E. Schaffernicht, and T. Stoyanov. Can context bridge the reality gap? sim-to-real transfer of context-aware policies. *arXiv:2511.04249 [cs.RO]*, 2025. doi: 10.48550/arXiv.2511.04249.

R. Kirk, A. Zhang, E. Grefenstette, and T. Rocktäschel. A survey of zero-shot generalisation in deep reinforcement learning. *Journal of Artificial Intelligence Research (JAIR)*, 76:201–264, 2023.

P. Klink, C. D'Eramo, J. Peters, and J. Pajarinen. Self-paced deep reinforcement learning. In Larochelle et al. [2020], pages 9216–9227.

A. Kumar, Z. Fu, D. Pathak, and J. Malik. RMA: rapid motor adaptation for legged robots. In D. A. Shell, M. Toussaint, and M. A. Hsieh, editors, *Robotics: Science and Systems XVII, (RSS 2021)*, 2021.

H. Larochelle, M. Ranzato, R. Hadsell, M.-F. Balcan, and H. Lin, editors. *Proceedings of the 33rd International Conference on Advances in Neural Information Processing Systems (NeurIPS'20)*, 2020. Curran Associates.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2017. URL https://arxiv.org/abs/1608.03983.

L. C. Melo. Transformers are meta-reinforcement learners. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning (ICML'22)*, volume 162 of *Proceedings of Machine Learning Research*, pages 15340–15359. PMLR, 2022.

A. Modi, N. Jiang, S. Singh, and A. Tewari. Markov decision processes with continuous side information. In *Algorithmic Learning Theory (ALT'18)*, volume 83, pages 597–618, 2018.

T. Camaret Ndir, A. Biedenkapp, and N. Awad. Inferring behavior-specific context improves zero-shot generalization in reinforcement learning. In *Seventeenth European Workshop on Reinforcement Learning*, 2024. URL https://openreview.net/forum?id=51XSWH0mgN.

OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang. Solving rubik's cube with a robot hand. *arXiv:1910.07113 [cs.LG]*, 2019.

C. Packer, K. Gao, J. Kos, P. Krähenbühl, V. Koltun, and D. Song. Assessing generalization in deep reinforcement learning. *arXiv:1810.12282 [cs.LG]*, 2019.

K. Panaganti, Z. Xu, D. Kalathil, and M. Ghavamzadeh. Robust reinforcement learning using offline data. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Proceedings of the 35th International Conference on Advances in Neural Information Processing Systems (NeurIPS'22)*. Curran Associates, 2022.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In Precup and Teh [2017], pages 2817–2826.

R. Portelas, C. Colas, K. Hofmann, and P.-Y. Oudeyer. Teacher algorithms for curriculum learning of deep RL in continuously parameterized environments. In L. Pack Kaelbling, D. Kragic, and K. Sugiura, editors, *Proceedings of the 3rd Annual Conference on Robot Learning, (CoRL 2019)*, pages 835–853. PMLR, 2019.

S. Prasanna, K. Farid, R. Rajan, and A. Biedenkapp. Dreaming of many worlds: Learning contextual world models aids zero-shot generalization. *Reinforcement Learning Journal*, 1, 2024.

D. Precup and Y. Teh, editors. *Proceedings of the 34th International Conference on Machine Learning (ICML'17)*, volume 70, 2017. Proceedings of Machine Learning Research.

A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *J. Mach. Learn. Res.*, 22:268:1–268:8, 2021.

K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning (ICML'19)*, volume 97, pages 5331–5340. Proceedings of Machine Learning Research, 2019.

T. Ren, C. Xiao, T. Zhang, N. Li, Z. Wang, S. Sanghavi, D. Schuurmans, and B. Dai. Latent variable representation for reinforcement learning. In *The Eleventh International Conference on Learning Representations (ICLR'23)*. ICLR, 2023.

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

G. Shala, A. Biedenkapp, P. Krack, F. Walter, and J. Grabocka. Efficient cross-episode meta-rl. In *The Thirteenth International Conference on Learning Representations (ICLR'25)*. ICLR, 2025. Published online: `iclr.cc`.

M. Suau, M. T. J. Spaan, and F. A. Oliehoek. Bad habits: Policy confounding and out-of-trajectory generalization in RL. *RLJ*, 4:1711–1732, 2024.

Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. de Las Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, T. P. Lillicrap, and M. A. Riedmiller. Deepmind control suite. *arXiv:1801.00690 [cs.AI]*, 2018.

G. Tennenholtz, N. Merlis, L. Shani, M. Mladenov, and C. Boutilier. Reinforcement learning with history dependent dynamic contexts. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning (ICML'23)*, volume 202 of *Proceedings of Machine Learning Research*, pages 34011–34053. PMLR, 2023.

J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'17)*, pages 23–30. IEEE, 2017.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, Aditya Vijaykumar, Alessandro Pietro Bardelli, Alex Rothberg, Andreas Hilboll, Andreas Kloeckner, Anthony Scopatz, Antony Lee, Ariel Rokem, C. Nathan Woods, Chad Fulton, Charles Masson, Christian Häggström, Clark Fitzgerald, David A. Nicholson, David R. Hagen, Dmitrii V. Pasechnik, Emanuele Olivetti, Eric Martin, Eric Wieser, Fabrice Silva, Felix Lenders, Florian Wilhelm, G. Young, Gavin A. Price, Gert-Ludwig Ingold, Gregory E. Allen, Gregory R. Lee, Hervé Audren, Irvin Probst, Jörg P. Dietrich, Jacob Silterra, James T Webber, Janko Slavič, Joel Nothman, Johannes Buchner, Johannes Kulick, Johannes L. Schönberger, José Vinícius de Miranda Cardoso, Joscha Reimer, Joseph Harrington, Juan Luis Cano Rodríguez, Juan Nunez-Iglesias, Justin Kuczynski, Kevin Tritz, Martin Thoma, Matthew Newville, Matthias Kümmerer, Maximilian Bolingbroke, Michael Tartre, Mikhail Pak, Nathaniel J. Smith, Nikolai Nowaczyk, Nikolay Shebanov, Oleksandr Pavlyk, Per A. Brodtkorb, Perry Lee, Robert T. McGibbon, Roman Feldbauer, Sam Lewis, Sam Tygier, Scott Sievert, Sebastiano Vigna, Stefan Peterson, Surhud More, Tadeusz Pudlik, Takuya Oshima, Thomas J. Pingel, Thomas P. Robitaille, Thomas Spura, Thouis R. Jones, Tim Cera, Tim Leslie, Tiziano Zito, Tom Krauss, Utkarsh Upadhyay, Yaroslav O. Halchenko, and Yoshiki Vázquez-Baeza. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature Methods*, 17(3):261–272, February 2020. ISSN 1548-7105. doi: 10.1038/s41592-019-0686-2. URL `http://dx.doi.org/10.1038/s41592-019-0686-2`.

J. Wang, M. King, N. Porcel, Z. Kurth-Nelson, T. Zhu, C. Deck, P. Choy, M. Cassin, M. Reynolds, H. F. Song, G. Buttimore, D. P. Reichert, N. C. Rabinowitz, L. Matthey, D. Hassabis, A. Lerchner, and M. M. Botvinick. Alchemy: A benchmark and analysis toolkit for meta-reinforcement learning agents. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, (NeurIPS DBT 2021)*, 2021.

W. Yu, J. Tan, C. K. Liu, and G. Turk. Preparing for the unknown: Learning a universal policy with online system identification. In *Robotics: Science and Systems XIII, (RSS 2017)*, 2017. doi: 10.15607/RSS.2017.XIII.048.

V. Zaburdaev, S. Denisov, and J. Klafter. Lévy walks. *Reviews of Modern Physics*, 87(2):483–530, June 2015. ISSN 1539-0756. doi: 10.1103/revmodphys.87.483. URL `http://dx.doi.org/10.1103/RevModPhys.87.483`.

A. Zhang, S. Sodhani, K. Khetarpal, and J. Pineau. Learning robust state abstractions for hidden-parameter block MDPs. In *The Ninth International Conference on Learning Representations (ICLR'21)*. ICLR, 2021.

H. Zhang, H. Chen, C. Xiao, B. Li, M. Liu, D. S. Boning, and C.-J. Hsieh. Robust deep reinforcement learning against adversarial perturbations on state observations. In Larochelle et al. [2020].

T. Zhou, J.-Hoon Cho, and C. Wu. Structure detection for contextual reinforcement learning. In S. Koenig, C. Jenkins, and M. E. Taylor, editors, *Fortieth AAAI Conference on Artificial Intelligence, AAAI 2026*, pages 29009–29016. AAAI Press, 2026.

W. Zhou, L. Pinto, and A. Gupta. Environment probing interaction policies. In *The Seventh International Conference on Learning Representations (ICLR'19)*. ICLR, OpenReview.net, 2019.

A Schedulers Details

Figure 5 illustrates trajectory examples for all eleven schedule families over three episodes.

Figure 5: Trajectory examples for all eleven schedule families, simulated over three episodes of 500 steps each (episode boundaries marked by dashed vertical lines). All values are normalised to $[-1, +1]$ relative to the training context range so that qualitative structure can be compared across context variables with different physical scales. Dotted horizontal lines mark the range boundaries (± 1) and midpoint (0). Multiple curves per panel show how the qualitative behaviour changes with the key hyperparameters of each family (amplitude, step size, drift rate, etc.).

In addition to the six canonical families described in Section 3, three further families appear in the figure above.

Sinusoidal Jump. Superimposes the sinusoidal oscillation (Section 3) onto an intermittently jumping baseline: at random timesteps drawn from a uniform interval range, the current context receives an additional signed displacement of magnitude $\Delta \sim \mathrm{Uniform}(m_{\mathrm{lo}}, m_{\mathrm{hi}})$. The result is a smooth oscillation punctuated by abrupt level shifts, combining the dense in-episode sweep of the sinusoidal family with the distributional tail coverage of stochastic jumps.

Ornstein-Uhlenbeck (OU). $dc_t = \theta(\mu - c_t)\, dt + \sigma\, dW_t$, discretised as $c_{t+1} = c_t + \theta(\mu - c_t) + \sigma\, \epsilon_t$, $\epsilon_t \sim \mathcal{N}(0, 1)$. The reversion target μ is the episode-initial context; θ controls mean-reversion speed and σ the noise level. OU produces temporally correlated drift that stays statistically anchored to a reference value, unlike the unbounded random walk.

Phased OU. Extends OU by periodically resampling the reversion target μ from a discrete set of anchor values (constructed identically to the context pool, Section 3). The retarget interval is

drawn uniformly from a specified range, producing a piece-wise-drifting trajectory where each phase smoothly relaxes toward a new anchor.

Identity (baseline). Context is held constant at its episode-initial value throughout the episode. This is the degenerate special case of a dynamic schedule and serves as the within-experiment reference against which all schedule-induced variation is measured.

B Hyperparameters and Training Details

All agents are trained with PPO [Schulman et al., 2017] via Stable-Baselines3 [Raffin et al., 2021]. Table 4 lists all hyperparameters; all values are the SB3 defaults and were not tuned — PPO is regarded as robust to hyperparameter variation and fixing the optimizer ensures that observed differences are attributable to the scheduling strategy rather than incidental tuning. The policy is an MlpPolicy with two fully-connected hidden layers of 64 units each for CartPole and BipedalWalker. CarRacing uses a MultiInputPolicy with a shared CNN feature extractor (default SB3 NatureCNN) followed by a fully-connected head; context is fed through a separate MLP branch and fused with the CNN output. The main experiments use a single parallel training environment (`N_TRAIN_ENVS` = 1); a small auxiliary set of runs explored the effect of increasing the number of parallel environments within PPO but are not part of the main comparison. Evaluation runs in four parallel workers for all conditions.

Table 4: PPO hyperparameters across environments.

Hyperparameter	**CartPole**	**BipedalWalker**	**CarRacing**
Total timesteps	150 000	3 000 000	1 500 000
Rollout steps (n)	2 048	2 048	2 048
Minibatch size	64	64	64
Learning rate	3×10^{-4}	3×10^{-4}	3×10^{-4}
Discount γ	0.99	0.99	0.99
GAE λ	0.95	0.95	0.95
Clip range ε	0.2	0.2	0.2
Entropy coefficient	0.0	0.0	0.0
Value function coefficient	0.5	0.5	0.5
Evaluation frequency	10 000	50 000	150 000
Episodes per eval context	30	30	30
Seeds	10	8	5
Policy network	MLP [64, 64]	MLP [64, 64]	CNN + MLP [64, 64]

C Environment and Context Details

CartPole

Table 5 summarises the training and evaluation context grids. The normalization bounds used for live context observation (Section 3) are manually specified conservative intervals chosen to cover a physically plausible operating range for each variable, in this case $(0.05, 2.0)$ m for the pole length. This normalization is motivated by PPO training stability; its effect is documented in Appendix D.3.6.

Table 5: CartPole context ranges and evaluation splits.

Context	**Training range**	**Eval ID**	**Eval OOD-low**	**Eval OOD-high**
Pole length (m)	$[0.35,\ 0.75]$	$\{0.37, 0.50, 0.60, 0.71\}$	$\{0.06 \ldots 0.12\}$	$\{0.92 \ldots 1.32\}$

Table 6 lists the CartPole observation space bounds used to set the bin boundaries in the 4D coverage diagnostic.

The pool-7 used in the main CartPole experiments contains the 7 midpoint-inserted values $\{0.35,\ 0.41,\ 0.48,\ 0.55,\ 0.62,\ 0.68,\ 0.75\}$ for pole length and $\{5.0,\ 7.5,\ 10.0,\ 12.5,\ 15.0\}$ for force magnitude and gravity (pool-5).

Table 6: Observation Space Bounds of CartPole

Observation	Min	Max
Cart Position	-4.8	4.8
Cart Velocity	$-\infty$	∞
Pole Angle	$\sim$ -0.418 rad (-24°)	$\sim$ 0.418 rad (24°)
Pole Angular Velocity	$-\infty$	∞

BipedalWalker

A rigid payload of mass $2.0\,\text{kg}$ and radius $0.5\,\text{m}$ is attached to the torso. The context COM_X shifts this payload along the body's longitudinal axis. Training range is $[-0.6,\ +0.6]$ m; pool-3 contains $\{-0.6,\ 0.0,\ +0.6\}$. The normalization bounds are $(-2.0,\ +2.0)$ m, a conservatively chosen physical operating range for the payload offset.

