Q-SHAPED OPTIONS FOR HIERARCHICAL REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning to tackle long-horizon, goal-conditioned tasks requires an agent to reason over extended timescales and act across a broad range of states. This is particularly difficult when the agent must do so entirely from offline data, without online interaction. In principle, Hierarchical Reinforcement Learning (HRL) addresses both challenges: action (temporal) abstraction reduces the effective decision horizon, while the resulting decomposition of primitive action selection into a hierarchy of decisions then permits coarser state (spatial) abstractions than in a non-hierarchical policy, as each level can discard different information irrelevant to its part of the decision process. However, realising these two benefits depends on learning an appropriate action abstraction, and current HRL methods fail in one of two ways. Some algorithms learn an action abstraction that discards distinctions between options that are needed for optimal control, undermining hierarchy altogether, while others retain unnecessary distinctions, which preserves horizon reduction, but forfeits coarser state abstraction. In this work, we characterise three desiderata for an action abstraction, and introduce Q-Shaped Options (QSO), which addresses all three. QSO leverages a prior architecture with distinct state-value functions, Q functions and policies at each level of the hierarchy to learn the action abstraction between consecutive levels as a shared encoder through their respective Q functions. The low-level Q function uses the action abstraction as a goal, encouraging it to retain distinctions between options necessary for optimal control. The high-level Q function uses the action abstraction as an action, encouraging it to discard unnecessary distinctions between options. Across offline goal-conditioned locomotion and manipulation environments, QSO learns semantically meaningful option spaces and outperforms baselines, achieving non-zero performance in tasks where all other evaluated algorithms fail. Our code is open-sourced.

1 INTRODUCTION

Learning to tackle long-horizon, goal-conditioned (Kaelbling, 1993; Schaul et al., 2015) tasks requires an agent to reason over extended timescales and act across a broad range of states (Sutton et al., 1999; Park et al., 2026). In two-tiered goal-conditioned Hierarchical Reinforcement Learning (HRL) a high-level policy outputs a learned high-level action or option (Sutton et al., 1999) conditioned on the current state and goal state; a low-level policy then conditions on this option as well as the current state to output a primitive action (Figure 1a). In maze navigation, for example, options might be reaching "a point two metres down" or "the other side of a corner", while primitive actions could be applying torques to individual joints.

In principle, HRL (Barto and Mahadevan, 2003) addresses both challenges through the interaction between action (temporal) and state (spatial) abstractions (Konidaris, 2019). Action abstraction firstly reduces the effective decision horizon, easing credit assignment (Park et al., 2026): the high-level policy acts only intermittently, e.g. every k steps, while the low-level policy pursues the next option rather than the distant goal. Decomposing primitive action selection into a hierarchy of decisions (Figure 1a) also permits coarser state abstractions than in a non-hierarchical policy, as each level can discard information irrelevant to its part of the decision (Konidaris, 2019; Konidaris et al., 2018). For example, to output "a point two metres down" the high-level policy can ignore the agent's pose, which is too fine-grained for such high-level control, while the low-level can ignore the agent's coordinates, since moving down requires the same control wherever the agent is. Intuitively, this

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

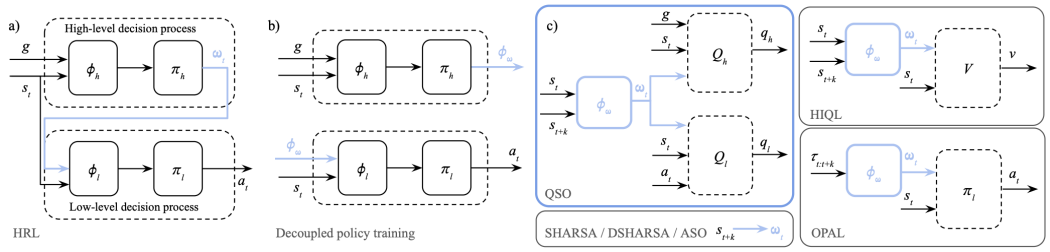

Figure 1: (**a**) **HRL** permits distinct state abstractions $\phi_h(s_t, g)$ and $\phi_l(s_t, \omega_t)$ at each level of the hierarchy. (**b**) **Decoupled policy training**. (**c**) Different methods for learning the action abstraction $\omega_t \in \Omega$.

enables experience aggregation during learning and behaviour reuse at deployment, as, for example, the high-level can aggregate all examples at a certain coordinate, irrespective of pose, and the agent can reach "a point two metres down", regardless of where in the maze it is. However, realising these benefits of horizon reduction and data aggregation depends on learning an appropriate action abstraction, and current HRL methods fail in one of two ways. Some algorithms, such as HIQL (Park et al., 2023), which relies on the action-independent state-value function to learn the action abstraction, can discard distinctions between options that are needed for optimal control (Li et al., 2006). This limits the quality of the recoverable policy and undermines hierarchy altogether. Others, like OPAL (Ajay et al., 2020), which encodes the entire k-step state-action trajectory sequence, learn an action abstraction that retains distinctions between options that are irrelevant for control (Ravindran and Barto, 2002). While this preserves horizon reduction, it forfeits coarser state abstractions as the high-level policy has distinct outputs and the low-level distinct inputs for functionally equivalent behaviour.

In this work, we introduce three desiderata for the action abstraction to realise these two benefits of hierarchy. It should (i) represent "achievement" instead of execution, encoding only the outcome of a trajectory segment and not the potentially suboptimal actions that produced it; (ii) retain distinctions between options necessary for optimal control, while also (iii) discarding unnecessary distinctions between options. Based on these desiderata, we introduce Q-Shaped Options (QSO), which builds on a prior architecture (Park et al., 2026) with separate state-value functions, Q functions and policies at each level of the hierarchy to learn the action abstraction between the two consecutive levels as a shared encoder to their respective Q functions. To our knowledge, QSO is the first HRL algorithm to learn the action abstraction via Q functions.

During training, the action abstraction encodes actions for the high-level decision process and goals for the low-level decision process. To address the first desideratum, we encode only the current and target states. Learning the abstraction via the low-level Q function addresses the second desideratum, encouraging different representations for options with different action values (Li et al., 2006), while learning via the high-level Q function addresses the third, providing an inductive bias towards discarding unnecessary distinctions between functionally equivalent behaviour. Across offline (Levine et al., 2020; Lange et al., 2012) goal-conditioned locomotion and manipulation environments, QSO learns semantically meaningful option spaces (Figure 2 and Figures 4, 5, 6 in Appendix G) and outperforms baselines (Table 2), achieving non-zero performance in tasks where all other evaluated algorithms fail (Table 7 in Appendix G). We summarise our contributions as follows.

1. We provide a formal definition of a control-sufficient action abstraction and prove that shaping the action abstraction via the low-level Q function satisfies control sufficiency.
2. We extend a Probably Approximately Correct bound on offline error in a finite-state and finite-action MDP to motivate discarding unnecessary distinctions between options.
3. We develop Q-Shaped Options (QSO), to our knowledge the first HRL algorithm to learn the action abstraction through a Q function, yielding a control-sufficient action abstraction that also discards unnecessary distinctions between options.

2 Related Works

As summarised in Table 1, prior works differs substantially in how they define or learn the action abstraction. Algorithms differ in what they encode to learn the action abstraction, ranging from a

Algorithm	Option	Method	Preserves	Discards
HIQL	$\phi_\omega(s_t, s_{t+k})$	V	No	Yes
OPAL	$\phi_\omega(\tau_{t:t+k})$	β-VAE (Surrogate Objective)	Yes	No
SHARSA	s_{t+k}	-	Yes	No
DSHARSA	s_{t+k}	-	Yes	No
ASO	s_{t+k}	-	Yes	No
QSO	$\phi_\omega(s_t, s_{t+k})$	Q_h and Q_l	Yes	Yes

Table 1: **Properties of prior action abstractions**: how the **option** is defined during training; the **method** used to learn ϕ_ω; whether the action abstraction objective encourages preserving distinctions between options **necessary for optimal control** and whether the abstraction empirically exhibits **discarding unnecessary distinctions** between options.

single target state s_{t+k} to the full k-step state-action trajectory segment $\tau_{t:t+k}$ that could be used to define the option, as well as the method in which the abstraction is learned.

HIQL (Park et al., 2023) (Figure 1, top right and Figure 2a) learns the action abstraction as an encoder to its single action-free state-value function $V(s_t, \phi_\omega(s_t, g))$. However, value equivalence does not imply action equivalence (Li et al., 2006) (two state-goal pairs can share a value, and hence option, while requiring entirely different immediate actions), meaning that the abstraction can discard information necessary for optimal control.

SHARSA and DSHARSA (Park et al., 2026), simply bypass learning an action abstraction by using subgoal options $\omega_t = s_{t+k}$ (Konidaris, 2019), which specify the entire state that the low-level policy should reach. The "encoder" is just an identity mapping. While these "options" retain the information required for low-level control, they distinguish functionally equivalent behaviour occurring at different points in the state space (Figure 2c). This limits data aggregation for learning and behaviour reuse at deployment. For example, if an agent has learned to "reach coordinate (a,b)" at one point in the state space, this does not necessarily mean that it can then "reach coordinate (c,d)", even though it might just have to reach "a point two metres down" in both situations.

OPAL (Ajay et al., 2020) (Figure 1, bottom right and Figure 2b) uses a surrogate objective to learn an option space $\omega_t \sim \phi_\omega(\omega_t \mid \tau_{t:t+k})$ with a β-VAE trained to reconstruct the entire action sequence. This means that the option encodes how a segment was executed, instead of its outcome. Similarly to SHARSA and DSHARSA, such an encoding hinders coarser high-level state abstraction (Konidaris, 2019) as two trajectory segments with the same outcome may be represented differently just because their executions differ (Figure 2), again making it hard to recombine outcomes with a different execution from somewhere else in the state space.

RL with Action Chunking (Zhao et al., 2023), QC (Li et al., 2026), which also provides temporal abstraction, is closely related to our work, but rather than having a multi-level structure, a single non-hierarchical policy directly outputs an action chunk $\pi(a_{t:t+k} \mid s_t, g)$. This prevents learning coarser state abstractions, and hence hinders the reuse of experience across the state space.

3 Preliminaries

Problem Setting We consider a goal-conditioned Markov Decision Process (MDP) (Sutton et al., 1998; Schaul et al., 2015) defined by the tuple $\mathcal{M} := (p_{s_0}, p_g, \mathcal{S}, \mathcal{G}, \mathcal{A}, \mathcal{T}, r, \bar{\gamma})$, where p_{s_0} is the initial state distribution, p_g is the goal distribution, $\mathcal{S}$, $\mathcal{G}$ and $\mathcal{A}$ respectively denote the state, goal and action spaces with $\mathcal{G} = \mathcal{S}$, and $\mathcal{T}$ is the transition function. $r(s, g) := -(1 - \beta(s, g))$ and $\bar{\gamma}(s, g) := \gamma \cdot (1 - \beta(s, g))$, where $\gamma \in [0, 1)$ is a constant, are the reward and discount or continuation functions (Andrychowicz et al., 2017; Sutton et al., 2011; White et al., 2012) defined through a goal-achievement indicator $\beta : \mathcal{S} \times \mathcal{G} \to \{0, 1\}$ (specified per environment in Appendix G): the agent receives -1 per step until the goal is reached, after which the return terminates. At the beginning of the episode, a state s_0 is sampled from p_{s_0}, and a goal g is sampled from p_g. The goal is fixed for the entire episode. At each timestep $t \geq 0$, an agent takes an action a_t conditioned on its current state s_t and goal state g, and deterministically transitions to a new state $s_{t+1} = \mathcal{T}(s_t, a_t)$; we adopt the fully observed deterministic MDP framework as used in Park et al. (2025) for simplicity. Our aim is to learn a policy $\pi : \mathcal{S} \times \mathcal{G} \to \Delta(\mathcal{A})$, where $\Delta(\mathcal{A})$ denotes the space of probability distributions over $\mathcal{A}$, from a fixed dataset $\mathcal{D}$ of N state-action trajectories of length H, collected by an arbitrary

policy, that maximises $\mathbb{E}_\pi[\sum_{t=0}^{\infty}(\prod_{i=0}^{t-1}\bar{\gamma}(s_i,g))\,r(s_t,g)]$. The return is bounded in $[-1/(1-\gamma),0]$ and is maximised by minimising the discounted time to reach the goal. In what follows we write the constant γ in place of $\bar{\gamma}(s,g)$ and omit goal-termination masking for clarity.

Hierarchical Reinforcement Learning In two-tiered offline goal-conditioned HRL (Barto and Mahadevan, 2003; Park et al., 2023) the aim is to learn a high-level policy $\pi_h(\omega_t \mid s_t, g)$ that outputs an option $\omega_t \in \Omega$ conditioned on the current state $s_t \in \mathcal{S}$ and goal $g \in \mathcal{G}$; a low-level policy $\pi_l(a_t \mid s_t, \omega_t)$ then conditions on this option as well as the current state to output a low-level action $a_t \in \mathcal{A}$ (Figure 1a), such that the overall policy is defined by:

$$\pi(a_t \mid s_t, g) := \int_{\omega_t \in \Omega} \pi_l(a_t \mid s_t, \omega_t)\pi_h(\omega_t \mid s_t, g)\mathrm{d}\omega_t,$$

where Ω denotes the option space.

While options may have fixed or variable temporal extent, in this work we focus on fixed-length options, which do not collapse to degenerate solutions (Bacon et al., 2017) and simplify the separation of the two decision processes with a fixed temporal and discount relationship. During training, an action abstraction ϕ_ω is somehow learned, and used to encode (a subset of) the k-step state-action trajectory segment $\tau_{t:t+k} = (s_t, a_t, \cdots s_{t+k})$ into an option $\omega_t = \phi_\omega(\tau_{t:t+k}) \in \Omega$. The high-level policy is trained to output this option $\omega_t = \phi_\omega(\tau_{t:t+k})$, while the low-level policy conditions on the option to output the primitive action. At deployment, the action abstraction, i.e. encoder ϕ_ω, is no longer required, since the two levels communicate entirely within the option space. The option, ω_t, which the low-level conditions on, is output by the high-level policy.

4 Q-Shaped Options

Our aim is to learn an action abstraction ϕ_ω that (i) represents "achievement" instead of a particular execution; (ii) retains distinctions between options necessary for optimal control, while also (iii) discarding unnecessary distinctions between options.

4.1 Desiderata

The action abstraction should represent "achievement" instead of a particular execution.

To enable two different executions with the same outcome to map to the same option, we restrict the action abstraction to be a function of the state-achieved-state pair $\phi_\omega(s_t, s_{t+k})$, instead of the entire state-action trajectory segment $\phi_\omega(\tau_{t:t+k})$. This prevents potentially suboptimal execution from becoming a part of the option space representation. Including the current state makes "relative achievement" rather than just "absolute achievement" representable, partly addressing our third desideratum. Note that HIQL also encodes (s_t, s_{t+k}), but motivates this choice empirically; here it follows from our desideratum.

Desideratum 1. Represent achievement instead of execution.

QSO: Action abstraction encodes (s_t, s_{t+k}) instead of $\tau_{t:t+k}$.

The action abstraction should preserve distinctions between options necessary for optimal control.

Even though we want the option space representation to discard how a state was reached in the dataset, it should retain distinctions between options necessary for optimal control. Intuitively, if two target states that require different primitive actions from the same current state have the same option space representation, a policy that conditions on this representation instead of the target state cannot be optimal for both. We formalise this in the following definition.

Definition 1. Control-sufficient action abstraction.

Let $\mathcal{G}_k(s) \subseteq \mathcal{S}$ denote the set of admissible target states for the k-step low-level decision problem from state s, and let $\Pi^\star(s, g_s) := \mathrm{argmax}_a Q^\star(s, g_s, a)$ denote the set of optimal actions of the

original MDP. An action abstraction ϕ_ω is control-sufficient if, for all $s \in \mathcal{S}$ and $g_s, g'_s \in \mathcal{G}_k(s)$,

$$\phi_\omega(s, g_s) = \phi_\omega(s, g'_s) \quad \Longrightarrow \quad \Pi^\star(s, g_s) = \Pi^\star(s, g'_s).$$

In particular, there exists a low-level policy π_l such that $\pi_l(\cdot \mid s, \phi_\omega(s, g_s))$ is supported on $\Pi^\star(s, g_s)$ for every admissible pair (s, g_s).

To address control sufficiency, we learn the action abstraction encoder through the low-level Q function, $Q_l(s_t, \phi_\omega(s_t, g_s), a_t)$. If two admissible target states require different actions from the same current state, collapsing them to the same option space representation requires $Q_l(s_t, \omega_t, \cdot)$ to represent incompatible action values. Minimising the Bellman error therefore pressures the encoder to preserve distinctions between state-target pairs that matter for low-level control. In the limit of perfectly fitting the Q function, this implies control sufficiency.

Learning the action abstraction via the Q function remains consistent with our first requirement. Because the dataset action enters Q_l as a separate argument, the action abstraction is shaped by the values of actions towards a target state rather than replicating the actions that the dataset happened to take. This means that a suboptimal action still informs the option space representation, but without fully determining it. To be explicit, two achievements can share an option if their action values are compatible, even if the dataset executed one well and the other poorly. The low-level policy conditioned on that option can draw on the better execution ("recombination").

We refer to Appendix B for the proposition and proof that low-level Q-shaping implies control sufficiency. Note that Q-shaping is stronger than control sufficiency: Q_l may preserve differences in option values or suboptimal action rankings even when they do not affect the optimal action.

Desideratum 2. Preserve distinctions between options necessary for optimal control.

QSO: Learn action abstraction ϕ_ω through the low-level Q function: $Q_l(s_t, \phi_\omega(s_t, g_s), a_t)$.

The action abstraction should discard unnecessary distinctions between options.

While preserving distinctions between options necessary for optimal control, we also want the action abstraction to discard unnecessary distinctions between options, to enable conservative extrapolation. We define conservative extrapolation as aggregating different examples across different levels of the hierarchy to produce a primitive action not observed for a particular state-goal pair.

Returning to the maze example, where, because of the level-specific state abstractions, the high-level decision process has learned to ignore pose and the low-level decision process has learned to ignore coordinates. If the dataset only contains a poor action at some state-goal pair, a high-level policy could still select the option to reach "a point two metres down" if a different pose at the same coordinate was seen reaching "a point two metres down" towards the same goal, since both state-goal pairs map to the same point through the high-level state abstraction. Similarly, the low-level policy could still output the optimal action if the same pose was seen reaching "a point two metres down" at a different coordinate, because both state-option pairs map to the same point through the low-level state abstraction. While we refer to Appendix C for the full example, the key point is that conservative extrapolation is only possible if we learn an action abstraction that discards unnecessary distinctions between options.

Proposition 1 formalises this idea by extending a Probably Approximately Correct bound on offline error to hierarchical policies with joint state and action abstraction (refer to Appendix E for the derivation).

Proposition 1. Offline Error under Joint State and Action Abstraction

For a finite-state, finite-action MDP with at most two possible next-state transitions, with probability of at least $1 - \delta$, the suboptimality in expected return, $\epsilon := \max_{s \in \mathcal{S}, g \in \mathcal{G}} \left(V^\star(s, g) - V^\pi(s, g) \right)$ of a goal-conditioned hierarchical policy learned from a dataset of N transitions

satisfies

$$\epsilon \propto \sqrt{\frac{|\mathcal{C}_h|\,|\Omega| \cdot \rho_h^{\text{rep}}}{(1-\gamma^k)^3 N}} + \sqrt{\frac{|\mathcal{C}_l|\,|\mathcal{A}|\,k^3 \cdot \rho_l^{\text{rep}}}{N}},$$

where $\phi_h : \mathcal{S} \times \mathcal{G} \to \mathcal{C}_h$ aggregates state-goal pairs requiring the same option such that $\mathcal{C}_h$ denotes the state-abstracted high-level space, and $\phi_l : \mathcal{S} \times \Omega \to \mathcal{C}_l$ aggregates state-option pairs requiring the same action such that $\mathcal{C}_l$ denotes the state-abstracted low-level space. Constants and logarithmic factors are suppressed and ρ_h and ρ_l are the concentrability coefficients at each level:

$$\rho_h^{\text{rep}} = \sup_{c_h \in \mathcal{C}_h,\, \omega \in \Omega} \frac{d^{\pi_h^\star}(c_h, \omega)}{d^{\pi_h^{\text{BC}}}(c_h, \omega)}, \qquad \rho_l^{\text{rep}} = \sup_{c_l \in \mathcal{C}_l,\, a \in \mathcal{A}} \frac{d^{\pi_l^\star}(c_l, a)}{d^{\pi_l^{\text{BC}}}(c_l, a)},$$

where d^π is underlying discounted occupancy measure of a policy π and occupancy measures are aggregated over each abstraction:

$$d^{\pi_h}(c_h, \omega) = \sum_{(s,g)\, \in\, \phi_h^{-1}(c_h)} d^{\pi_h}(s, g, \omega), \qquad d^{\pi_l}(c_l, a) = \sum_{(s,\omega)\, \in\, \phi_l^{-1}(c_l)} d^{\pi_l}(s, \omega, a).$$

Intuitively, the two terms respectively bound the high and low-level error terms; the concentrability coefficients ρ^{rep} measure the distribution shift between the occupancy of the optimal policy and that of the dataset: as suprema of occupancy ratios, a single state-goal or state-option pair that the optimal policy visits but the dataset does not can cause the term to go to infinity, unless aggregation lets that pair borrow mass from equivalent ones. Compressing the option space by discarding unnecessary distinctions between options reduces the bound in three ways. It reduces the high-level action space $|\Omega|$, tightens the bound $|\mathcal{C}_l| \leq |\mathcal{S}||\Omega|$ on the low-level state-abstraction space, and aggregates occupancy mass across a larger number of equivalent state-goal and state-option pairs, thereby reducing both concentrability coefficients ("conservative extrapolation"). In a continuous space, rather than making $|\Omega|$ as small as possible, this equates to mapping functionally equivalent behaviour to similar points in the option space (Abel et al., 2016; Ferns et al., 2012; Zhang et al., 2020), which reduces the effective complexities of both error terms (Munos and Szepesvári, 2008; Chen and Jiang, 2019).