Table 7: BipedalWalker evaluation splits (COM_X, m).

	Eval ID	Eval OOD-low	Eval OOD-high
COM_X (m)	$\{-0.4, -0.2, 0.0, +0.2, +0.4\}$	$\{-1.05, \ldots, -0.65\}$	$\{+0.9, \ldots, +1.7\}$

Table 8 lists the BipedalWalker observation space bounds used to define the coverage projections.

Table 8: BipedalWalker Observation Space Bounds

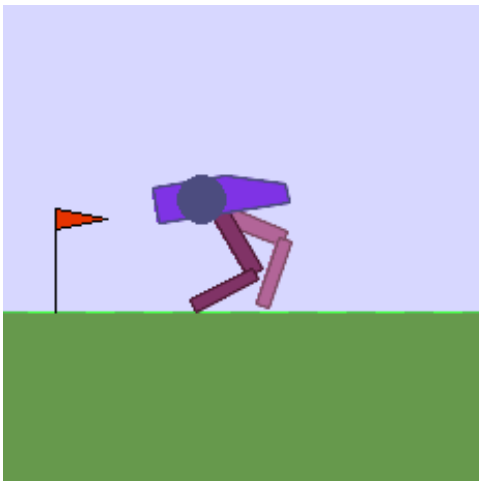
Index	Description	Min	Max
0	Hull Angle	$-\pi$	π
1	Hull Angular Velocity	-5.0	5.0
2	Horizontal Speed	-5.0	5.0
3	Vertical Speed	-5.0	5.0
4	Hip 1 (Joint Angle)	$-\pi$	π
5	Hip 1 Speed	-5.0	5.0
6	Knee 1 (Joint Angle)	$-\pi$	π
7	Knee 1 Speed	-5.0	5.0
8	Leg 1 Contact	0.0	5.0
9	Hip 2 (Joint Angle)	$-\pi$	π
10	Hip 2 Speed	-5.0	5.0
11	Knee 2 (Joint Angle)	$-\pi$	π
12	Knee 2 Speed	-5.0	5.0
13	Leg 2 Contact	0.0	5.0
14–23	LiDAR Measurements (x10)	-1.0	1.0

CarRacing

Two context axes: COM_X and COM_Y (vehicle center-of-mass offsets), each in $[-0.8,\ +0.8]$ m, with pool-3 containing $\{-0.8,\ 0.0,\ +0.8\}$ per axis. The normalization bounds are $(-2.0,\ +2.0)$ m per axis, conservatively covering the physical operating range. The payload has mass $1.0\,\text{kg}$ and radius $0.5\,\text{m}$. Table 9 summarises the evaluation splits; both axes use identical ranges.

Table 9: CarRacing evaluation splits (COM_X and COM_Y, m). Both axes share the same ranges.

	Eval ID	Eval OOD-low	Eval OOD-high
COM_X / COM_Y (m)	$\{-0.5, \ldots, +0.5\}$	$\{-1.5,\ -1.3,\ -1.1,\ -0.9\}$	$\{+0.9,\ +1.1,\ +1.3,\ +1.5\}$



(a) BipedalWalker with attached payload (dark mass on torso).

(b) CarRacing with attached longitudinally offset payload (cyan marker).

Figure 6: Visualisation of the payload-extended environments. Both payloads are novel additions to the CARL benchmark introduced in this work. Shifting the payload position changes the center-of-mass of the agent, altering its dynamics in a physically grounded and continuously parameterisable way.

D Additional Results

D.1 Cross-Environment Scheduler Analysis

We present three complementary views of the scheduler search results across all environments, aggregating over the full set of scheduler families evaluated.

Fraction of schedulers outperforming the static baseline. Figure 7 shows, for each environment and context-observability mode, the fraction of scheduler families whose last-checkpoint IQM exceeds the corresponding static baseline. Context-observed dynamic schedulers beat the static observed baseline in nearly all cases (89–100% across environments), confirming that any dynamic curriculum helps when the agent can observe the context. The blind mode is more nuanced: in CartPole and Walker roughly 70–78% of blind schedulers improve over the static blind baseline, whereas in CarRacing the picture splits sharply — the longitudinal (COM_X) static blind baseline is exceptionally strong, with only 3 out of 9 schedulers surpassing it, while the lateral (COM_Y) static blind baseline is weak and all 9 schedulers beat it.

Cross-environment performance ranking. Figure 8 shows a heatmap of each scheduler family's performance relative to the static baseline (score $= 0$) and the best scheduler in that column (score $= 1$), with families sorted by their mean score across all six columns. No single family dominates across all environments and modes: Sinusoidal, Lévy Walk, and Sudden Jump rank most consistently above the static baseline on average (Piecewise Constant ranks highest overall but was only evaluated on CartPole, leaving its Walker and CarRacing cells empty), while Phased OU and Cosine Annealing are the weakest overall. Dynamic schedulers reliably outperform the static baseline in the context-observed columns (zero red cells for CartPole and Walker observed; only one marginal exception for CarRacing observed), and the CarRacing (lat.) blind column is entirely green. The most notable failures occur in the Walker blind column, where Cosine Annealing and Random Walk fall clearly below the static blind baseline, suggesting these families are particularly sensitive to the absence of context information in that environment.

Context-observed vs. context-blind gap. Figure 9 quantifies the blind−observed IQM gap per scheduler family, normalised by each environment's own IQM range so that results are comparable across environments. The pattern is consistent across nearly all scheduler families: CartPole benefits from observing the context (negative gap, observed wins), CarRacing benefits strongly from *not*

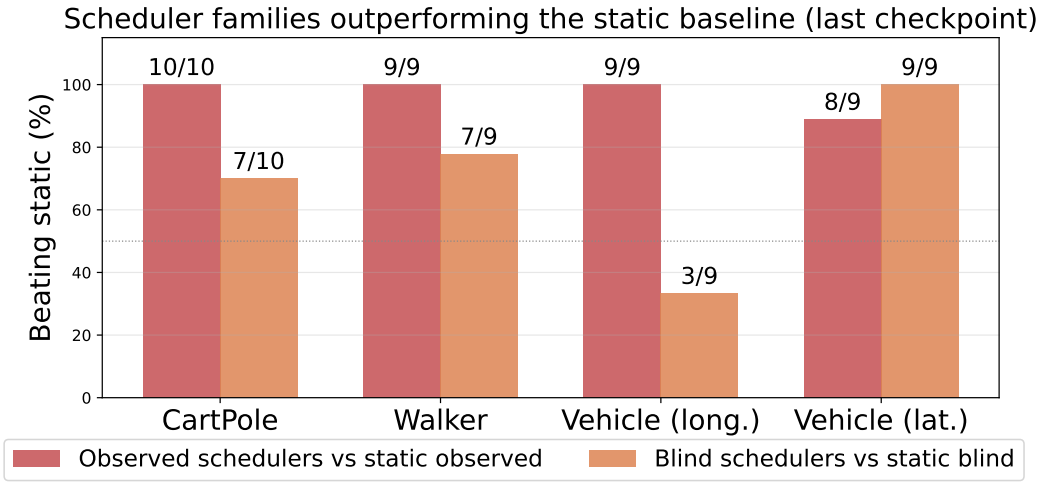


Figure 7: Fraction of scheduler families (last checkpoint IQM) that outperform the static baseline per environment and mode. Numbers above bars indicate the exact count over the total number of families evaluated.

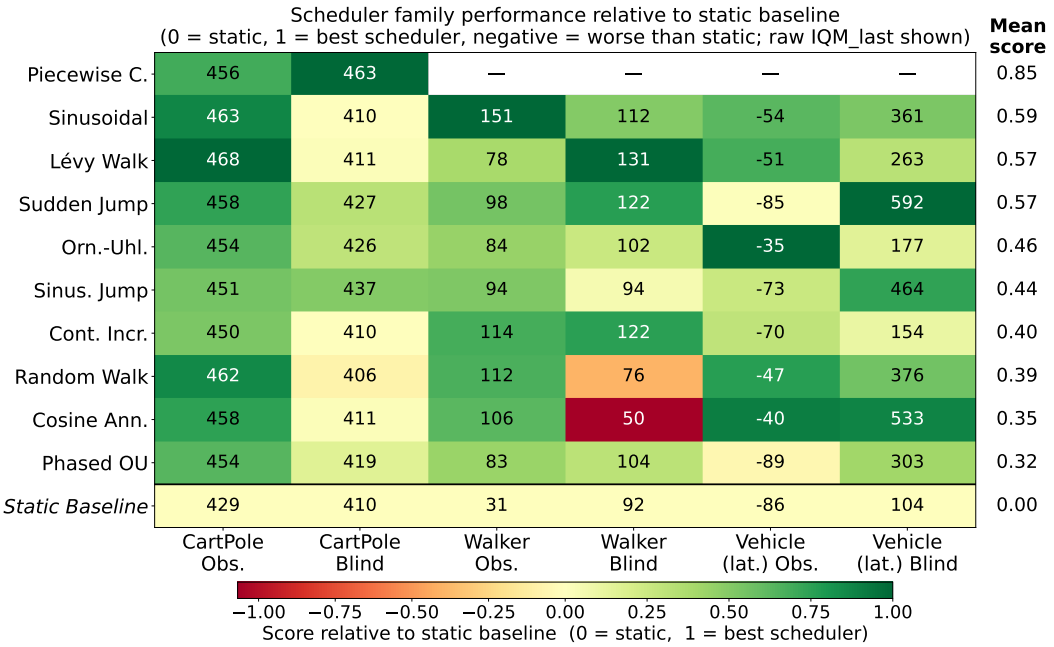


Figure 8: Heatmap of scheduler family performance relative to the static baseline. Colour encodes the static-anchored score: yellow ($= 0$) matches the static baseline, green ($= 1$) is the best scheduler in that column, red indicates worse than static. Raw IQM values (last checkpoint) are shown in each cell. Families are sorted by mean score across all columns (right margin).

observing it (large positive gap, blind wins), and Walker sits in between with mixed results depending on the family. This environment-level reversal holds independently of which scheduler is used, suggesting the effect is driven by the environment rather than the scheduler choice. Notably, Cosine Annealing, Random Walk, and Sinusoidal are the only families that consistently prefer the observed mode in both CartPole and Walker, while Sudden Jump and Cosine Annealing show the largest blind advantage in CarRacing.

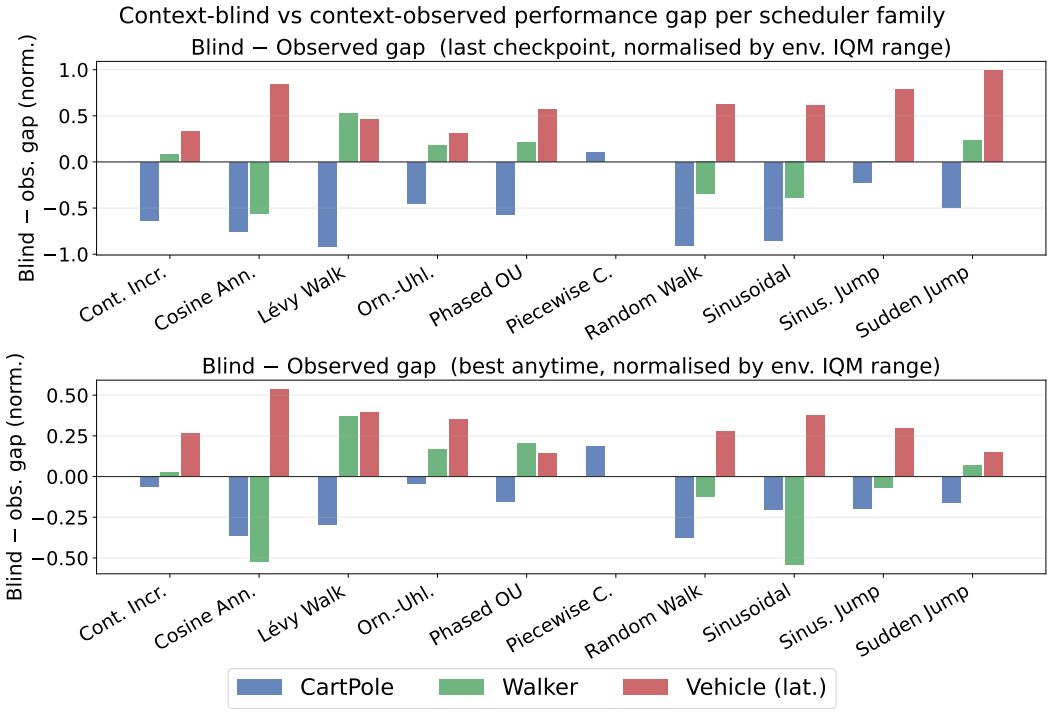


Figure 9: Blind−observed IQM gap per scheduler family and environment, normalised by the environment's IQM range (max−min across all schedulers). Positive values indicate blind schedulers outperform observed ones; negative values indicate the reverse. Top panel: last checkpoint; bottom panel: best anytime.

D.2 Best-Anytime Checkpoint Results for Scheduler Search

Table 10 reports the best-anytime IQM — the highest `avg_combined` score achieved at any evaluation checkpoint during training — as a complement to the last-checkpoint results in Table 2. Dynamic schedulers improve over the static baseline across all environments and modes at peak performance, and the margins are generally larger than at the final checkpoint, indicating that the best policies are learned earlier in training and partially lost to instability or overfitting by the end.

The gains are most pronounced for Walker, where the best observed scheduler reaches 193.0 vs. static observed 134.2 (+59 IQM), and for CarRacing, where the best blind scheduler reaches 778.9 vs. static blind 617.7 (+161 IQM). CartPole improvements are modest in comparison, consistent with the ceiling effect noted in the main text. Notably, the best CarRacing observed scheduler (634.1) now substantially exceeds the static observed baseline (358.5) at peak, though it still falls short of the static blind baseline (617.7), confirming that context observability remains a net negative in that environment even at best-achieved performance.

Table 10: Combined IQM score of the **best anytime checkpoint**, computed from the per-seed `avg_combined` metric (average over all ID and OOD contexts). Values: IQM [Q1, Q3]. Bold marks the best scheduler per environment.