We encourage discarding unnecessary distinctions between options by additionally shaping the option space through the high-level Q function $Q_h(s_t, g, \phi_\omega(s_t, s_{t+k}))$. Unlike control sufficiency, mapping functionally equivalent behaviour to similar points in the option space is not explicitly enforced by the objective. Instead, under shared function approximation and gradient-based optimisation, high-level Q-shaping provides an inductive bias towards it, as such state-target pairs frequently induce similar high-level Bellman targets, so distinctions between them provide little predictive benefit to Q_h. Empirically, we find that this is sufficient to form coherent clusters in the option space.

Discarding unnecessary distinctions between options also motivates conditioning the action abstraction on the current state, as $\phi_\omega(g_s)$ must separate two target states if they require different actions from any admissible state, whereas $\phi_\omega(s, g_s)$ only has to separate them if they do so from the same state. A single option can then represent reaching "a point two metres down" wherever the agent is.

Desideratum 3. Discard unnecessary distinctions between options.

QSO: Learn action abstraction ϕ_ω through the high-level Q function: $Q_h(s_t, g, \phi_\omega(s_t, s_{t+k}))$.

4.2 Practical Implementation

Action Abstraction and Conservative Value Learning We jointly learn the action abstraction between the two consecutive levels of the hierarchy as a shared encoder to their respective Q functions. During training, the action abstraction acts as a goal for the low-level, and an action for the high-level:

$$\min_{Q_l, Q_h, \phi_\omega} \mathcal{L}_{Q_l, \phi_\omega} + \mathcal{L}_{Q_h, \phi_\omega}$$

where the critic losses are the following, conservative Implicit Q Learning (Kostrikov et al., 2021) critic losses:

$$\mathcal{L}_{Q_l,\phi_\omega} = \mathbb{E}_{\substack{(s_t,a_t,s_{t+1},g_s)\sim\mathcal{D} \\ \omega_{t+1}=\phi_\omega(s_{t+1},g_s)}}\left[\Big(Q_l\big(s_t,\phi_\omega(s_t,g_s),a_t\big) - r_l(s_t,g_s) - \gamma V_l(s_{t+1},\omega_{t+1})\Big)^2\right],$$

$$\mathcal{L}_{Q_h,\phi_\omega} = \mathbb{E}_{(s_t,s_{t+k},g)\sim\mathcal{D}}\left[\Big(Q_h\big(s_t,g,\phi_\omega(s_t,s_{t+k})\big) - R_h(s_{t:t+k},g) - \gamma V_h(s_{t+k},g)\Big)^2\right],$$

and the normalised k-step option return is $R_h(s_{t:t+k},g) = \frac{1-\gamma}{1-\gamma^k}\sum_{t'=0}^{k-1}\gamma^{t'}\, r_h(s_{t+t'},g)$.

Full equations, dataset sampling methods and pseudocode are provided in Appendix F, and we refer to Appendix A for our choice of IQL. We use a continuous option space, parameterising the action abstraction with an MLP followed by length normalisation (Park et al., 2023) onto a hypersphere. Unlike a discrete option space, this avoids imposing arbitrary boundaries between semantically similar options, while the length normalisation keeps proposed options in-distribution for the low-level policy. At deployment, the high-level policy resamples an option at every timestep, allowing the agent's intent to slowly shift across the option space (Figure 7 in Appendix G). We use the same discount factor at both levels so that each level looks $1/(1-\gamma)$ of its own decisions ahead, and normalise the k-step option return so that a full-length option has the same cost as a single primitive action. This keeps the two Q functions on comparable scales and hence avoids a hyperparameter to balance their losses. Ideally, $k = 1/(1-\gamma)$ so that one option spans exactly the horizon that the low-level reasons over, but, in practice and as in prior work, k is bounded by the data (Park et al., 2023; 2026; Li et al., 2026). In Appendix G, Figures 8, 9 and 10 show performance and option embeddings when ablating Q function gradient flow, while Figure 20 shows performance when ablating k.

Conservative Policy Extraction Given the learned action abstraction and value functions, we extract a policy at each level using conservative policy extraction algorithms (Peng et al., 2019; Ghasemipour et al., 2021). We refer to Appendix A for our choice of policy extraction algorithms. The high-level policy uses Rejection Sampling (RS) over an advantage-weighted behaviour cloning (BC) flow policy; the low-level uses BC alone. We use advantage-weighting at the high-level because its dataset contains a large proportion of random goals, needed to cover the otherwise out-of-distribution state-goal pairs faced at deployment (Appendix F). While pure BC learns to ignore the goals, advantage-weighting restores goal-conditioning and leads to better proposals for RS. This issue does not arise at the low-level, for which we sampled almost no random target states. In fact, we empirically found that advantage-weighting and RS degraded low-level performance by substantially reducing the effective batch size and exploiting errors in the Q function. Even though the low-level is trained with BC, we still get conservative extrapolation, as the high-level policy can pair a state with a better option than observed at that state during training; the low-level then pursues the option with behaviour learned elsewhere in the dataset. Figure 18 and Table 9 in Appendix G show performance and goal sensitivity when ablating high-level policy extraction.

5 Experiments

We first evaluate QSO's downstream offline RL performance, which implicitly assesses our second and third desiderata: discarding necessary distinctions between options limits the recoverable policy, while retaining unnecessary ones prevents data aggregation. We then analyse the action abstraction directly, checking whether functionally different behaviour maps to distinct points in the option space and functionally equivalent behaviour to similar ones. Finally, we ablate QSO's components to assess their effects on the action abstraction and downstream performance.

Experimental Protocol We evaluate on the most challenging OGBench (Park et al., 2025) environments using the default datasets, as OGBench is a standard offline RL benchmark with complex, long-horizon, goal-conditioned tasks. We compare against hierarchical algorithms from Section 2: SHARSA, DSHARSA, HIQL and OPAL. To control for high-level policy extraction, we also introduce ASO, which uses advantage-weighted high-level policy extraction like QSO but bypasses learning an action abstraction, instead using absolute subgoal options like SHARSA and DSHARSA. As non-hierarchical baselines, we include IQL with DDPGBC (Fujimoto and Gu, 2021) policy extraction, the strongest non-hierarchical baseline (Park et al., 2025; 2024), and the action-chunking method QC. All baselines use the hyperparameters from their original implementations; QSO introduces no

	Non-Hierarchical		Hierarchical						QSO /
Environment	IQL	QC	HIQL	OPAL	SHARSA	DSHARSA	ASO	QSO	Best Mean
antmaze-giant-stitch-v0	0 [0, 0]	0 [0, 0]	54 [48, 60]	53 [52, 55]	68 [61, 76]	**72** [67, 77]	68 [66, 70]	68 [66, 70]	94%
antmaze-teleport-stitch-v0	47 [44, 51]	37 [32, 40]	36 [34, 39]	**69** [64, 74]	53 [52, 55]	58 [55, 62]	66 [62, 69]	54 [52, 57]	79%
humanoidmaze-giant-stitch-v0	1 [1, 1]	0 [0, 0]	6 [4, 8]	82 [78, 85]	30 [27, 33]	30 [27, 32]	44 [38, 48]	**90** [88, 92]	100%
cube-double-play-v0	20 [17, 23]	72 [70, 74]	7 [5, 9]	29 [26, 33]	72 [68, 77]	**80** [75, 84]	68 [65, 70]	70 [65, 74]	87%
cube-triple-play-v0	6 [6, 7]	**23** [19, 27]	16 [14, 18]	3 [2, 4]	14 [10, 18]	14 [13, 16]	4 [4, 4]	22 [20, 23]	92%
cube-quadruple-play-v0	0 [0, 0]	2 [1, 2]	3 [1, 5]	0 [0, 0]	0 [0, 1]	0 [0, 1]	0 [0, 0]	**4** [4, 5]	100%
scene-play-v0	48 [45, 51]	82 [81, 83]	41 [35, 46]	84 [78, 87]	92 [90, 93]	89 [87, 91]	**94** [91, 95]	85 [82, 90]	91%
puzzle-3x3-play-v0	**100** [99, 100]	81 [46, 99]	48 [46, 49]	70 [65, 74]	78 [71, 87]	85 [81, 90]	76 [72, 80]	72 [69, 75]	72%
puzzle-4x4-play-v0	71 [61, 78]	25 [23, 27]	0 [0, 1]	64 [61, 68]	74 [67, 79]	72 [69, 75]	69 [65, 75]	**92** [81, 100]	100%
puzzle-4x5-play-v0	14 [11, 15]	16 [13, 19]	4 [3, 5]	**24** [22, 25]	17 [15, 20]	18 [18, 19]	14 [11, 18]	15 [11, 20]	65%
puzzle-4x6-play-v0	8 [5, 10]	16 [14, 19]	5 [4, 6]	17 [15, 18]	14 [10, 16]	16 [14, 17]	11 [8, 15]	**18** [15, 22]	100%
Average over 11 Environments	29 [27, 30]	32 [29, 34]	20 [20, 20]	45 [44, 46]	46 [44, 49]	49 [48, 49]	47 [46, 48]	**54** [51, 56]	100%

Table 2: **Summary of results on selected OGBench environments using the default datasets.** Success rate averaged over 5 evaluation tasks after 1M training steps. Mean and 95% bootstrap confidence interval over 4 seeds.

additional hyperparameters, so we use those of SHARSA and DSHARSA. Environment, baseline and hyperparameter details are in Appendix G.

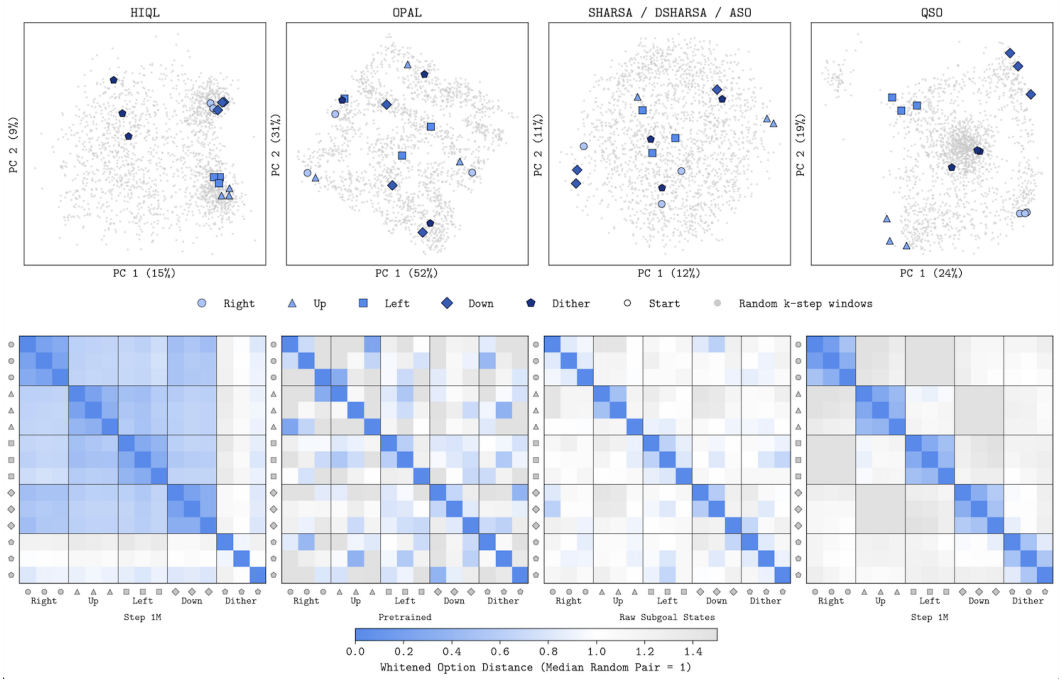

Figure 2: **PCA** (**top**) of the action abstraction for antmaze-giant fitted on random segments, showing different groups of segments with semantically similar behaviour (randomly selected from the validation dataset). **L2 distances** (**bottom**) in units of median distance between two random segments. QSO is the only algorithm that consistently groups functionally equivalent behaviours (**Desideratum 3**) while separating behaviour groups (**Desideratum 2**).

Results Table 2 shows that QSO achieves the highest average success rate across the 11 environments, 54% against 49% for the next best method (DSHARSA). Particularly notable is its performance in humanoidmaze, which has the largest action space dimension (21, against at most 8 in the other environments) where it achieves 90% against 82% for the next best method and three times that of SHARSA and DSHARSA (30%), and in puzzle-4x4, where it achieves 92% against at most 74% for any baseline. In cube-quadruple, the hardest manipulation environment, QSO is the only method with non-zero success on two of the five tasks (Table 7 in Appendix G). QSO performs consistently across environments, achieving at least 65% of the best success rate in all environments, while every other algorithm drops below 35% of the best in at least one.

Is the action abstraction semantically meaningful? We test whether QSO maps functionally equivalent behaviours to similar parts of the option space by identifying groups of semantically similar behaviours in antmaze-giant, cube-double, and cube-quadruple (Figure 2 and Figures 4, 5, and 6 in Appendix G respectively). For each behaviour, we randomly sample trajectory sequences from the validation dataset and compare PCA projections and whitened distances of their encodings. An invariant encoder should produce a clear block-diagonal similarity structure. QSO exhibits this structure across all environments, whereas HIQL also assigns high similarity between behaviour groups, and OPAL and absolute subgoal options produce low similarity within behaviour groups.

How necessary is joint option shaping by both the high and low-level Q functions? Table 3 and Figures 9 and 10 in Appendix G showing the option space indicate that joint shaping is important for both the learned action abstraction and downstream performance. Neither Q function alone recovers the clean semantic structure obtained when both are used. Shaping with the high-level only performs well in antmaze-giant but degrades significantly in cube-double. We hypothesise that for cube-double continued high-level shaping increasingly removes distinctions that are irrelevant to high-level value but necessary for fine-grained control; without the low-level Q function to preserve this information, the resulting abstractions become insufficient for the low-level policy.

Do the decoupled levels form distinct state abstractions? For the high and low-level decision processes, we measure how much each network's output varies with slow and fast moving dimensions; a larger percentage share of the output variance being associated to a particular dimension means that the network is more dependent on that dimension. We would expect the low-level decision process to focus more on high frequency (fast-moving) dimensions, and the high-level decision process on low frequency (slow-moving) ones. As expected, the two levels focus on different states (Table 8 in Appendix G). For example, the high-level state-value function depends almost entirely on slow-moving dimensions (88% for antmaze-giant and 74% for cube-double), while the low-level policy depends much more on fast-moving ones (59% and 77% respectively). We refer to Appendix G for further discussion.

	Default	Q Shaping Ablation		Option Resampling Ablation			
Environment	QSO	Low-Level Only	High-Level Only	Every Step	5% (2 Steps)	25% (7 Steps)	50% (13 Steps)
antmaze-giant-stitch-v0	68 [66, 70]	38 [38, 40]	69 [67, 72]	60 [53, 70]	68 [66, 70]	62 [61, 64]	54 [47, 60]
cube-double-play-v0	70 [65, 74]	70 [67, 73]	47 [42, 50]	70 [65, 74]	69 [63, 73]	50 [44, 55]	14 [10, 18]

Table 3: **Q shaping and option resampling ablations of QSO.** Success rate averaged over 5 evaluation tasks after 1M training steps. Mean and 95% bootstrap confidence interval over 4 seeds.

How does the option vary during deployment? Figure 7 in Appendix G shows that sampled options gradually shift across the option space during deployment: the distance between consecutively sampled options is roughly 30% of that between two randomly sampled options in antmaze-giant, and 25% of that in cube-double. Although consecutive options may partly be similar due to smoothness of the high-level policy, the above analysis showing that option-space proximity reflects semantic similarity suggests that it also corresponds to gradual changes in intent. Table 3 ablates how long the option is held for at deployment: the rapid decrease in performance for cube-double suggests that the option encodes significant information about the immediate next action.

We refer to Appendix G for further experiments ablating option representation input, goal sampling, reward labelling, policy extraction, option representation dimension and temporal horizon.

6 Discussion and Future Work

We introduce QSO, which, unlike prior algorithms, learns a control-sufficient action abstraction that discards unnecessary distinctions between options. Future work will extend QSO to do targeted exploration at the option-level in online RL. A limitation of QSO arises in stochastic environments: by representing an option retrospectively, the high-level critic evaluates the option conditioned on its realised outcome, which can lead to optimistic bias. Although we note that this issue is shared by all hindsight-based hierarchical methods (HIQL, OPAL, SHARSA and DSHARSA included), future work will address this through progress-adaptive high-level control. **Reproducibility Statement** Our code is included and open-sourced. Hyperparameters and implementation details are in Appendices F and G. The proposition and proof for Desideratum 2 are in Appendix B, and the derivation of Proposition 1 is in Appendix E.

REFERENCES

Leslie Pack Kaelbling. Learning to achieve goals. In *Ijcai*, volume 2, pages 1094–8, 1993.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pages 1312–1320. PMLR, 2015.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Seohong Park, Kevin Frans, Deepinder Mann, Benjamin Eysenbach, Aviral Kumar, and Sergey Levine. Horizon reduction makes rl scalable. *Advances in Neural Information Processing Systems*, 38:8350–8389, 2026.

Andrew G Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13(4):341–379, 2003.

George Konidaris. On the necessity of abstraction. *Current opinion in behavioral sciences*, 29:1–7, 2019.

George Konidaris, Leslie Pack Kaelbling, and Tomas Lozano-Perez. From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 61:215–289, 2018.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. Hiql: Offline goal-conditioned rl with latent states as actions. *Advances in Neural Information Processing Systems*, 36:34866–34891, 2023.

Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. *AI&M*, 1(2):3, 2006.

Anurag Ajay, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum. Opal: Offline primitive discovery for accelerating offline reinforcement learning. *arXiv preprint arXiv:2010.13611*, 2020.

Balaraman Ravindran and Andrew G Barto. Model minimization in hierarchical reinforcement learning. In *International symposium on abstraction, reformulation, and approximation*, pages 196–211. Springer, 2002.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning: State-of-the-art*, pages 45–73. Springer, 2012.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

Qiyang Li, Zhiyuan Paul Zhou, and Sergey Levine. Reinforcement learning with action chunking. *Advances in Neural Information Processing Systems*, 38:55518–55553, 2026.

Richard S Sutton, Andrew G Barto, and Andrew Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Richard S Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M Pilarski, Adam White, and Doina Precup. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th international conference on autonomous agents and multiagent systems-volume 2*, pages 761–768, 2011.

Adam White, Joseph Modayil, and Richard S Sutton. Scaling life-long off-policy learning. In *2012 ieee international conference on development and learning and epigenetic robotics (icdl)*, pages 1–6. IEEE, 2012.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations*, volume 2025, pages 94937–94982, 2025.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

David Abel, David Hershkowitz, and Michael Littman. Near optimal behavior via approximate state abstraction. In *International Conference on Machine Learning*, pages 2915–2923. PMLR, 2016.

Norman Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite markov decision processes. *arXiv preprint arXiv:1207.4114*, 2012.

Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*, 2020.

Rémi Munos and Csaba Szepesvári. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9(5), 2008.

Jinglin Chen and Nan Jiang. Information-theoretic considerations in batch reinforcement learning. In *International conference on machine learning*, pages 1042–1051. PMLR, 2019.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Seyed Kamyar Seyed Ghasemipour, Dale Schuurmans, and Shixiang Shane Gu. Emaq: Expected-max q-learning operator for simple yet effective offline and online rl. In *International Conference on Machine Learning*, pages 3682–3691. PMLR, 2021.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Seohong Park, Kevin Frans, Sergey Levine, and Aviral Kumar. Is value learning really the main bottleneck in offline rl? *Advances in Neural Information Processing Systems*, 37:79029–79056, 2024.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. Pmlr, 2014.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. Pmlr, 2018.

Sham Machandranath Kakade. *On the sample complexity of reinforcement learning*. University of London, University College London (United Kingdom), 2003.

Tor Lattimore and Marcus Hutter. Pac bounds for discounted mdps. In *International Conference on Algorithmic Learning Theory*, pages 320–334. Springer, 2012.

Arnaud Robert, Ciara Pike-Burke, and Aldo Faisal. Sample complexity of goal-conditioned hierarchical reinforcement learning. *Advances in Neural Information Processing Systems*, 36:62696–62712, 2023.

ETHICS STATEMENT

This work aims to advance machine learning, specifically offline hierarchical reinforcement learning, and is evaluated only in simulated environments using public datasets. While it may have broader societal consequences, we do not believe any require specific discussion here.

AI DISCLOSURE

All of the work is our own. We used AI to render figures, help debug code, and check spelling and grammar, but all ideas, analysis, and written content were produced by the authors, who take full responsibility for the final submission.