Condition	CartPole	Walker	CarRacing (lateral)
Static observed	449.7 [440.3, 462.8]	134.2 [89.5, 167.9]	358.5 [183.6, 450.8]
Static blind	463.4 [453.4, 474.0]	146.9 [118.3, 181.4]	617.7 [272.5, 803.1]
Best sched. observed	**479.6** [469.3, 491.0]	**193.0** [156.7, 233.4]	634.1 [520.7, 659.4]
Best sched. blind	479.3 [465.5, 495.9]	175.5 [151.9, 200.5]	**778.9** [718.5, 813.5]

D.3 CartPole

D.3.1 Scheduler Search

Figure 10 shows the full combined IQM training progression for all scheduler families on CartPole pole length (pool-7), with each family represented by its best-performing configuration selected by final checkpoint combined IQM.

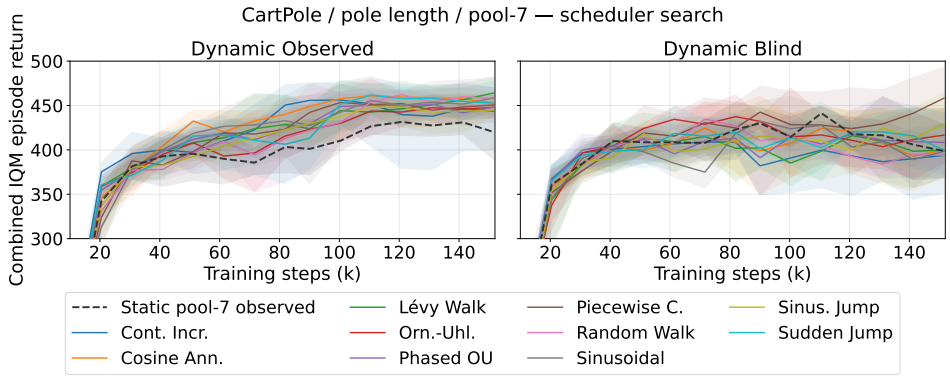


Figure 10: CartPole / pole length / pool-7 — combined IQM training progression per scheduler family (10 seeds). Each line shows the best representative configuration for that family, selected by final combined IQM. Shaded bands indicate Q1–Q3 across seeds. The dashed line marks the static pool-7 baseline.

D.3.2 Dynamic vs. Static: Coverage and Score

To directly compare dynamic and static schedulers on coverage independently of pool size effects, we use all sequential tracking runs that include representative dynamic families: identity (static baseline), sinusoidal, Lévy walk, and sudden jump, across all available pool sizes (10 seeds per condition). This selection deliberately spans scheduler families with qualitatively different temporal structure — smooth oscillation, heavy-tailed jumps, and abrupt discrete displacements — while avoiding any configuration tied to the scheduler search or the pool size study.

The 4D coverage metric discretises CartPole's four observation dimensions into 12 bins each ($12^4 = 20{,}736$ cells), with boundaries set to physically plausible ranges: cart position $[-2.5, 2.5]$ m, cart velocity $[-3.5, 3.5]$ m/s, pole angle $[-0.30, 0.30]$ rad, and pole angular velocity $[-4.0, 4.0]$ rad/s.

Figure 11 plots coverage against final eval score for each seed. No scheduler family occupies a systematically higher or lower region of the coverage axis: dynamic and static seeds are interleaved throughout. The Pearson correlation is $r = 0.21$ ($p = 0.003$), indicating a negligible linear relationship between coverage and score — knowing how much of the state space a scheduler visits tells you almost nothing about how well it generalises.

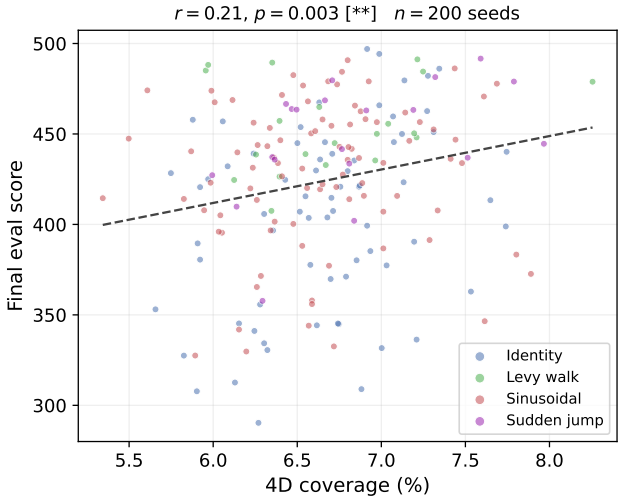


Figure 11: 4D state-space coverage vs. final eval score across all sequential seeds (pool sizes 3 and 7 for dynamic schedulers, pool sizes 1–61 for identity), coloured by scheduler. Dynamic and static seeds are interleaved throughout the coverage axis. The dashed line shows the least-squares fit ($r = 0.21$).

D.3.3 Pool Size Scaling

We vary the number of training contexts from 1 to 61 for the static identity baseline and two dynamic schedulers (sinusoidal, cosine annealing) on CartPole pole-length, using 10 seeds per condition. The scheduler hyperparameters used here are fixed reference configurations chosen for the tracking runs and are not the result of the scheduler search described in Section 4.2; absolute scores for the dynamic conditions should therefore not be compared directly to the best-found schedulers reported elsewhere. Figure 12 shows the last-checkpoint IQM score and final 4D state-space coverage. The static baseline is sensitive to pool size: it peaks around pool-3 and degrades at both extremes, with pool-1 providing too little diversity and pool-61 diluting training across too many contexts. Dynamic schedulers are largely unaffected by pool size, remaining competitive from pool-1 upwards. The training curves in Figure 13 confirm this: static pool-3 converges fastest among static conditions, while pools 31 and 61 plateau noticeably lower, and sinusoidal pool-7 tracks or exceeds the best static condition throughout.

Crucially, 4D state-space coverage is flat at $\approx 6.5\%$ across all pool sizes and all schedulers, and Table 11 shows that this coverage is concentrated almost entirely in the first 50 k training steps: stages 2 and 3 contribute less than 1%, and virtually no cells are newly discovered after stage 1 ($\leq 0.03\%$). Pool size therefore modulates the *distribution* of the training signal across context values, not the breadth of states visited.

Table 11: Fraction of the 12^4 CartPole hypergrid visited per 50 k-step training window (mean over 10 seeds, %). Coverage in stages 2–3 is below 1% for all conditions; virtually no cells are newly discovered after stage 1.

	Identity			**Sinusoidal**			**Cosine annealing**		
Pool	S1	S2	S3	S1	S2	S3	S1	S2	S3
1	6.45	0.57	0.27	6.48	0.52	0.29	6.68	0.71	0.27
3	6.93	0.75	0.32	6.68	0.61	0.28	6.52	0.53	0.26
5	6.65	0.77	0.29	6.70	0.69	0.29	6.80	0.62	0.32
7	6.65	0.64	0.29	6.39	0.77	0.29	6.61	0.61	0.30
15	6.67	0.64	0.29	6.69	0.58	0.27	6.61	0.62	0.31
31	6.82	0.71	0.31	6.57	0.56	0.29	6.70	0.63	0.30
61	6.41	0.69	0.32	6.29	0.57	0.29	6.90	0.79	0.34

S1 = 0–50k; S2 = 50–100k; S3 = 100–150k steps.

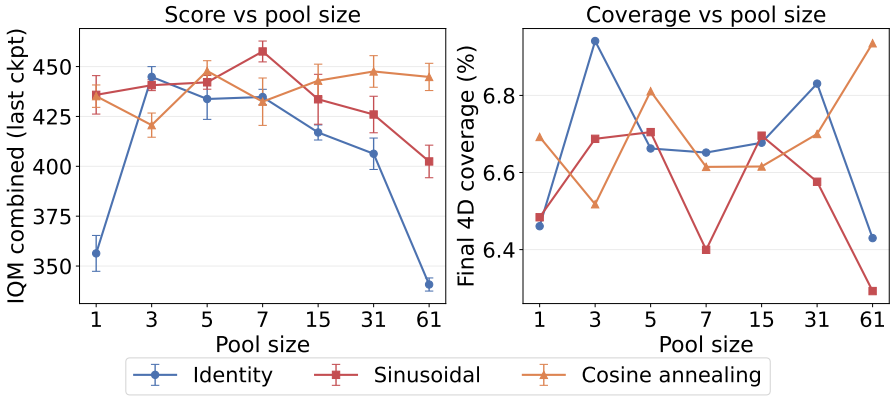


Figure 12: Last-checkpoint IQM score (A) and final 4D coverage (B) vs. pool size for CartPole pole-length (10 seeds). Error bars show SEM.

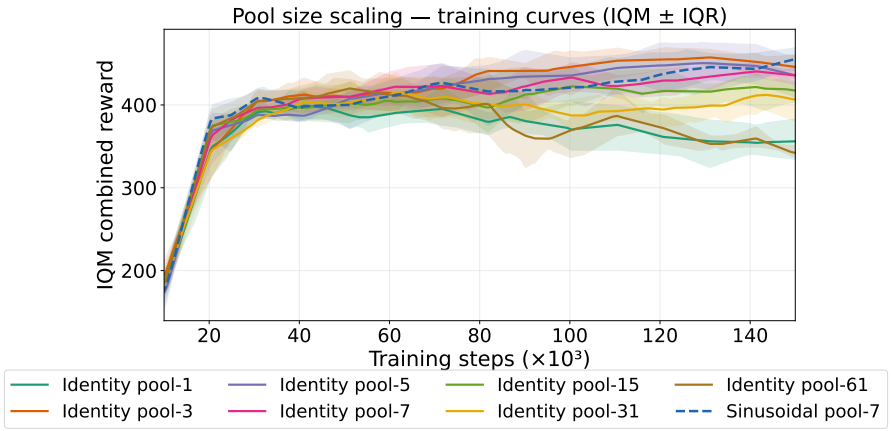


Figure 13: IQM training curves (shading: Q1–Q3) for identity pools 1–61; sinusoidal pool-7 shown dashed for reference.

D.3.4 Parallel vs. Sequential Training

We compare running 8 parallel environments against a single sequential environment for identity, sinusoidal, and cosine annealing schedulers at pool sizes 5 and 15 (and identity at pool sizes 3 and 7), using 10 seeds per condition on CartPole pole-length. Figure 14 shows the last-checkpoint IQM score and final 4D coverage for each condition, with the diagonal marking equal performance.

Both metrics scatter tightly around the diagonal with no consistent direction (Table 12): score differences range from -21 to $+28$ IQM with no pattern across schedulers or pool sizes, and coverage differences are below 0.5% in all cases. The summed deltas across all eight conditions are $+13.1$ IQM and -0.56% coverage, both negligible. Parallel training neither expands state-space coverage nor reliably improves generalisation.

This result is informative beyond the numerical statement. Parallel environments, typical for PPO, introduce context diversity *across* simultaneous rollouts, enriching each policy gradient update with transitions from multiple dynamics regimes at once. Dynamic scheduling, by contrast, introduces context variation *within* a single episode, exposing the policy to a structured temporal sequence of dynamics changes during one rollout. The fact that the former does not replicate the gains of the latter suggests that the benefit of dynamic scheduling is not simply a consequence of seeing more diverse contexts per update, but is tied to the intra-episode temporal structure of the training signal itself.

Table 12: Score and coverage difference (parallel − sequential) per condition. Positive = parallel preferred; negative = sequential preferred.

Scheduler	Pool	Δ IQM score	Δ Coverage
Identity	3	−7.1	−0.47%
Identity	5	+1.8	+0.22%
Identity	7	+11.8	+0.12%
Identity	15	+28.3	−0.02%
Sinusoidal	5	+0.4	−0.25%
Sinusoidal	15	−20.7	−0.25%
Cosine annealing	5	−15.8	−0.04%
Cosine annealing	15	+14.4	+0.13%
Sum		+13.1	−0.56%
Parallel preferred		4/8	3/8

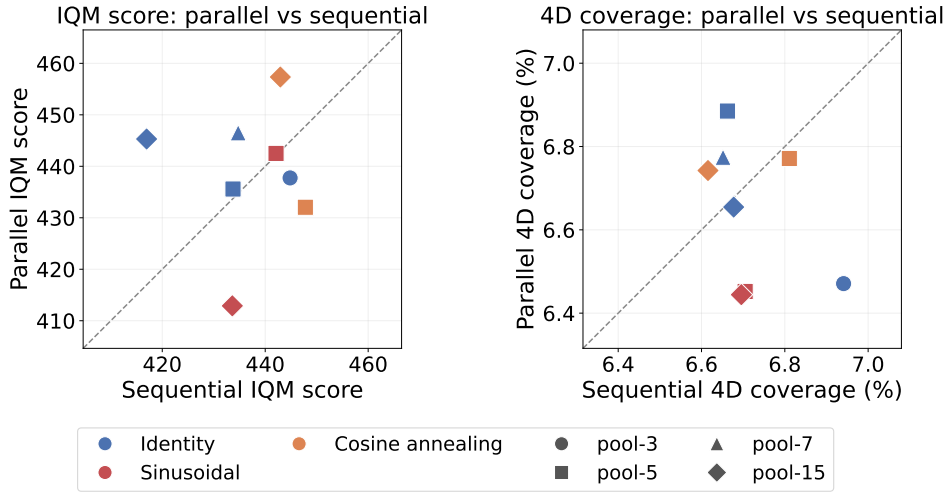


Figure 14: Sequential vs. parallel IQM score (A) and 4D coverage (B) for CartPole. Points on the dashed diagonal indicate no difference. Colour encodes scheduler family; marker shape encodes pool size.

D.3.5 BipedalWalker State-Space Coverage

We extend the coverage analysis to BipedalWalker (COM_X context, 4 seeds per pool size, pool sizes 1–15, sequential only) using four complementary projections of the 24-dimensional observation space. For each projection we compute the fraction of discretised cells visited over 1 M training steps and compare across scheduler families (identity, sinusoidal, cosine annealing, Lévy walk). No projection yields a significant difference across schedulers (Figure 15), consistent with the CartPole finding.

Table 13 reports full-run coverage across all four projections alongside mean training return (averaged over pool sizes; note this is rollout return, not the IQM evaluation metric used in the main text). None of the per-projection ANOVAs are significant. Across projections, the static identity baseline consistently ranks highest on coverage and lowest on return, while sinusoidal ranks lowest on coverage and highest on return. Given the limited number of seeds, this trend is suggestive rather than conclusive; the broader state-space spread of the static baseline may reflect slower convergence rather than active exploration.