A Extended Preliminaries

Value Learning To learn our state-value and Q functions, we adopt Implicit Q-Learning (Kostrikov et al., 2021), a conservative value learning algorithm that evaluates the state-value function under the dataset action that leads to the maximum value rather than evaluating the Q-value under the current policy or using the maximum over actions, as is typically done in online RL (Watkins and Dayan, 1992; Mnih et al., 2015). This avoids querying the Q function on out-of-distribution actions, to which the Q function can assign erroneously high values (Levine et al., 2020). The state-value function and critic are trained with the following losses:

$$\mathcal{L}_V = \mathbb{E}_{(s_t,a_t,g)\sim\mathcal{D}}\left[\ell_\nu^2\left(V(s_t,g) - \tilde{Q}(s_t,g,a_t)\right)\right],$$
$$\mathcal{L}_Q = \mathbb{E}_{(s_t,a_t,s_{t+1},g)\sim\mathcal{D}}\left[\left(Q(s_t,g,a_t) - r(s_t,g) - \gamma V(s_{t+1},g)\right)^2\right],$$

where ℓ_κ^2 denotes the expectile loss $\ell_\nu^2(x) :=\mid \nu - \mathbf{1}\cdot(x<0) \mid \cdot x^2$. For $\nu > 0.5$, underestimating a high Q value is penalised more strongly than overestimating a low one, so $V(s_t,g)$ is pulled to the upper end of the dataset Q values.

Policy Extraction To extract our policies, we use Advantage Weighted Regression (AWR) (Peng et al., 2019) and Rejection Sampling (RS) (Ghasemipour et al., 2021), which are are conservative policy extraction algorithms that constrain the policy to sample the highest quality actions within the dataset. As the name suggests, AWR does advantage-weighted behaviour cloning of dataset actions:

$$\mathcal{L}_\pi = -\mathbb{E}_{(s_t,a_t,g)\sim\mathcal{D}}\left[\exp\left(\frac{Q(s_t,g,a_t) - V(s_t,g)}{\alpha}\right)\cdot\log\pi(a_t \mid s_t,g)\right],$$

while RS learns a behaviour-cloned policy to imitate the dataset and then defines the policy by choosing the action that maximises the Q function:

$$\pi := \text{argmax}_{a_t\in\{a_t^i\},a_t^i\sim\pi^{\text{BC}}(\cdot|s_t,g)} Q(s_t,g,a_t^i).$$

Unlike algorithms that optimise the policy by differentiating through the learned Q-function (Silver et al., 2014; Lillicrap et al., 2015; Fujimoto et al., 2018; Fujimoto and Gu, 2021), these conservative policy extraction algorithms avoid a direct source of Q function exploitation, whereby errors in the learned Q values are propagated into the policy. However, a non-hierarchical policy extracted with AWR or RS is bounded by the quality of the data at that state: where a region of the state space contains only low-quality actions, there is nothing "better" among the proposals to upweight.

B LOW-LEVEL Q-SHAPING IMPLIES CONTROL SUFFICIENCY

Proposition 2. Low-level Q-shaping implies control sufficiency.

Suppose there exists a low-level critic Q_l such that $Q_l\big(s, \phi_\omega(s, g_s), a\big) = Q^\star(s, g_s, a)$ for all $s \in \mathcal{S}$, $g_s \in \mathcal{G}_k(s)$ and $a \in \mathcal{A}$, i.e. ϕ_ω is a $Q^\star$-irrelevance abstraction in the sense of Li et al. (2006). Then ϕ_ω is control-sufficient.

Proof 1. Low-level Q-shaping implies control sufficiency.

If $\phi_\omega(s, g_s) = \phi_\omega(s, g'_s) =: \omega$, then for every a, $Q^\star(s, g_s, a) = Q_l(s, \omega, a) = Q^\star(s, g'_s, a)$, so the two targets induce the same action-value function at s and hence the same set of maximisers. Taking $\pi_l(\cdot \mid s, \omega)$ to be any distribution supported on $\operatorname{argmax}_a Q_l(s, \omega, a)$ gives the policy required by Definition 1.

C Conservative Extrapolation

To provide intuition on how joint state and action abstractions can enable conservative extrapolation, consider an agent learning a locomotion task, where its state can be decomposed into the agent's pose and coordinate $s = (p, c)$. Imagine three state-goal pairs:

$$(s_1, g, a_1) = \big(\underbrace{(p, c)}_{s_1},\ g, a_1\big), \quad (s_2, g, a_2) = \big(\underbrace{(p', c)}_{s_2},\ g, a_2\big), \quad (s_3, g', a_3) = \big(\underbrace{(p, c')}_{s_3},\ g', a_3\big).$$

The first state-goal pair is the one that the agent must act at, and its dataset action is poor (it might just dither on the spot), while the dataset actions at the second and third pair are good. Imagine that in all situations, the agent must move two metres down to reach its respective goal because the goals lie in the same direction from their respective coordinates.

A non-hierarchical policy extracted with AWR or RS never learns a useful action at (s_1, g) because the only action the dataset offers there is the poor one. We now show how a hierarchical policy with abstract relative options (such as stay "on the spot" or reach "a point two metres down") can output a good action for the first state-goal pair using conservative extrapolation (where each level only selects among actions its own data supports).

The second state-goal pair shares the coordinate and goal of the first, so the high-level abstraction (which ignores low-level pose information) groups them. This means that the high-level decision is rescued by the second state-goal pair: conservative extraction has the good option to reach "a point two metres down" available from the second state-goal pair, rather than only to stay "on the spot" from the first:

$$\phi_h(s_1, g) = \phi_h(s_2, g) = (c, g)$$
$$\pi_h(\omega \mid s_1, g) = \pi_h(\omega \mid s_2, g) = \delta_{\omega=\omega_{\text{down}}}$$

Note that the second pair supplies the right option, but its action is executed from a different pose, so it cannot be copied directly.

Similarly, the third state-goal pair then shares the pose and option of the first, so the low-level abstraction (which ignores high-level coordinate information) groups them. The low-level decision is then rescued by the third state-goal pair: conservative extraction has the good action a_3 available from the third pair, rather than only a_1 from the first:

$$\phi_l(s_1, \omega_{\text{down}}) = \phi_l(s_3, \omega_{\text{down}}) = (p, \omega_{\text{down}})$$
$$\pi_l(a \mid s_1, \omega_{\text{down}}) = \pi_l(a \mid s_3, \omega_{\text{down}}) = \delta_{a=a_3}$$

Under the hierarchical policy, the agent therefore takes a_3 at (s_1, g_1), an action never observed for that state-goal pair.

Note that this example hinges on the learning of abstract relative options so that reach "a point two metres down" is a single option rather than, for example, two coordinate-specific ones.

D State and Action Abstraction Sample Complexity in Online Goal-Conditioned RL

In this section, we should how joint state and action abstraction can reduce the sample complexity for finding an optimal goal-conditioned policy such that the error in expected return is smaller than a fixed constant ϵ from

$$N \propto \frac{|\mathcal{S}||\mathcal{G}||\mathcal{A}|}{(1-\gamma)^3\epsilon^2},$$

for a policy with no state or action abstraction, to

$$N \propto \frac{|\mathcal{C}_h||\Omega|}{(1-\gamma^k)^3\epsilon^2} + \frac{|\mathcal{C}_l||\mathcal{A}|k^3}{\epsilon^2},$$

for a policy with both state and action abstraction. We build on the works of (Kakade, 2003; Munos and Szepesvári, 2008; Lattimore and Hutter, 2012; Robert et al., 2023; Li et al., 2006).

Sample Complexity in Online GCRL In online GCRL, the aim is to find the optimal policy with the smallest number of samples or online interactions, N, such that the error in optimal return is smaller than a fixed constant ϵ. Consider the case of a discrete, discounted horizon MDP with a finite state space $\mathcal{S}$, action space $\mathcal{A}$ and discount factor $\gamma \in [0, 1)$. The Probably-Approximately Correct (PAC) Learning Kakade (2003) upper-bound sample complexity to find an ϵ optimal policy reaching a unique goal-state optimally (assuming at most two possible next-states for each state/action pair) with constant probability of $1 - \delta$ is given by Lattimore and Hutter (2012):

$$N \propto \frac{|\mathcal{S}||\mathcal{A}|}{(1-\gamma)^3\epsilon^2}.$$

Hence, the minimax sample complexity required to find an ϵ optimal policy reaching any given state (such that we have $|\mathcal{G}|$ unique goal-states) is given by:

$$N \propto \frac{|\mathcal{S}||\mathcal{G}||\mathcal{A}|}{(1-\gamma)^3\epsilon^2}. \tag{1}$$

Consider the case of a discrete finite horizon (of length H) MDP with a finite state space $\mathcal{S}$, action space $\mathcal{A}$ and discount factor $\gamma \in [0, 1)$. The minimax sample complexity to find an ϵ optimal policy reaching a unique goal-state optimally is given by:

$$N \propto \frac{|\mathcal{S}||\mathcal{G}||\mathcal{A}|H^3}{\epsilon^2}.$$

Action Abstraction Sample Complexity in Online GCRL Using a hierarchical policy, we can break the distant goal g from our current state s into options defined over an option space Ω. Hence, the high-level policy becomes

$$N \propto \frac{|\mathcal{S}||\mathcal{G}||\Omega|}{(1-\gamma^k)^3\epsilon^2},$$

where we have substituted the option-space to Equation 1, and use the environment's discount factor raised to a factor of k, assuming that the high-level policy acts every k steps.

The low-level policy has a sample complexity of

$$N \propto \frac{|\mathcal{S}||\Omega||\mathcal{A}|k^3}{\epsilon^2},$$

since $|\Omega|$ options can be executed. Hence, the overall sample complexity of the hierarchical policy is (Robert et al., 2023):

$$N \propto \frac{|\mathcal{S}||\mathcal{G}||\Omega|}{(1-\gamma^k)^3\epsilon^2} + \frac{|\mathcal{S}||\Omega||\mathcal{A}|k^3}{\epsilon^2}. \tag{2}$$

Note that in the case of no temporal abstraction, when $k = 1$, we approximately recover the original sample complexity of a non-hierarchical policy. Since the option just becomes a single primitive action, Equation 2 becomes:

$$N \propto \frac{|\mathcal{S}||\mathcal{G}||\mathcal{A}|}{(1-\gamma)^3\epsilon^2} + \frac{|\mathcal{S}||\mathcal{A}|^2}{\epsilon^2} \approx \frac{|\mathcal{S}||\mathcal{G}||\mathcal{A}|}{(1-\gamma)^3\epsilon^2},$$

where the approximation follows due to the dominating $1/(1-\gamma)^3$ factor in the first term, assuming long-horizon problems such that $\gamma \to 1$ and that the goal-space is larger or equal to the size of the action space $|\mathcal{G}| \geq |\mathcal{A}|$. Even though this might not be the case, usually the goal-space is unknown, so we train the policy to reach any state within the state-space i.e. such that $\mathcal{G} = \mathcal{S}$. Assuming that $|\mathcal{S}| \gg |\mathcal{A}|$ is a standard assumption.