Figure 16 breaks 4D joint coverage into three equal training stages (0–333k, 333–667k, 667k–1 M steps) with a shared y-axis. All schedulers drop from $\approx$10–13% in stage 1 to $\approx$3–5% by stage 3, confirming that BipedalWalker policies also converge to a narrow behavioral manifold well before the end of training, regardless of scheduler.

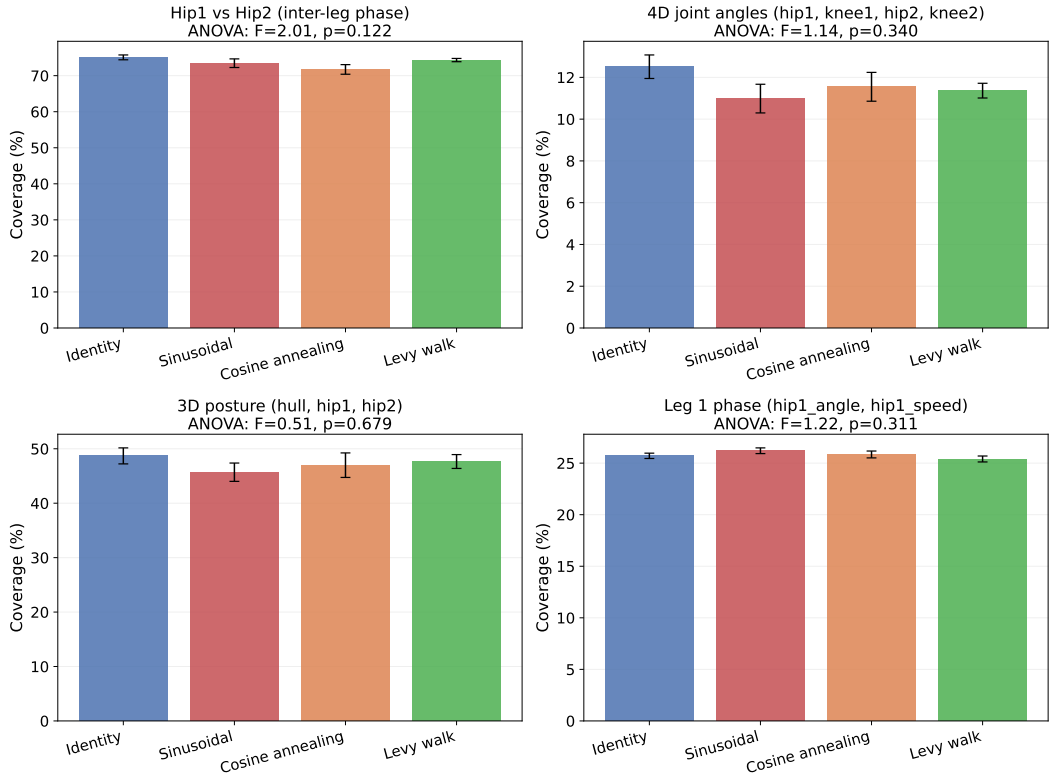


Figure 15: BipedalWalker full-run coverage per scheduler across four state-space projections (4 seeds $\times$ 4 context ranges, aggregated). **A**: inter-leg phase (30^2, $p = 0.12$). **B**: 4D joint angles (12^4, $p = 0.34$). **C**: 3D posture (15^3, $p = 0.68$). **D**: leg-1 phase (30^2, $p = 0.31$).

Table 13: Full-run coverage (%) per scheduler across four projections and mean rollout return, averaged over pool sizes 1, 3, 7, 15 (4 seeds each). Bold marks the highest value per column.

	Coverage (%)				
Scheduler	**A: inter-leg**	**B: 4D joints**	**C: posture**	**D: leg phase**	**Return**
Identity	**75.1**	**12.5**	**48.7**	25.7	142
Cosine annealing	71.7	11.6	47.0	25.8	150
Lévy walk	74.3	11.4	47.7	25.4	163
Sinusoidal	73.5	11.0	45.7	**26.2**	**177**
ANOVA p	0.12	0.34	0.68	0.31	—

A: hip_1 vs hip_2 (30^2); B: hip_1, knee_1, hip_2, knee_2 (12^4); C: hull, hip_1, hip_2 (15^3); D: leg-1 phase (30^2).

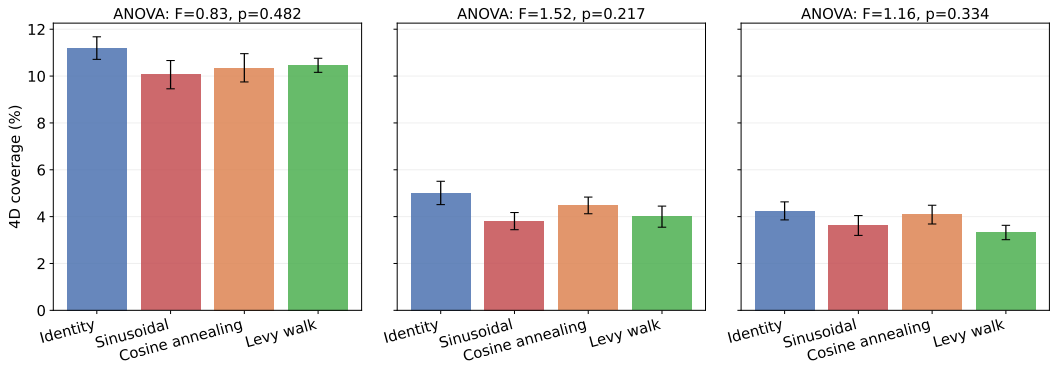
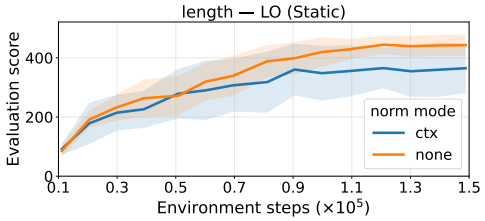


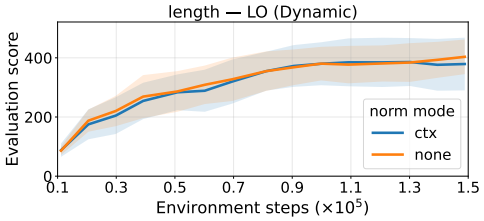
Figure 16: BipedalWalker 4D joint-space coverage per scheduler across three equal training stages ($\approx 333\,\mathrm{k}$ steps each). Shared y-axis shows the progressive contraction of the visited state space. ANOVA $p > 0.05$ for all stages.

D.3.6 Context Normalization Ablation

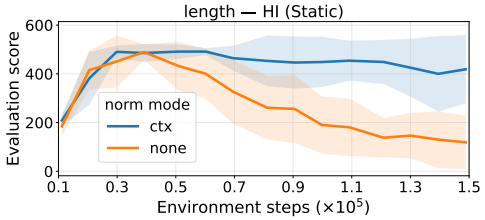
The figure below support the normalization design choice discussed in Section 3. Results are from a preliminary subset of schedule families; the effect is consistent across conditions.



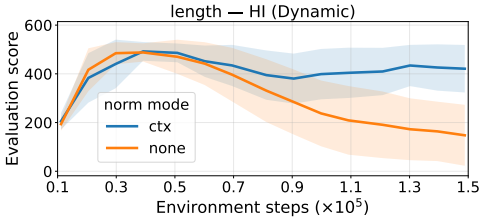
(a) Static training — low pole length.



(b) Dynamic training — low pole length.



(c) Static training — high pole length.



(d) Dynamic training — high pole length.

Figure 17: Effect of context normalization on static and dynamic PPO training (CartPole, pole length context). The dynamic curves aggregate performance across a preliminary set of non-stationary schedules. Lines show mean evaluation score across seeds; shaded regions indicate standard deviation.

D.4 Additional Bar Plots per Eval Context

CartPole. The last-checkpoint bar plot appears in the main paper (Figure 2); here we include the best-anytime companion. All conditions perform strongly as the ID region saturates at the maximum reward of 500. The meaningful spread is in OOD-low, where static observed is the weakest (IQM 313 at last checkpoint) and dynamic blind the strongest (IQM 456), with static blind and dynamic observed in between. Dynamic observed leads at the last checkpoint (combined IQM 464) while dynamic blind leads at best-anytime (488); the two are close throughout, suggesting that even a blind scheduler captures most of the benefit of temporal context variation on this environment.

BipedalWalker. Figure 19 shows per-context IQM at the last checkpoint (pool-3). The static observed baseline achieves strong in-distribution performance (IQM 114) but collapses on OOD contexts (IQM -48 high, -10 low), pulling its combined IQM to only 31: with three fixed training contexts the policy overfits to those specific dynamics and fails to generalise. Dynamic schedulers substantially outperform both static conditions; even dynamic blind comfortably beats static blind, confirming that intra-episode temporal structure adds value beyond naïve domain randomisation.

CarRacing. Context-observed conditions exhibit significant late-training policy degradation on this environment; we therefore show both the last-checkpoint and best-anytime figures side by side for each axis. At the last checkpoint, dynamic blind is the strongest condition on both axes. At best-anytime, static blind leads on COM_X (IQM 842.9 vs. 803.7 for dynamic blind) while dynamic blind leads on COM_Y (760.7 vs. 636.4 for static blind); all observed conditions struggle to stably exploit the live context signal through the CNN pipeline. Per-scheduler IQM values are in Tables 17 and 18.

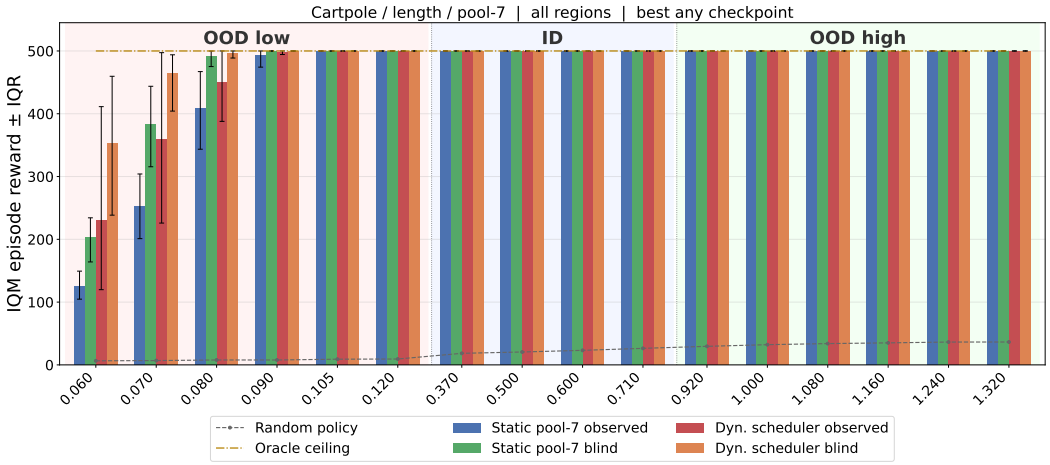


Figure 18: CartPole combined IQM (best anytime, pool-7). Per-scheduler values in Table 15.

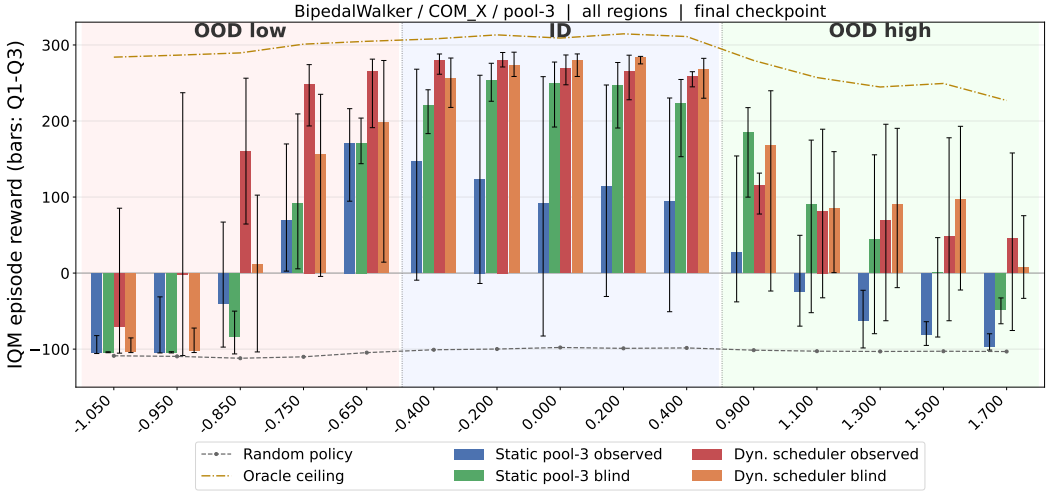


Figure 19: BipedalWalker combined IQM (last checkpoint, pool-3). Per-scheduler values in Table 16.

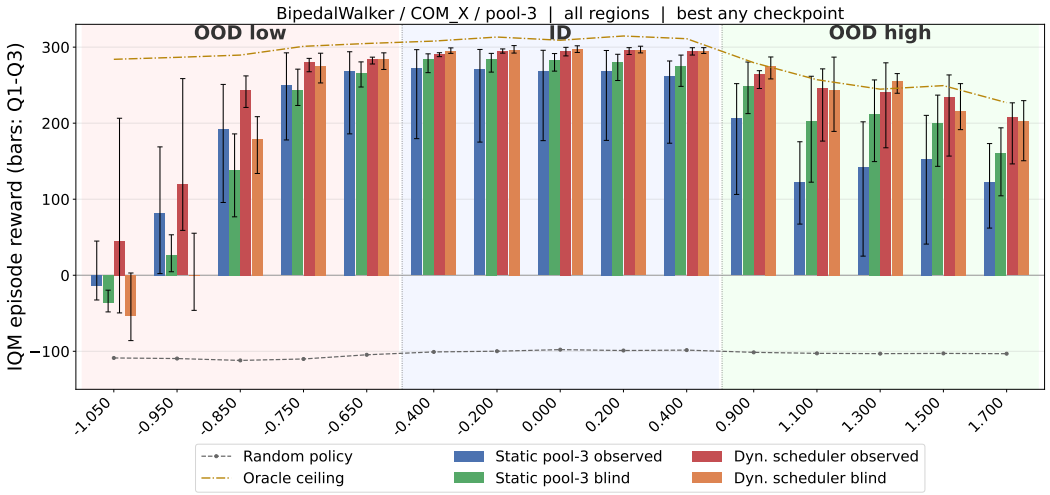


Figure 20: BipedalWalker combined IQM (best anytime, pool-3). The gap relative to the last-checkpoint figure above is most pronounced for the static observed baseline, which collapses late in training.

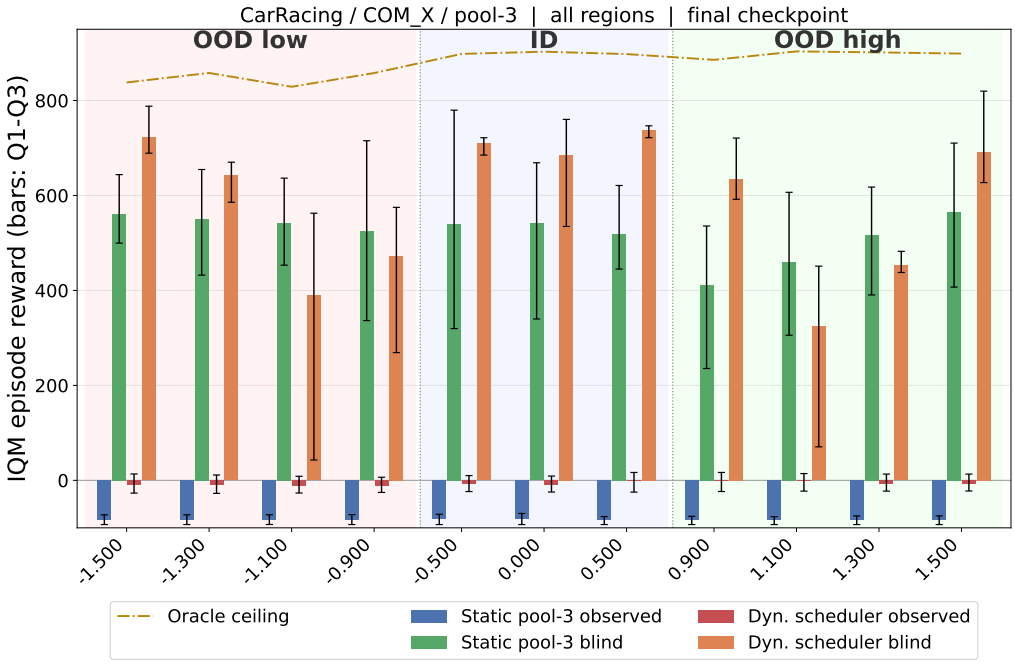


Figure 21: CarRacing COM_X last-checkpoint IQM (pool-3).

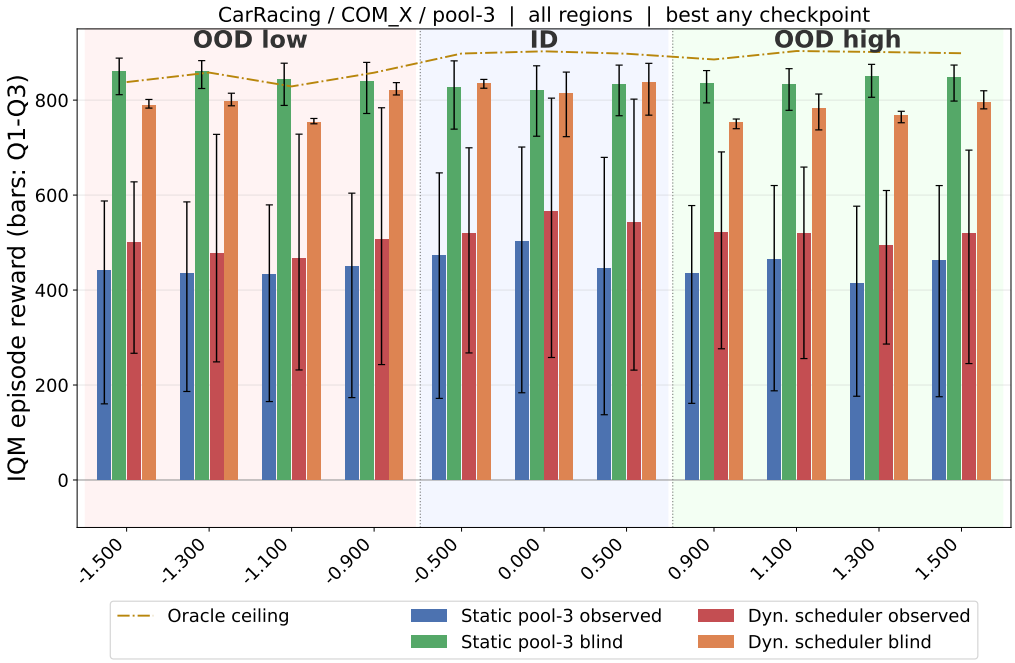


Figure 22: CarRacing COM_X best-anytime IQM (pool-3).

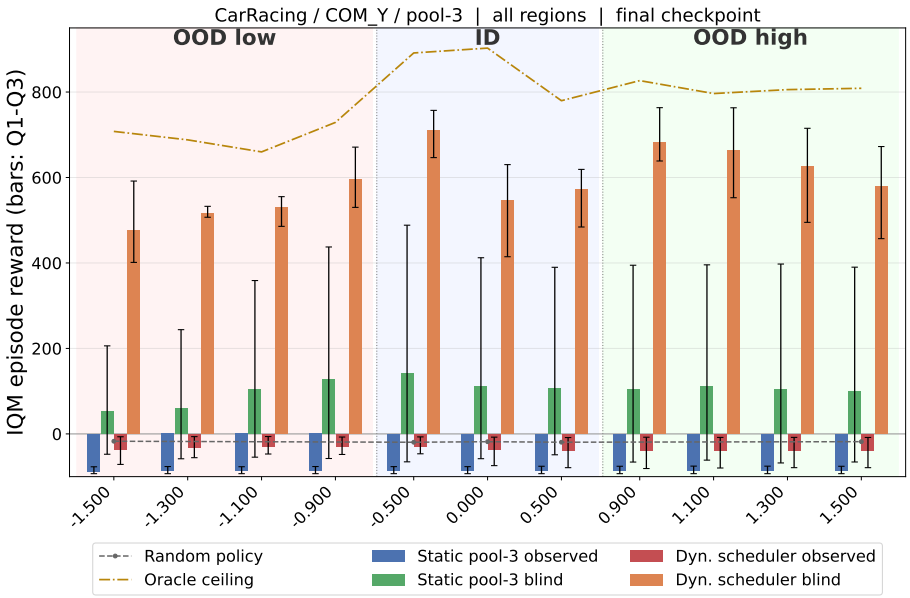


Figure 23: CarRacing COM_Y last-checkpoint IQM (pool-3).

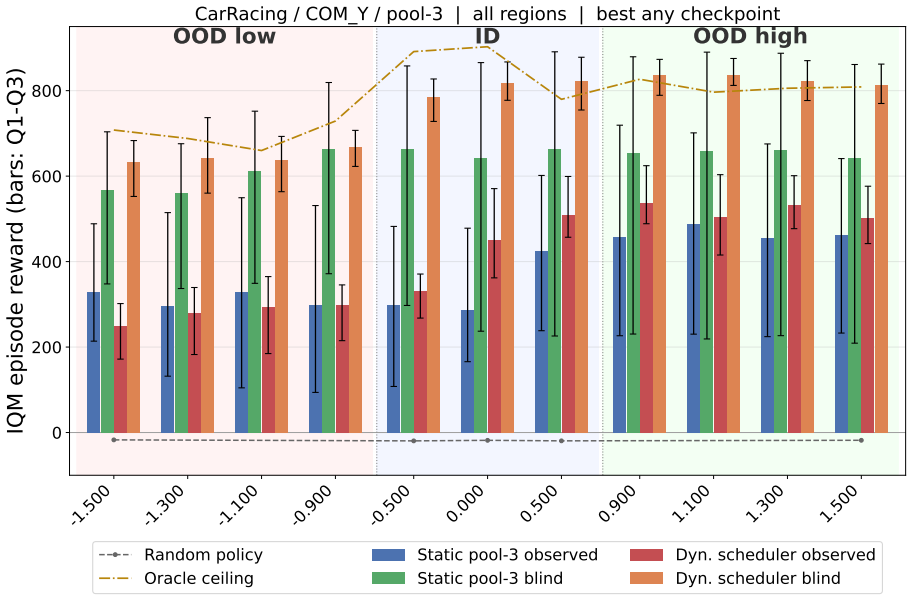


Figure 24: CarRacing COM_Y best-anytime IQM (pool-3).

E Multi-stage Schedulers Additional Results

E.1 Best-Anytime Results

Table 14 complements Table 3 with best-anytime IQM scores — the highest combined evaluation score achieved at any stage boundary during training. Walker values remain at the matched 1.5 M-step budget.

Table 14: Multi-stage Optuna retrain results at the **best-anytime checkpoint**. Walker values at the matched 1.5 M-step budget. Values: IQM [Q1, Q3]. Top-3 retrained trials shown per environment.

Condition	CartPole	Walker (1.5 M)
Static observed	449.7 [440.3, 462.8]	143.6 [114.5, 167.9]
Static blind	463.4 [453.4, 474.0]	136.3 [101.8, 160.4]
Best sched. observed	**479.6** [469.3, 491.0]	**176.9** [135.6, 197.0]
Best sched. blind	479.3 [465.5, 495.9]	140.3 [108.7, 164.9]
Optuna #1 (T168 / T118)	470.1 [450.9, 492.1]	171.4 [148.6, 196.2]
Optuna #2 (T218 / T117)	469.3 [447.4, 482.9]	165.2 [149.2, 177.3]
Optuna #3 (T145 / T066)	477.6 [458.2, 492.6]	128.8 [111.7, 150.1]

E.2 Optuna Results

The figures below detail the CartPole Optuna retrain: training curves (Figure 25), scheduler mode frequency per stage (Figure 26), hyperparameter distributions for active stages (Figure 27), and rank shift after retraining with more seeds (Figure 28).

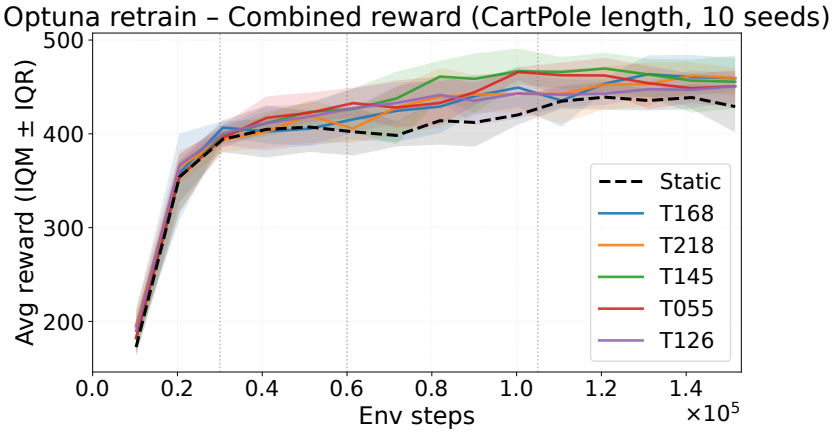


Figure 25: Learning curves (IQM ± IQR) for the top-5 retrained CartPole trials vs. static baseline.

E.3 Walker

The figures below mirror the CartPole analysis for BipedalWalker: scheduler patterns and final scores (Figure 29a), training curves (Figure 30), rank shift (Figure 31), mode frequency (Figure 32), and hyperparameter distributions (Figure 33).

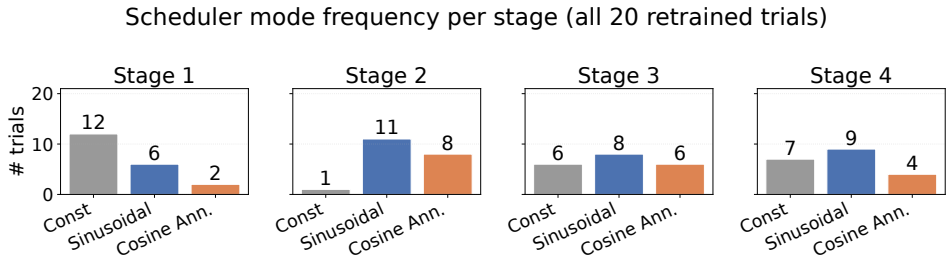


Figure 26: Frequency of each scheduler mode (constant, sinusoidal, cosine annealing) at each stage position across all 20 retrained CartPole trials. Stage 2 is active in nearly all configurations; Stage 1 is most often constant.

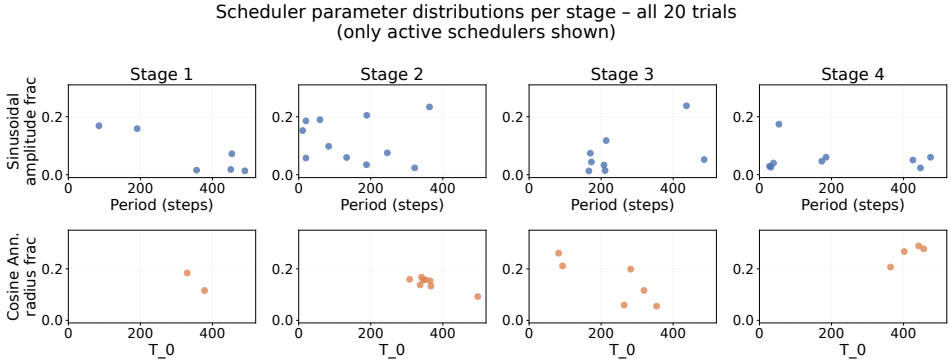


Figure 27: Hyperparameter distributions for active stages across all 20 retrained CartPole trials. Top row: sinusoidal amplitude fraction vs. period; bottom row: cosine annealing neighbourhood radius vs. T_0. Only trials where the respective mode is active are shown.