In most real-world settings we expect $|\Omega| \ll |\mathcal{A}|^k$. The option space does not have to perfectly recreate the sequence of low-level actions, only contain the sufficient information to approximate the high-level value and choose the low-level action.

State and Action Abstraction Sample Complexity in Online GCRL Finally, since the high-level decision process can discard irrelevant low-level information, and the low-level decision process can discard irrelevant high-level information, state abstraction reduces the effective complexity of the state space at each level of the hierarchy such that Equation 2 becomes:

$$N \propto \frac{|\mathcal{C}_h||\Omega|}{(1-\gamma^k)^3\epsilon^2} + \frac{|\mathcal{C}_l||\mathcal{A}|k^3}{\epsilon^2}, \tag{3}$$

where $|\mathcal{C}_h|$ is the effective state-space required for the high-level decision process and $|\mathcal{C}_l|$ is the effective state-space for the low-level decision process. By definition, $|\mathcal{C}_h| \leq |\mathcal{S}||\mathcal{G}|$ and $|\mathcal{C}_l| \leq |\mathcal{S}||\Omega|$.

E State and Action Abstraction Error in Offline Goal-Conditioned RL

In offline GCRL, the aim is to bound the error ϵ given an offline dataset $\mathcal{D}$ of fixed size N.

Unlike in the online setting, where it is assumed that the agent can sample any state-action pair to learn the environment's dynamics, in offline RL, a concentrability coefficient ρ is incorporated to account for the distribution shift in data collected by the policy π^{BC}, and the data that would have been collected induced under the optimal policy $\pi^\star$. Intuitively, if the dataset does not include the states required to learn the optimal policy, the algorithm may never learn that optimal policy. The concentrability coefficient ρ is defined as:

$$\rho = \sup_{s\in\mathcal{S}, g\in\mathcal{G}, a\in\mathcal{A}} \frac{d^{\pi^\star}(s,g,a)}{d^{\pi^{\text{BC}}}(s,g,a)},$$

where $d^\pi(s,g,a)$ is the discounted occupancy measure (i.e. stationary distribution) of the policy π:

$$d^\pi(s,g,a) = (1-\gamma)\mathbb{E}_{\tau\sim\pi, s_0, g_0\sim\text{Unif}(\mathcal{G})}\left[\sum_{t=0}^{\infty}\gamma^t \mathbb{1}(s_t = s, a_t = a, g_0 = g)\right],$$

and where the trajectory is induced by the MDP and following the policy $\pi(a \mid s, g)$. If the dataset is highly exploratory and covers the optimal paths well, ρ will be small. If the dataset is narrow, or misses critical regions of the state-action space, ρ will be large.

Rearranging Equation 1 and incorporating the concentrability coefficient, the offline bound for a goal-conditioned non-hierarchical is given by:

$$\epsilon \propto \sqrt{\frac{|\mathcal{S}||\mathcal{G}||\mathcal{A}|\cdot\rho}{(1-\gamma)^3 N}}.$$

Action Abstraction Error in Offline GCRL Since the size of the offline dataset is fixed, the only way to reduce the error ϵ is to use a hierarchical policy with distinct state-goal representations.

Using a hierarchical policy, the error term becomes:

$$\epsilon \propto \sqrt{\frac{|\mathcal{S}||\mathcal{G}||\Omega|\cdot\rho_h}{(1-\gamma^k)^3 N}} + \sqrt{\frac{|\mathcal{S}||\Omega||\mathcal{A}|k^3\cdot\kappa_l}{N}},$$

where the concentrability coefficients are defined as:

$$\rho_h = \sup_{s\in\mathcal{S}, g\in\mathcal{G}, \omega\in\Omega} \frac{d^{\pi_h^\star}(s,g,\omega)}{d^{\pi_h^{\text{BC}}}(s,g,\omega)} \quad \text{and} \quad \rho_l = \sup_{s\in\mathcal{S}, \omega\in\Omega, a\in\mathcal{A}} \frac{d^{\pi_l^\star}(s,\omega,a)}{d^{\pi_l^{\text{BC}}}(s,\omega,a)}.$$

State and Action Abstraction Error in Offline GCRL Then, ignoring irrelevant low-level information for the high-level decision process and high-level information for the low-level decision process, the error term is bounded by:

$$\epsilon \propto \sqrt{\frac{|\mathcal{C}_h||\Omega|\cdot\rho_h^{\text{rep}}}{(1-\gamma^k)^3 N}} + \sqrt{\frac{|\mathcal{C}_l||\mathcal{A}|k^3\cdot\rho_l^{\text{rep}}}{N}}.$$

Again, $|\mathcal{C}_h| \leq |\mathcal{S}||\mathcal{G}|$ and $|\mathcal{C}_l| \leq |\mathcal{S}|\Omega|$.

Because the concentrability coefficients are now defined over the constrained spaces $\mathcal{C}_h$ and $\mathcal{C}_l$, i.e.

$$\rho_h^{\text{rep}} = \sup_{c_h\in\mathcal{C}_h, \omega\in\Omega} \frac{d^{\pi_h^\star}(c_h,\omega)}{d^{\pi_h^{\text{BC}}}(c_h,\omega)}, \qquad \rho_l^{\text{rep}} = \sup_{c_l\in\mathcal{C}_l, a\in\mathcal{A}} \frac{d^{\pi_l^\star}(c_l,a)}{d^{\pi_l^{\text{BC}}}(c_l,a)},$$

the probability mass of the offline dataset is aggregated across state-goal or state-option pairs that are equivalent under this information loss (i.e. for which $c_h = \phi_h(s,g)$ and $c_l = \phi_l(s,\omega)$):

$$d^{\pi_h}(c_h,\omega) = \sum_{(s,g)\,\in\,\phi_h^{-1}(c_h)} d^{\pi_h}(s,g,\omega), \qquad d^{\pi_l}(c_l,a) = \sum_{(s,\omega)\,\in\,\phi_l^{-1}(c_l)} d^{\pi_l}(s,\omega,a).$$

This reduces the likelihood of a support mismatch, where the optimal policy requires a state transition on which the dataset places zero mass. Hence,

$$\rho_h^{\text{rep}} \leq \rho_h \quad \text{and} \quad \rho_l^{\text{rep}} \leq \rho_l.$$

F Q-SHAPED OPTIONS

F.1 ACTION ABSTRACTION

Action Abstraction and Conservative Value Learning The action abstractions are learned jointly with the high-level and low-level Q functions, acting as a goal-space for the low-level Q function and an action space for the high-level Q function. The high-level value and critic losses are given by the following IQL loss:

$$\mathcal{L}_{V_h} = \mathbb{E}_{(s_t, s_{t+k}, g)\sim\mathcal{D}}\left[\ell_\nu^2\Big(V_h(s_t, g) - \tilde{Q}_h\big(s_t, g, \tilde{\phi}_\omega(s_t, s_{t+k})\big)\Big)\right] \tag{4}$$

$$\mathcal{L}_{Q_h, \phi_\omega} = \mathbb{E}_{(s_t, s_{t+k}, g)\sim\mathcal{D}}\left[\Big(Q_h\big(s_t, g, \phi_\omega(s_t, s_{t+k})\big) - R_h(s_{t:t+k}, g) - \gamma V_h(s_{t+k}, g)\Big)^2\right], \tag{5}$$

where the normalised k-step option return is

$$R_h(s_{t:t+k}, g) = \frac{1-\gamma}{1-\gamma^k}\sum_{t'=0}^{k-1}\gamma^{t'}\, r_h(s_{t+t'}, g).$$

The low-level value and critic losses are given by the following IQL loss:

$$\mathcal{L}_{V_l} = \mathbb{E}_{(s_t, a_t, g_s)\sim\mathcal{D}}\left[\ell_\nu^2\Big(V_l(s_t, \phi_\omega(s_t, g_s)) - \tilde{Q}_l\big(s_t, \tilde{\phi}_\omega(s_t, g_s), a_t\big)\Big)\right] \tag{6}$$

$$\mathcal{L}_{Q_l, \phi_\omega} = \mathbb{E}_{\substack{(s_t, a_t, s_{t+1}, g_s)\sim\mathcal{D} \\ \omega_{t+1}=\phi_\omega(s_{t+1}, g_s)}}\left[\Big(Q_l\big(s_t, \phi_\omega(s_t, g_s), a_t\big) - r_l(s_t, g_s) - \gamma V_l(s_{t+1}, \omega_{t+1})\Big)^2\right]. \tag{7}$$

Continuous Option Space We use a continuous option space encoded using a a feed-forward MLP followed by length-normalisation so that the options lie on a hypersphere of radius $\sqrt{d_\omega}$ in $\mathbb{R}^{d_\omega}$, preventing any proposed option from being significantly out-of-distribution for the low-level policy:

$$\phi_\omega(s, g) = \sqrt{d_\omega}\,\frac{f_\omega(s, g)}{\|f_\omega(s, g)\|_2},$$

where $f_\omega : \mathcal{S}\times\mathcal{G}\to\mathbb{R}^{d_\omega}$ denotes the feed-forward MLP encoder. At deployment, the options are continually resampled at every timestep.

Discounting The same discount factor γ is used at each level of the hierarchy, so that each level has the same effective horizon. For a two-level hierarchy, this means that the low-level decision process looks $1/(1-\gamma)$ actions ahead, and the higher-level decision process looks $1/(1-\gamma)$ options ahead. Since one high-level option corresponds to k low-level actions, we would ideally set

$$k = \frac{1}{1-\gamma},$$

so that the low-level decision process reasons over exactly the span of a single option. The effective horizons then compose multiplicatively: in other words, the higher level decision process covers k^2 actions, while only reasoning over $1/(1-\gamma)$ decisions.

In practice (and like prior work on hierarchical RL and QC) we find that k is bounded by the data. Since options are sampled as k-steps within a single trajectory, k must be small relative to the trajectory length T for the dataset to contain enough distinct options. For example, with our target discount factor of 0.99, we would want to use 100-step options, but `antmaze-giant-stitch-v0`'s trajectories are only 200 steps.

Reward Calibration Accumulated option rewards are normalised so that a full-length option costs the same as a single action in the low-level decision process. This means that the Q functions work on similar scales, preventing one Q function from dominating the other in magnitude (bypassing the need to introduce any hyperparameters), while options terminating early incur a proportionally smaller cost. The accumulated option reward is normalised by

$$\frac{1-\gamma}{1-\gamma^k},$$

so that a full-length option (i.e. $\beta_g(s_{t+t'}) = 0$ for $t' \in \{0, \cdots, t-1\}$) costs exactly -1, identical to the cost of a single action in the low-level decision process.