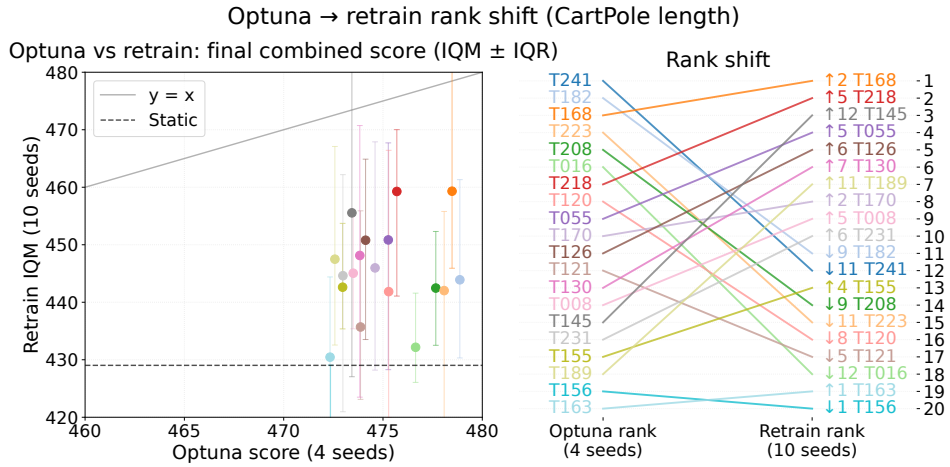
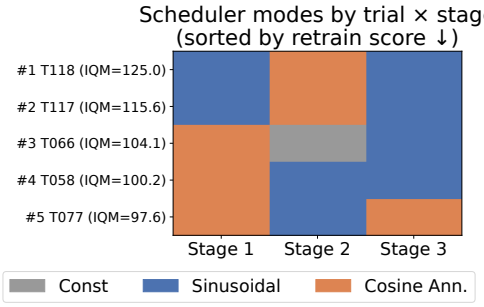
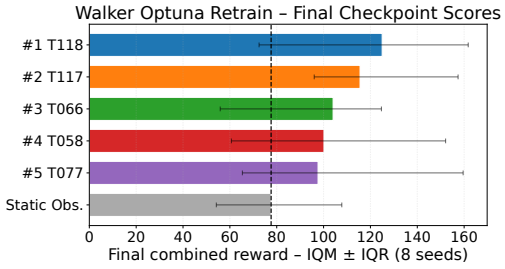


Figure 28: Rank shift between the 4-seed Optuna search ranking and the 10-seed retrain ranking for CartPole. Left: scatter of Optuna score vs. retrain IQM $\pm$ IQR; the dashed line marks the static pool-7 baseline. Right: bump chart showing the rank change for each trial after retraining with more seeds.



(a) Scheduler mode per stage for the top-5 retrained trials, sorted by retrain IQM.



(b) Final-checkpoint IQM ± IQR for top-5 retrained trials vs. static baseline (1.5 M steps).

Figure 29: BipedalWalker Optuna retrain: scheduler patterns and final scores.

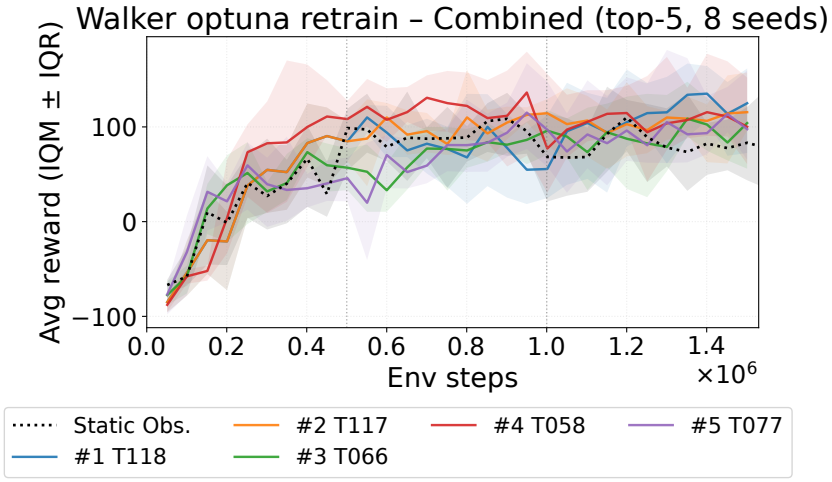


Figure 30: Learning curves (IQM ± IQR) for the top-5 retrained Walker trials vs. static baseline.

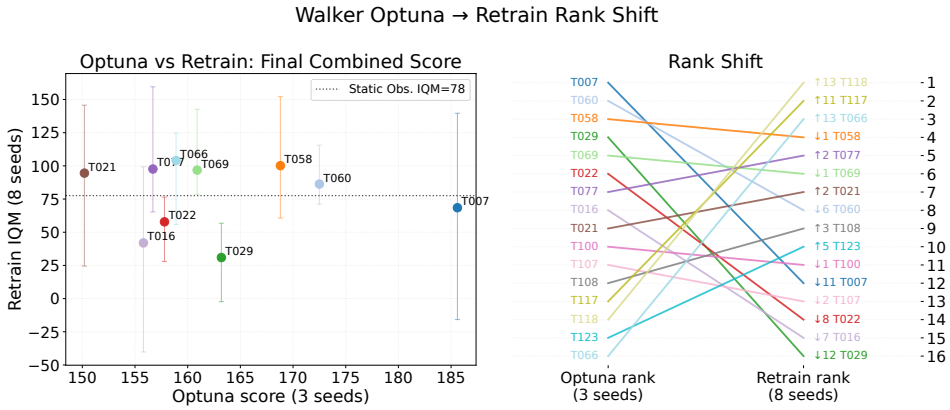


Figure 31: Rank shift between the 3-seed Optuna ranking and the 8-seed retrain ranking for BipedalWalker.

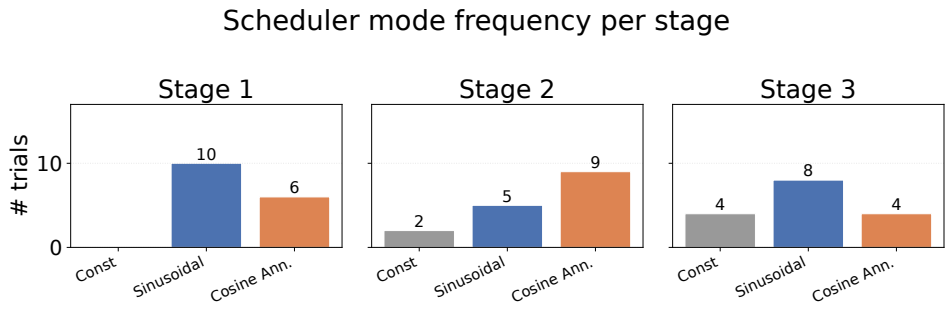


Figure 32: Scheduler mode frequency per stage across all retrained Walker trials.

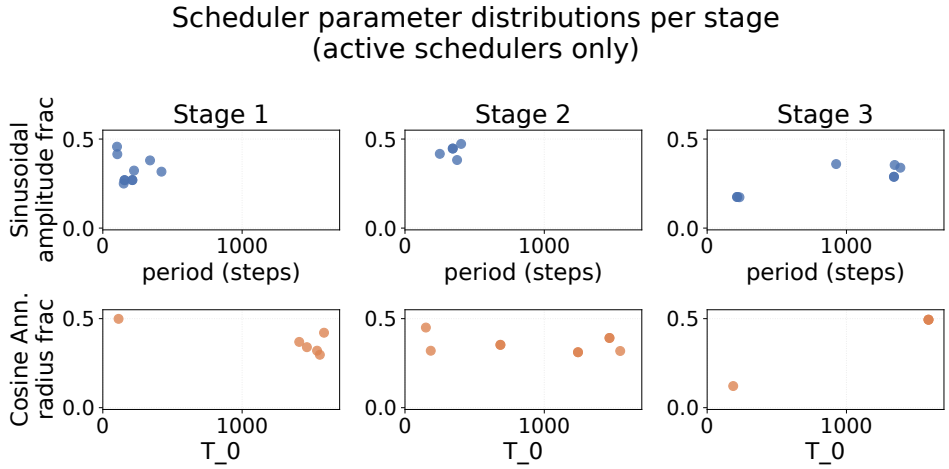


Figure 33: Scheduler parameter distributions per stage for all retrained Walker trials (active schedulers only).

F Scheduler Performance

Tables 15–18 report the per-scheduler-family IQM [Q1, Q3] of `avg_combined` for both context-observability modes across all three environments. For each family and mode, we show the single best-performing configuration found during the scheduler search, selected by IQM of the last-checkpoint `avg_combined` score. Both last-checkpoint and best-anytime IQM are reported. Bold marks the highest IQM in each column; in Table 17 the best-anytime blind column is an exception where the static baseline exceeds all dynamic schedulers, with the best dynamic result marked in italic bold for reference.

Table 15: CartPole / length / pool-7. Per-scheduler-type IQM [Q1, Q3] of `avg_combined` over 10 seeds.

Scheduler	Mode	Last IQM [Q1, Q3]	Best IQM [Q1, Q3]
Static Baseline	Observed	429.0 [401.5, 451.4]	449.7 [440.3, 462.8]
	Blind	410.0 [383.1, 436.5]	463.4 [453.4, 474.0]
Cont. Incrementer	Observed	449.8 [411.5, 473.2]	477.6 [449.6, 494.7]
	Blind	409.9 [381.7, 437.5]	473.4 [455.0, 487.8]
Cosine Annealing	Observed	458.2 [444.6, 468.8]	477.9 [469.4, 484.6]
	Blind	411.0 [384.5, 452.6]	455.0 [444.6, 468.6]
Lévy Walk	Observed	**468.2** [454.7, 484.3]	471.8 [457.1, 493.2]
	Blind	411.0 [383.2, 452.8]	453.1 [435.0, 473.5]
Ornstein–Uhlenbeck	Observed	453.8 [443.3, 467.0]	467.1 [460.3, 474.6]
	Blind	425.9 [398.2, 452.5]	464.2 [446.7, 480.3]
Phased OU	Observed	454.0 [448.3, 460.4]	467.6 [453.1, 481.2]
	Blind	418.6 [383.9, 447.3]	458.1 [434.7, 475.4]
Piecewise Const.	Observed	456.1 [439.3, 465.4]	467.5 [461.1, 477.5]
	Blind	**462.9** [430.9, 494.1]	**479.3** [465.5, 495.9]
Random Walk	Observed	462.4 [452.0, 472.9]	475.3 [462.1, 484.2]
	Blind	405.7 [390.4, 422.0]	451.9 [442.5, 465.6]
Sinusoidal	Observed	463.2 [450.9, 483.4]	475.9 [463.9, 490.1]
	Blind	409.7 [396.8, 425.3]	463.0 [457.9, 470.3]
Sinusoidal Jump	Observed	450.7 [444.7, 460.2]	466.1 [457.9, 475.5]
	Blind	436.8 [417.6, 468.3]	453.7 [435.4, 472.5]
Sudden Jump	Observed	457.9 [442.8, 479.9]	**479.6** [469.3, 491.0]
	Blind	427.0 [402.5, 449.2]	469.6 [452.5, 480.5]

Table 16: BipedalWalker / COM_X / pool-3. Per-scheduler-type IQM [Q1, Q3] of `avg_combined` over 8 seeds.

Scheduler	Mode	Last IQM [Q1, Q3]	Best IQM [Q1, Q3]
Static Baseline	Observed	31.0 [−34.9, 63.8]	134.2 [89.5, 167.9]
	Blind	91.9 [60.6, 113.0]	146.9 [118.3, 181.4]
Cont. Incrementer	Observed	113.6 [97.6, 130.2]	154.5 [140.1, 179.6]
	Blind	121.5 [62.1, 178.3]	157.5 [121.1, 196.8]
Cosine Annealing	Observed	106.5 [75.9, 142.5]	153.7 [134.7, 180.4]
	Blind	50.0 [−36.0, 121.8]	100.6 [10.3, 172.1]
Lévy Walk	Observed	77.6 [54.9, 105.8]	137.5 [121.6, 154.6]
	Blind	**131.1** [94.9, 167.6]	**175.5** [151.9, 200.5]
Ornstein–Uhlenbeck	Observed	83.9 [10.9, 157.8]	131.6 [49.1, 220.9]
	Blind	102.3 [46.8, 154.7]	148.8 [120.5, 182.7]
Phased OU	Observed	83.4 [70.5, 96.1]	129.1 [100.7, 164.9]
	Blind	104.4 [90.8, 124.3]	150.0 [117.5, 179.9]
Random Walk	Observed	111.6 [81.6, 142.2]	171.3 [158.6, 184.1]
	Blind	76.4 [25.2, 134.2]	158.7 [96.9, 198.1]
Sinusoidal	Observed	**151.4** [122.2, 185.3]	**193.0** [156.7, 233.4]
	Blind	111.7 [58.7, 153.3]	137.9 [73.2, 183.8]
Sinusoidal Jump	Observed	93.7 [82.0, 104.4]	166.9 [143.2, 180.0]
	Blind	94.0 [79.8, 105.4]	160.1 [144.6, 178.5]
Sudden Jump	Observed	97.6 [34.9, 147.9]	143.3 [101.9, 160.6]
	Blind	121.8 [103.8, 138.8]	150.3 [132.9, 189.4]

Table 17: CarRacing / longitudinal / pool-3. Per-scheduler-type IQM [Q1, Q3] of `avg_combined` over 5 seeds.

Scheduler	Mode	Last IQM [Q1, Q3]	Best IQM [Q1, Q3]
Static Baseline	Observed	−81.0 [−93.1, −69.4]	446.7 [160.9, 606.1]
	Blind	521.9 [377.7, 656.7]	**833.5** [775.6, 873.4]
Cont. Incrementer	Observed	−28.9 [−35.4, −17.4]	327.5 [221.3, 436.2]
	Blind	262.8 [94.8, 466.2]	703.5 [530.4, 833.7]
Cosine Annealing	Observed	−59.6 [−86.5, −12.9]	594.7 [350.5, 745.6]
	Blind	**602.5** [409.6, 664.3]	762.0 [617.0, 771.4]
Lévy Walk	Observed	**−7.0** [−24.6, 12.1]	503.5 [224.7, 716.4]
	Blind	542.6 [246.4, 762.8]	***811.8*** [723.0, 854.3]
Ornstein–Uhlenbeck	Observed	−30.8 [−60.5, 0.1]	470.8 [447.3, 512.4]
	Blind	503.7 [453.5, 537.9]	731.7 [697.4, 795.6]
Phased OU	Observed	−40.2 [−78.6, −17.3]	441.1 [232.0, 649.9]
	Blind	254.5 [−48.7, 458.3]	632.4 [552.8, 732.5]
Random Walk	Observed	−77.5 [−84.3, −66.1]	454.1 [379.1, 582.3]
	Blind	308.3 [145.9, 494.0]	798.3 [767.1, 825.3]
Sinusoidal	Observed	−11.0 [−38.1, 14.6]	574.5 [513.0, 657.2]
	Blind	589.6 [534.7, 656.8]	788.4 [727.0, 809.7]
Sinusoidal Jump	Observed	−50.2 [−81.3, −4.9]	373.9 [206.6, 479.2]
	Blind	280.4 [76.3, 526.6]	754.7 [702.2, 840.7]
Sudden Jump	Observed	−57.0 [−63.4, −51.1]	**692.3** [641.7, 733.5]
	Blind	305.9 [235.6, 477.2]	742.9 [634.8, 815.9]

Table 18: CarRacing / lateral / pool-3. Per-scheduler-type IQM [Q1, Q3] of `avg_combined` over 5 seeds.

Scheduler	Mode	Last IQM [Q1, Q3]	Best IQM [Q1, Q3]
Static Baseline	Observed	-85.9 $[-93.1,\ -76.2]$	358.5 $[183.6,\ 450.8]$
	Blind	103.6 $[-59.0,\ 378.9]$	617.7 $[272.5,\ 803.1]$
Cont. Incrementer	Observed	-70.5 $[-83.7,\ -45.1]$	380.3 $[110.8,\ 643.4]$
	Blind	153.8 $[-16.6,\ 391.1]$	564.0 $[443.6,\ 656.1]$
Cosine Annealing	Observed	-40.1 $[-82.5,\ -10.1]$	395.8 $[273.8,\ 560.3]$
	Blind	533.2 $[458.5,\ 618.7]$	762.3 $[720.9,\ 796.3]$
Lévy Walk	Observed	-51.3 $[-77.7,\ -17.4]$	488.1 $[354.0,\ 570.5]$
	Blind	263.4 $[164.6,\ 372.5]$	756.3 $[661.9,\ 781.4]$
Ornstein–Uhlenbeck	Observed	$\mathbf{-35.2}$ $[-67.3,\ -7.4]$	397.9 $[323.9,\ 471.0]$
	Blind	177.2 $[-67.5,\ 333.2]$	638.0 $[374.1,\ 771.8]$
Phased OU	Observed	-88.7 $[-93.1,\ -82.7]$	554.4 $[491.3,\ 624.4]$
	Blind	302.5 $[242.7,\ 392.1]$	654.3 $[536.9,\ 730.3]$
Random Walk	Observed	-47.3 $[-68.8,\ -18.1]$	587.0 $[556.9,\ 639.5]$
	Blind	375.6 $[336.6,\ 423.1]$	$\mathbf{778.9}$ $[718.5,\ 813.5]$
Sinusoidal	Observed	-54.1 $[-79.9,\ -31.2]$	446.3 $[397.0,\ 513.5]$
	Blind	360.6 $[226.8,\ 545.9]$	705.5 $[647.8,\ 755.8]$
Sinusoidal Jump	Observed	-72.8 $[-82.1,\ -60.3]$	525.7 $[342.8,\ 667.6]$
	Blind	463.9 $[364.5,\ 548.6]$	729.8 $[659.1,\ 817.7]$
Sudden Jump	Observed	-85.1 $[-90.9,\ -60.6]$	$\mathbf{634.1}$ $[520.7,\ 659.4]$
	Blind	$\mathbf{592.3}$ $[549.1,\ 625.7]$	735.2 $[713.3,\ 754.5]$

G Scheduler Training Progression

Each figure shows the IQM episode return averaged over training checkpoints for all scheduler families, with one line per family using its best representative run. Left panel: dynamic **observed** schedulers. Right panel: dynamic **blind** schedulers. The dashed line is the static pool baseline. Shaded bands indicate Q1–Q3 across seeds.

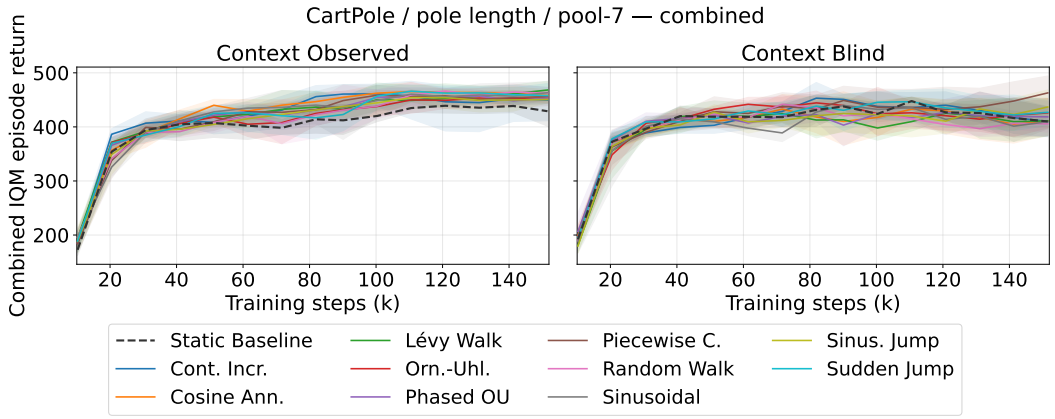


Figure 34: CartPole / pole length / pool-7 — **combined** IQM (all contexts, 10 seeds). Each family's best representative is selected by final combined IQM.

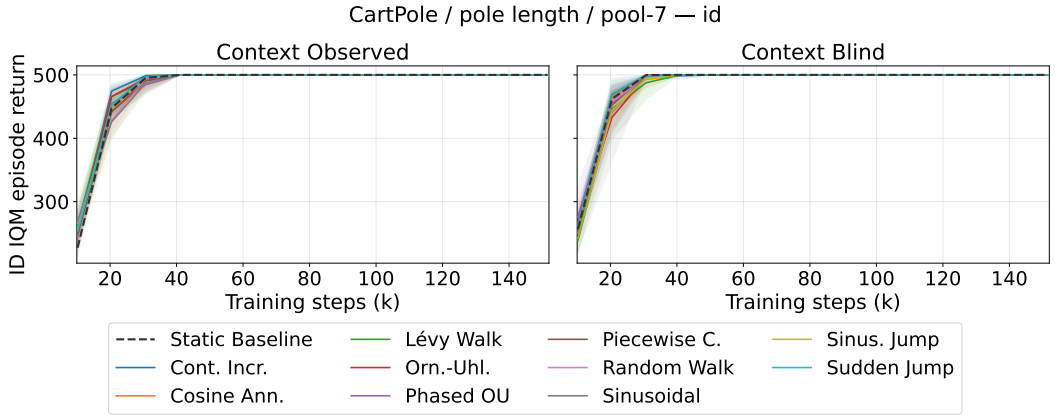


Figure 35: CartPole / pole length / pool-7 — **ID** IQM (in-distribution contexts only, 10 seeds).

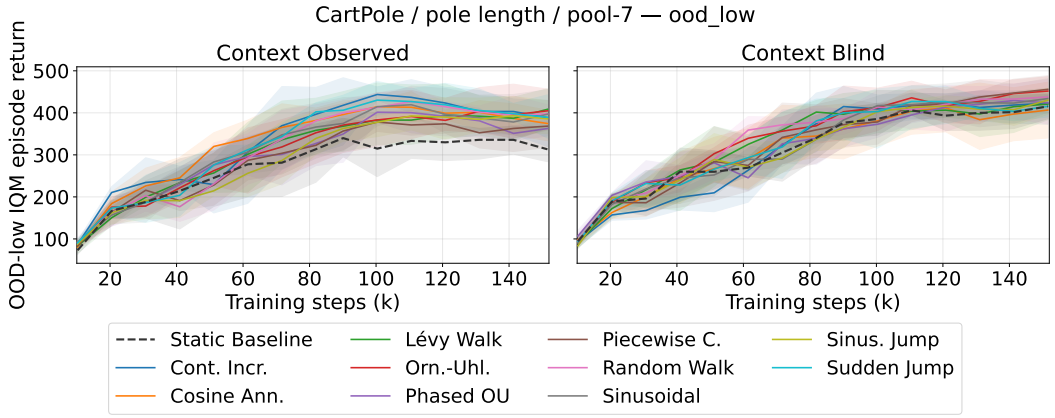


Figure 36: CartPole / pole length / pool-7 — **OOD-low** IQM (short-pole contexts below training range, 10 seeds).

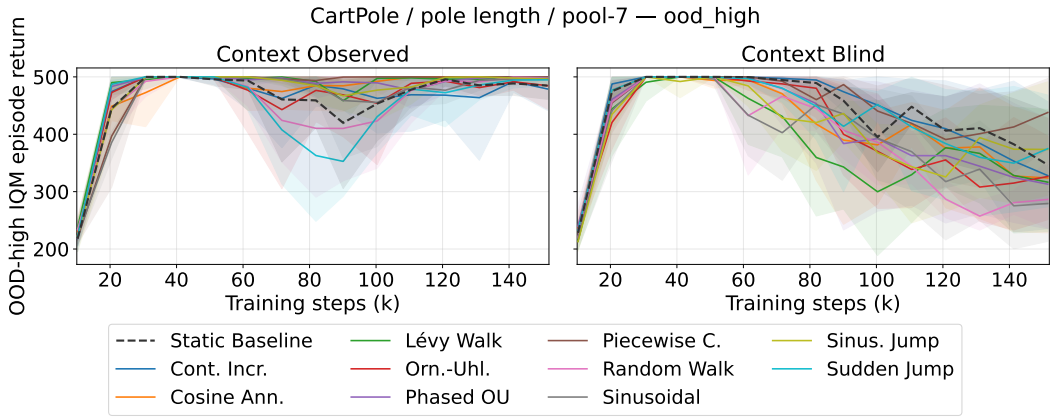


Figure 37: CartPole / pole length / pool-7 — **OOD-high** IQM (long-pole contexts above training range, 10 seeds).

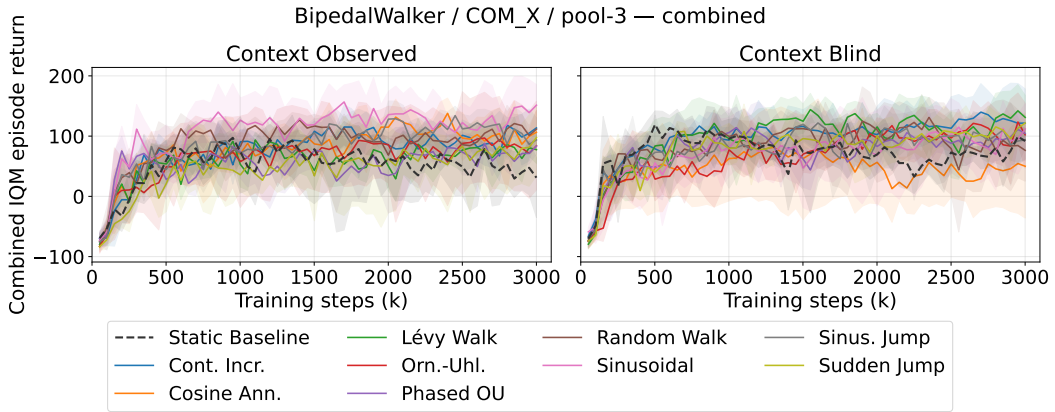


Figure 38: BipedalWalker / COM_X / pool-3 — **combined** IQM (all contexts, 8 seeds).

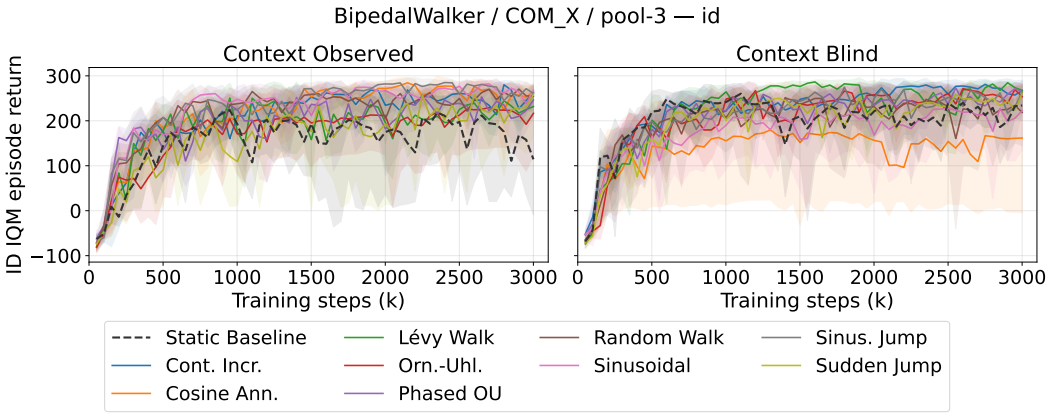


Figure 39: BipedalWalker / COM_X / pool-3 — **ID** IQM (in-distribution contexts only, 8 seeds).

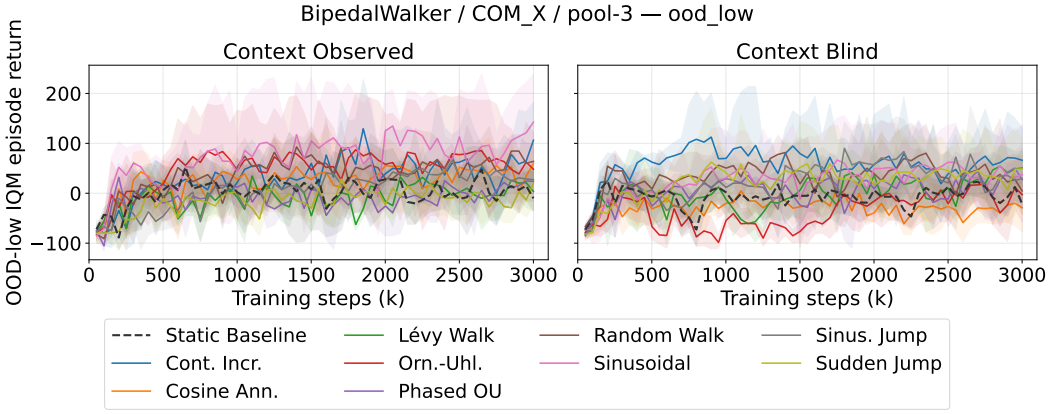


Figure 40: BipedalWalker / COM_X / pool-3 — **OOD-low** IQM (centre-of-mass shifted below training range, 8 seeds).