F.2 State Abstractions

Rather than introducing an explicit representation-learning objective, we induce meaningful state abstractions through level-specific datasets. At the high-level, we sample a broad mixture of local, trajectory, and random goals, encouraging it to generalise across a broad combinatorial coverage of state-goal pairs; at the low-level, we predominantly sample nearby trajectory target states, encouraging precise control over the option horizon. We replace the stricter reward-labelling criterion used by prior work (Park et al., 2025), which requires an exact match with the goal on every state dimension with the environment success criterion at the high-level, avoiding distinctions between states that are equivalent for the task. Note that, to isolate the effect of action abstraction and because we find that it improves all hierarchical baselines (Figure 14 in Appendix G) we apply this change consistently in our experiments. At the low-level, where we want to focus on fine-grained control, we retain the stricter criterion. We ablate using the environment criterion at both levels: in Appendix G, Figure 12 shows performance ablating high-level goal sampling, while Figures 13, 16 and 17 show performance and option embeddings when ablating reward labelling. Table 8 quantifies state abstraction.

Goal Sampling During training, goals are sampling according to the following, where $\tau = (s_0, \dots, s_T)$ denotes the T-length trajectory containing s_t and $\mathrm{Unif}_{\mathcal{D}}$ denotes a uniform distribution over all states in the dataset, and the weights w of each of the three sampling components sum to 1. As usual, k denotes the fixed option length.

High-level Decision Process

For the high-level decision process, goals are sampled according to

$$p_h(g \mid s_t) = w_h^{\mathrm{opt}}\, p_h^{\mathrm{opt}}(g \mid s_t) + w_h^{\mathrm{traj}}\, p_h^{\mathrm{traj}}(g \mid s_t) + w_h^{\mathrm{rand}}\, \mathrm{Unif}_{\mathcal{D}}(g), \tag{8}$$

where the first component samples a goal uniformly from within the option window,

$$p^{\mathrm{opt}}(g \mid s_t) = \mathcal{U}\big(\{\, \min(t+1,\, T),\ \dots,\ \min(t+k,\, T)\,\}\big),$$

the second component samples the goal uniformly from the remainder of the trajectory beyond the option window,

$$p_h^{\mathrm{traj}}(g \mid s_t) = \mathcal{U}\big(\{\, \min(t+k,\, T),\ \dots,\ T\,\}\big),$$

and the third samples goals uniformly from the entire dataset.

The first component ensures that a fraction of every batch returns success, the second component provides goals that are distant but reachable by dataset stitching, and the third component provides examples of the out-of-distribution combinations of state-goal pairs that the high-level policy will face at deployment.

Low-level Decision Process

For the low-level decision process, goals are sampled according to

$$p_l(g_s \mid s_t) = w_l^{\mathrm{cur}}\, \delta_{s_t}(g_s) + w_l^{\mathrm{traj}}\, p_l^{\mathrm{traj}}(g_s \mid s_t) + w_l^{\mathrm{rand}}\, \mathrm{Unif}_{\mathcal{D}}(g_s), \tag{9}$$

where the first component places all mass on the current state, the second geometrically samples a state a number of steps ahead,

$$p_l^{\mathrm{traj}}(g_s \mid s_t): \quad g_s = s_{\min(t+\kappa,\, T)}, \qquad \kappa \sim \mathrm{Geom}\big(\tfrac{1}{k}\big)\,,$$

so that the expected look-ahead is k steps and the third samples uniformly from the entire dataset.

The first component again ensures that a fraction of every batch returns a success, the second component acts in accordance with the option length of the high-level decision process, and the third component calibrates the value functions across the state space, making the learned values neater to visualise, but is unnecessary for algorithm functioning, since, even at deployment, the low-level value functions are never queried at such states.

Reward Labelling We use the environment success criterion (goal-achievement indicator) for the high-level

$$\beta^h(s, g) = \beta(s, g),$$

specified per-environment in Appendix G. For the low-level, we follow prior work and use a stricter success criterion that requires an exact match on every state dimension:

$$\beta^l(s, g_s) = \mathbb{1}\{s = g_s\}.$$

F.3 POLICY EXTRACTION

The high-level policy is defined using Rejection Sampling of an advantage-weighted behaviour cloning policy, and the low-level simply as a behaviour cloning policy:

$$\pi_h(\omega \mid s_t, g) := \text{argmax}_{\omega \in \{\omega^i\}, \omega^i \sim \pi_h^{\text{AWR}}(\cdot \mid s_t, g)} Q_h(s_t, g, \omega^i),$$
$$\pi_l(a_t \mid s_t, \omega) := \pi_l^{\text{BC}}(a_t \mid s_t, \omega)$$

High-Level Policy Extraction The high-level policy is fitted using the following advantage-weighted flow-matching loss:

$$\mathcal{L}_{\pi_h^{\text{AWR}}} = \mathbb{E}_{\substack{(s_t, s_{t+k}, g) \sim \mathcal{D},\ z \sim \mathcal{N}(0, I_{d_\omega}) \\ u \sim \text{Unif}([0,1]),\ \omega = \phi_\omega(s_t, s_{t+k}),\ \omega^u = (1-u)z + u\,\omega}} \left[\exp\Big(Q_h(s_t, g, \omega) - V_h(s_t, g) \Big) \cdot \left\| v_{\pi_h^{\text{AWR}}}\big(s_t, g, u, \omega^u\big) - (\omega - z) \right\|_2^2 \right]. \tag{10}$$

Low-Level Policy Extraction The low-level policy is fitted using the following behaviour-cloning flow-matching loss:

$$\mathcal{L}_{\pi_l^{\text{BC}}} = \mathbb{E}_{\substack{(s_t, a_t, s_{t+k}) \sim \mathcal{D},\ z \sim \mathcal{N}(0, I_d) \\ u \sim \text{Unif}([0,1]),\ a^u = (1-u)z + u\,a_t}} \left[\left\| v_{\pi_l}^{\text{BC}}\big(s_t, \phi_\omega(s_t, s_{t+k}), u, a^u\big) - (a_t - z) \right\|_2^2 \right]. \tag{11}$$

F.4 ALGORITHM

Algorithm 1 Q-Shaped Options (QSO)

Initialise option representation $\phi_\omega(s_t, g_s)$, high-level value $V_h(s_t, g)$, critic $Q_h(s_t, g, \omega)$ and flow policy $\pi_h(\omega \mid s, g)$, low-level value $V_l(s_t, \omega)$, critic $Q_l(s_t, \omega, a_t)$ and flow policy $\pi_l(a_t \mid s, \omega)$ and target option and critic networks $\tilde{\phi}_\omega, \tilde{Q}_h, \tilde{Q}_l$.

▷ **Training**
repeat
  ▷ Dataset Sampling and Reward Labelling.
  Sample a batch of states from $\mathcal{D}$. Determine a_t, s_{t+1}, s_{t+k}.
  Sample high-level goals g and low-level targets g_s from Distributions 8 and 9.
  Determine rewards $r_l(s_{t+1}, g_s)$ and $r_h(s_{t:t+k}, g_s)$.

  ▷ QSO Action Abstraction via Conservative Value Learning.
  Encode high-level options $\omega \leftarrow \phi_\omega(s_t, s_{t+k})$ and target options $\tilde{\omega} \leftarrow \tilde{\phi}_\omega(s_t, s_{t+k})$.
  Encode low-level options $\phi_\omega(s_t, g_s)$ and target options $\tilde{\phi}_\omega(s_t, g_s)$.
  Train V_h and V_l using Equations 4 and 6.
  Train Q_h and Q_l and ϕ_ω using Equations 5 and 7.
  Update $\tilde{\phi}_\omega, \tilde{Q}_h, \tilde{Q}_h$ using exponential moving averages.

  ▷ Conservative Policy Extraction.
  Train π_h^{AWR} and π_l^{BC} using Equations 10 and 11.
until convergence

▷ **Deployment**
while episode not terminated **do**
  ▷ High-level Rejection Sampling.
  Sample $\omega^1 \cdots \omega^N \sim \pi_h^{\text{AWR}}(\cdot \mid s_t, g)$.
  Select $\omega \leftarrow \text{argmax}_{\omega^1 \ldots \omega^N} Q_h(s_t, g, \omega^i)$.

  ▷ Low-level Behaviour Cloning.
  Sample $a_t \sim \pi_l^{\text{BC}}(\cdot \mid s_t, \omega)$.
end while

G EXPERIMENTS

G.1 ENVIRONMENTS

Environment	State Dim.	Action Dim.	Data Trajectory Length	Dataset Size	Max. Episode Length
antmaze-giant-stitch-v0	29	8	200	1M	1000
antmaze-teleport-stitch-v0	29	8	200	1M	1000
humanoidmaze-giant-stitch-v0	69	21	400	4M	4000
cube-double-play-v0	37	5	1000	1M	500
cube-triple-play-v0	46	5	1000	3M	1000
cube-quadruple-play-v0	55	5	1000	5M	1000
puzzle-3x3-play-v0	55	5	1000	1M	500
puzzle-4x4-play-v0	83	5	1000	1M	500
puzzle-4x5-play-v0	99	5	1000	3M	1000
puzzle-4x6-play-v0	115	5	1000	5M	1000
scene-play-v0	40	5	1000	1M	750

Table 4: **OGBench environment and dataset characteristics.**

Maze Environments

For `antmaze-giant-stitch-v0` and `humanoidmaze-giant-stitch-v0`, high-level success is defined as the agent lying within ϵ of the goal's xy coordinates, where $x(s) \in \mathbb{R}^2$ denotes a state s's xy coordinates:

$$\beta(s, g) = \mathbb{1}\left\{\|x(s) - x(g)\|_2 \leq \epsilon\right\}.$$

Cube Environments

For `cube-double-play-v0`, `cube-triple-play-v0` and `cube-quadruple-play-v0`, high-level success is defined as every cube being within ϵ of its goal xyz position, where $c_i(s) \in \mathbb{R}^3$ denotes cube i's xyz coordinates in state s, and n is the number of cubes in the environment:

$$\beta(s, g) = \prod_{i=1}^{n} \mathbb{1}\left\{\|c_i(s) - c_i(g)\|_2 \leq \epsilon\right\}.$$

Puzzle Environments

For `puzzle-3x3-play-v0`, `puzzle-4x4-play-v0`, `puzzle-4x5-play-v0` and `puzzle-4x6-play-v0`, high-level success is defined as every button matching its goal configuration, where $b_i(s) \in \mathbb{R}^3$ denotes button i's configuration in state s, and n is the number of buttons in the environment:

$$\beta(s, g) = \prod_{i=1}^{n} \mathbb{1}\left\{b_i(s) = b_i(g)\right\}.$$

Scene Environment

For `scene-play-v0`, high-level success is defined as the n_c cubes, drawer and window being within ϵ_c, ϵ_d and ϵ_w of their goal positions and the n_b buttons matching their goal configurations:

$$\begin{aligned}\beta(s, g) = &\prod_{i=1}^{n_c} \mathbb{1}\left\{\|c_i(s) - c_i(g)\|_2 \leq \epsilon_c\right\} \cdot \prod_{i=1}^{n_b} \mathbb{1}\left\{b_i(s) = b_i(g)\|_2\right\} \\ &\cdot \mathbb{1}\left\{|d(s) - d(g)| \leq \epsilon_{\text{pos}}\right\} \cdot \mathbb{1}\left\{|w(s) - w(g)| \leq \epsilon_{\text{pos}}\right\}.\end{aligned}$$