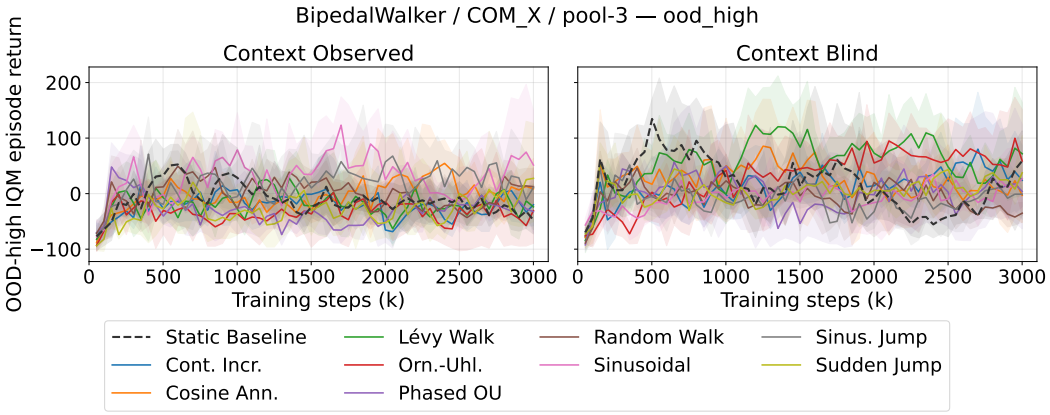


Figure 41: BipedalWalker / COM_X / pool-3 — **OOD-high** IQM (centre-of-mass shifted above training range, 8 seeds).

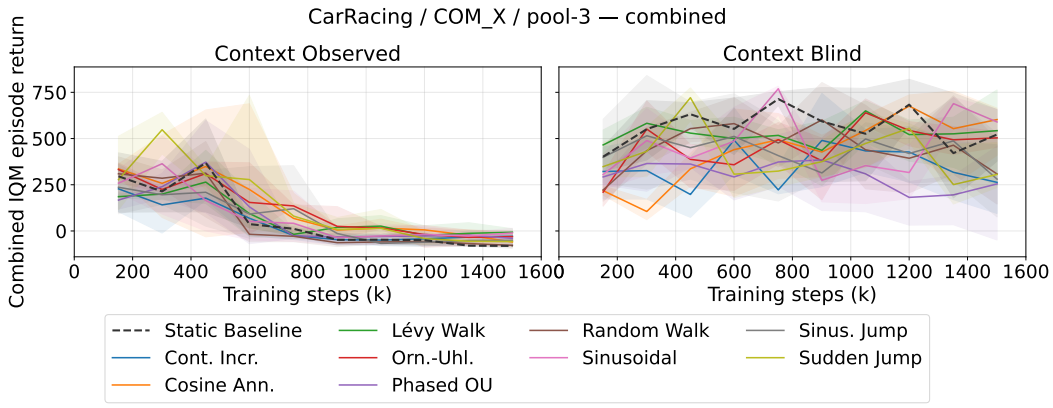


Figure 42: CarRacing / COM_X (longitudinal CoM) / pool-3 — **combined** IQM (all contexts, 5 seeds).

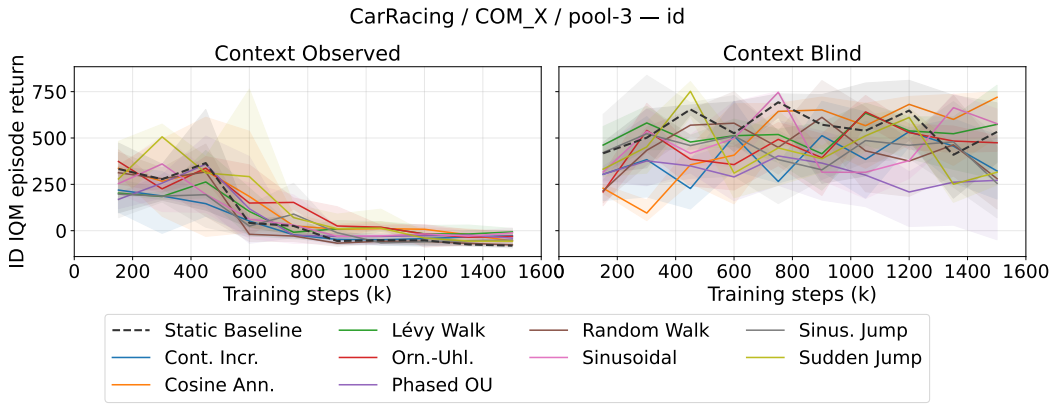


Figure 43: CarRacing / COM_X / pool-3 — **ID** IQM (in-distribution contexts only, 5 seeds).

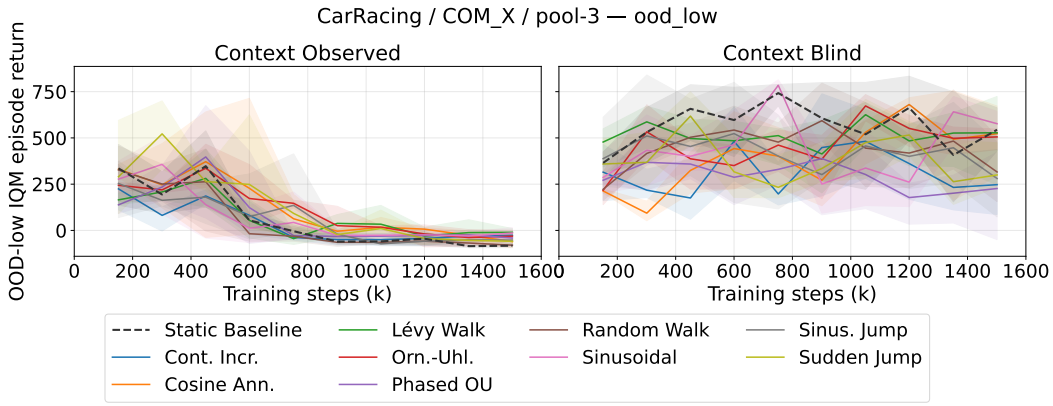


Figure 44: CarRacing / COM_X / pool-3 — **OOD-low** IQM (longitudinal CoM shifted below training range, 5 seeds).

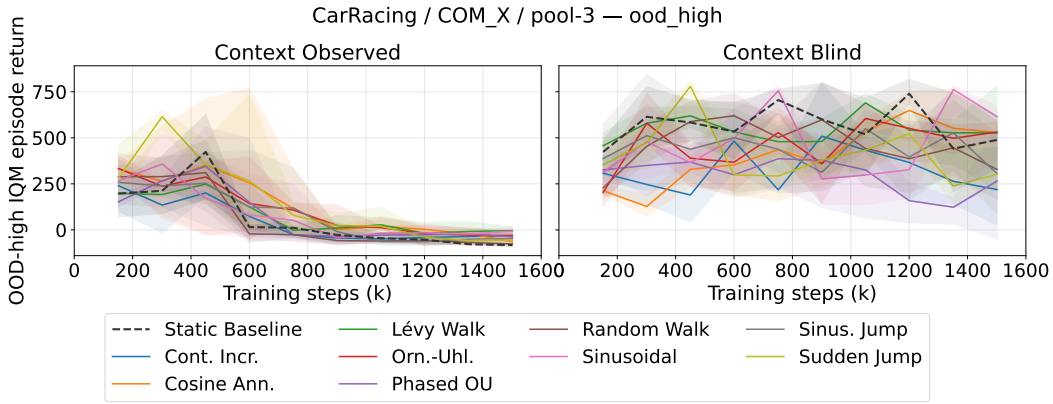


Figure 45: CarRacing / COM_X / pool-3 — **OOD-high** IQM (longitudinal CoM shifted above training range, 5 seeds).

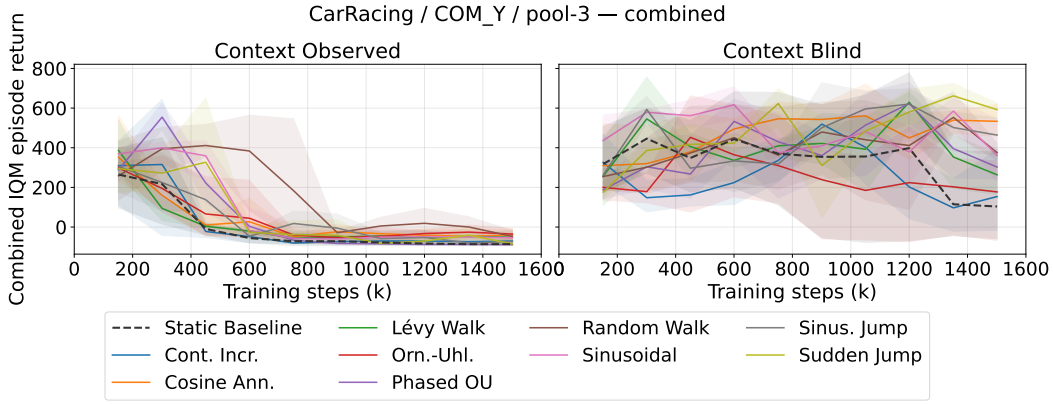


Figure 46: CarRacing / COM_Y (lateral CoM) / pool-3 — **combined** IQM (all contexts, 5 seeds).

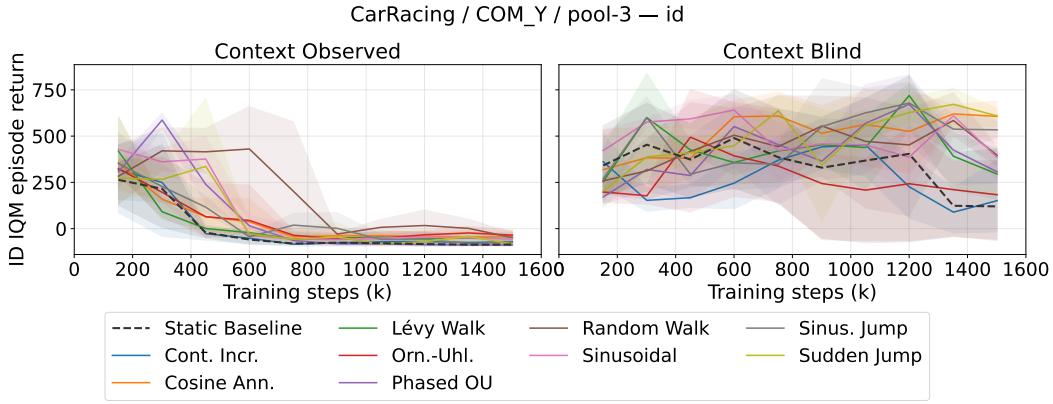


Figure 47: CarRacing / COM_Y / pool-3 — **ID** IQM (in-distribution contexts only, 5 seeds).

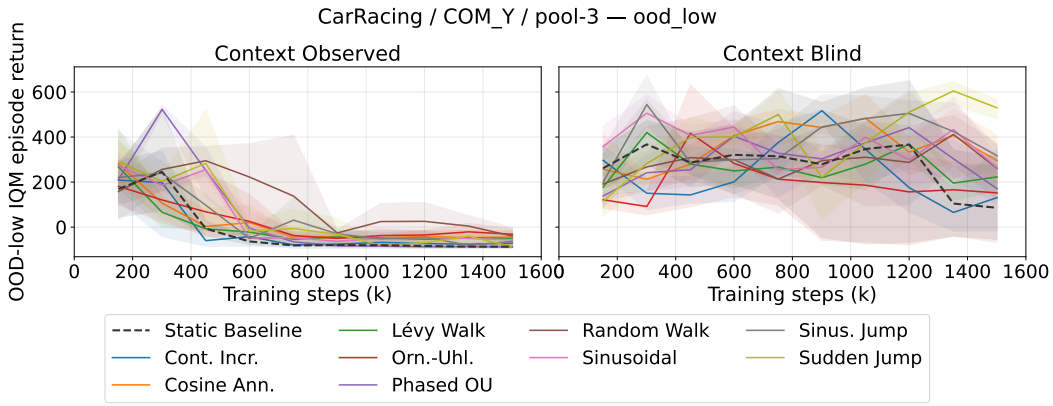


Figure 48: CarRacing / COM_Y / pool-3 — **OOD-low** IQM (lateral CoM shifted below training range, 5 seeds).

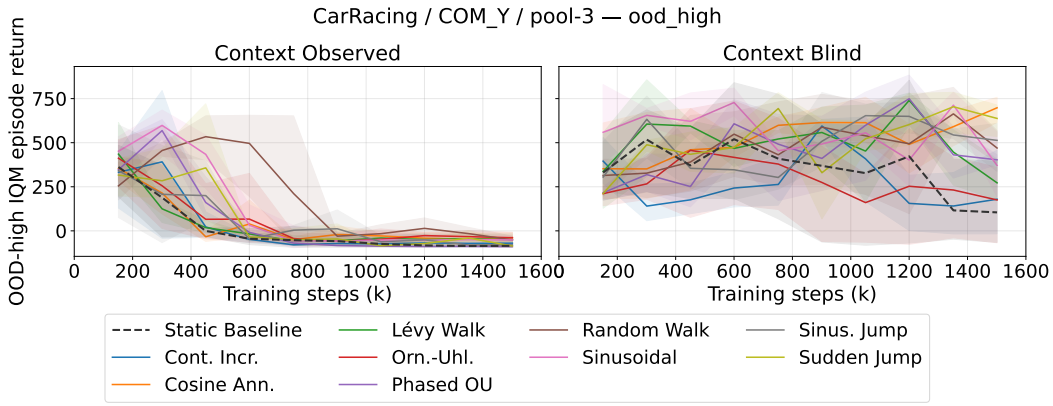


Figure 49: CarRacing / COM_Y / pool-3 — **OOD-high** IQM (lateral CoM shifted above training range, 5 seeds).