G.2 BASELINES

G.2.1 ALGORITHMS

Absolute Subgoal Options: DSHARSA For DSHARSA Park et al. (2026), both the high and low-level value and critic functions are learned using IQL:

$$\mathcal{L}_{V_h} = \mathbb{E}_{(s_t, s_{t+k}, g)\sim\mathcal{D}}\left[\ell_\nu^2\left(V_h(s_t, g) - \tilde{Q}_h(s_t, g, s_{t+k})\right)\right],$$
$$\mathcal{L}_{Q_h} = \mathbb{E}_{(s_t, s_{t+k}, g)\sim\mathcal{D}}\left[\left(Q_h(s_t, g, s_{t+k}) - \sum_{t'=0}^{k-1}\gamma^{t'} r_h(s_{t+t'}, g) - \gamma^k V_h(s_{t+k}, g)\right)^2\right],$$

and

$$\mathcal{L}_{V_l} = \mathbb{E}_{(s_t, a_t, g_s)\sim\mathcal{D}}\left[\ell_\nu^2\left(V_l(s_t, g_s) - \tilde{Q}_l(s_t, g_s, a_t)\right)\right],$$
$$\mathcal{L}_{Q_l} = \mathbb{E}_{(s_t, a_t, s_{t+1}, g_s)\sim\mathcal{D}}\left[\left(Q_l(s_t, g_s, a_t) - r_l(s_{t+1}, g_s) - \gamma V_l(s_t, g_s)\right)^2\right],$$

The high-level and low-level policies are fitted using behaviour-cloning flow-matching losses:

$$\mathcal{L}_{\pi_h^{\text{BC}}} = \mathbb{E}_{\substack{(s_t, s_{t+k}, g)\sim\mathcal{D},\ z\sim\mathcal{N}(0, I_{d_s}) \\ u\sim\text{Unif}([0,1]),\ g_s = s_{t+k},\ g_s^u = (1-u)z + u\, g_s}}\left[\left\| v_{\pi_h^{\text{BC}}}\left(s_t, g, u, g_s^u\right) - (g_s - z)\right\|_2^2\right],$$

and

$$\mathcal{L}_{\pi_l^{\text{BC}}} = \mathbb{E}_{\substack{(s_t, a_t, s_{t+k})\sim\mathcal{D},\ z\sim\mathcal{N}(0, I_d) \\ u\sim\text{Unif}([0,1]),\ a^u = (1-u)z + u\, a_t}}\left[\left\| v_{\pi_l}\left(s_t, s_{t+k}, u, a^u\right) - (a_t - z)\right\|_2^2\right].$$

At deployment, both the high and low-level policies are defined with Rejection Sampling:

$$\pi_h(g_s \mid s_t, g) := \text{argmax}_{g_s\in\{g_s^i\}, g_s^i\sim\pi_h^{\text{BC}}(\cdot|s_t, g)} Q_h(s_t, g, g_s^i),$$
$$\pi_l(a_t \mid s_t, g_s) := \text{argmax}_{a_t\in\{a_t^i\}, a_t^i\sim\pi_l^{\text{BC}}(\cdot|s_t, g_s)} Q_l(s_t, g_s, a_t^i).$$

Absolute Subgoal Options: SHARSA SHARSA (Park et al., 2026) is almost identical to DSHARSA, but only uses Rejection Sampling for the high-level policy. The low-level policy is defined using pure behaviour cloning:

$$\pi_h(g_s \mid s_t, g) := \text{argmax}_{g_s\in\{g_s^i\}, g_s^i\sim\pi_h^{\text{BC}}(\cdot|s_t, g)} Q_h(s_t, g, g_s^i),$$
$$\pi_l(a_t \mid s_t, g_s) := \pi_l^{\text{BC}}(a_t \mid s_t, g_s).$$

Note that conservative extrapolation is still possible as the low-level policy might be queried at an absolute subgoal option (proposed by the high-level policy) that was unseen during training.

Absolute Subgoal Options: ASO As explained in Section 4, for QSO we use advantage-weighting to train the high-level policy. Therefore, as a baseline, we introduce a new variant of SHARSA that, like QSO, uses AWR to train the high-level policy: :

$$\mathcal{L}_{\pi_h^{\text{AWR}}} = \mathbb{E}_{\substack{(s_t, s_{t+k}, g)\sim\mathcal{D},\ z\sim\mathcal{N}(0, I_{d_s}) \\ u\sim\text{Unif}([0,1]),\ g_s = s_{t+k},\ g_s^u = (1-u)z + u\, g_s}}\left[\exp\left(Q_h(s_t, g, g_s) - V_h(s_t, g)\right)\cdot\left\| v_{\pi_h^{\text{AWR}}}\left(s_t, g, u, g_s^u\right) - (g_s - z)\right\|_2^2\right].$$

At deployment, the high-level policy is still defined using Rejection Sampling and the low-level with behaviour cloning:

$$\pi_h(g_s \mid s_t, g) := \text{argmax}_{g_s\in\{g_s^i\}, g_s^i\sim\pi_h^{\text{AWR}}(\cdot|s_t, g)} Q_h(s_t, g, g_s^i),$$
$$\pi_l(a_t \mid s_t, g_s) := \pi_l^{\text{BC}}(a_t \mid s_t, g_s).$$

Abstract Relative Options: HIQL HIQL (Park et al., 2023) shapes the abstract subgoal option space through its single, action-free state-value function $V(s, \hat{\phi}_\omega(s, g))$, which is trained using Implicit V-Learning (IVL) (Park et al., 2025):

$$\mathcal{L}_{V,\phi_\omega} = \mathbb{E}_{(s_t,a_t,g)\sim\mathcal{D},g\sim p^{\mathcal{D}}(\cdot|s,a))}\left[\ell_\nu^2\left(V(s_t,\phi_\omega(s_t,g)) - r(s_t,g_t) - \gamma\tilde{V}(s_{t+1},\tilde{\phi}_\omega(s_{t+1},g))\right)\right],$$

where $\tilde{V}$ and $\tilde{\phi}_\omega$ denote the target network and representations and ℓ_ν^2 the expectile loss. HIQL extract high and low-level Gaussian policies using AWR-like objectives:

$$\mathcal{L}_{\pi_h} = -\mathbb{E}_{(s_t,s_{t+k},g)\sim\mathcal{D},\omega=\phi_\omega(s_t,s_{t+k})}\left[\exp\left(\frac{V(s_{t+k},g) - V(s_t,g)}{\alpha}\right)\cdot\log\pi_h(\omega \mid s_t,g)\right],$$
$$\mathcal{L}_{\pi_l} = -\mathbb{E}_{(s_t,s_{t+k},g)\sim\mathcal{D},\omega=\phi_\omega(s_t,s_{t+k})}\left[\exp\left(\frac{V(s_{t+1},s_{t+k}) - V(s_t,s_{t+k})}{\alpha}\right)\cdot\log\pi_l(a_t \mid s_t,\omega)\right].$$

Abstract Options: OPAL OPAL (Ajay et al., 2020) shapes the abstract subgoal option space using a β-VAE trained to encode state-action trajectory sequences:

$$\min_{\pi_l,p_\omega,q_\omega}\mathbb{E}_{\tau\sim\mathcal{D},\omega\sim q_\omega(\cdot|\tau)}\left[\sum_{t=0}^{c-1}\log\pi_l^{\text{BC}}(a_t \mid s_t,\omega)\right] - \beta\cdot\mathrm{D}_{\text{KL}}(q_\omega(\omega \mid \tau)\,||\,p_\omega(\omega \mid s_0)),$$

where p_ω and p_ω respectively denote the option-space encoder and option-space prior. Using this learned option space to encode trajectory sequences, state-value and Q functions are trained using IQL:

$$\mathcal{L}_V = \mathbb{E}_{(\tau,g)\sim\mathcal{D},\omega\sim q_\omega(\cdot|\tau)}\left[\ell_\nu^2\left(V(s_t,g) - \tilde{Q}(s_t,g,\omega)\right)\right],$$
$$\mathcal{L}_Q = \mathbb{E}_{(\tau,g)\sim\mathcal{D},\omega\sim q_\omega(\cdot|\tau)}\left[\left(Q(s_t,g,\omega) - \sum_{t'=0}^{k-1}\gamma^{t'}r(s_{t+t'},g) - \gamma^k V(s_{t+k},g)\right)^2\right],$$

and a high-level flow-matching policy is fitted using the following behaviour-cloning flow-matching loss:

$$\mathcal{L}_{\pi_h^{\text{BC}}} = \mathbb{E}_{\substack{(\tau,g)\sim\mathcal{D},\ z\sim\mathcal{N}(0,I_{d_\omega})\\ u\sim\text{Unif}([0,1]),\ \omega\sim q_\omega(\cdot|\tau),\ \omega^u=(1-u)z+u\,\omega}}\left[\left\|v_{\pi_h^{\text{BC}}}\left(s_t,g,u,\omega^u\right) - (\omega - z)\right\|_2^2\right],$$

At deployment, the high-level policy is defined using Rejection Sampling, and the low-level with behaviour cloning:

$$\pi_h(\omega \mid s_t,g) := \text{argmax}_{\omega\in\{\omega^i\},\omega^i\sim\pi_h^{\text{BC}}(\cdot|s_t,g)}Q_h(s_t,g,\omega^i),$$
$$\pi_l(a_t \mid s_t,\omega) := \pi_l^{\text{BC}}(a_t \mid s_t,\omega).$$

In practice, the low-level policy is finetuned to output the immediate action.

Non-hierarchical: IQL IQL (Kostrikov et al., 2021) trains a value and Q function with the following losses:

$$\mathcal{L}_V = \mathbb{E}_{(s_t,a_t,g)\sim\mathcal{D}}\left[\ell_\nu^2\left(V(s_t,g) - \tilde{Q}(s_t,g,a_t)\right)\right],$$
$$\mathcal{L}_Q = \mathbb{E}_{(s_t,a_t,s_{t+1},g)\sim\mathcal{D}}\left[\left(Q(s_t,g,a_t) - r(s_{t+1},g) - \gamma V(s_t,g)\right)^2\right],$$

We then extract a Gaussian policy by maximising the Deep Deterministic Policy Gradient objective Fujimoto and Gu (2021):

$$\mathcal{L}_\pi = -\mathbb{E}_{(s_t,a_t,g)\sim\mathcal{D}}\Big[Q\big(s,\mu_\pi(s,g),g\big) + \alpha\log\pi(a \mid s,g)\Big],$$

where μ_π denotes the mean of the Gaussian policy. Following Park et al. (2024), we use DDPGBC rather than AWR for policy extraction.

Action Chunking: QC QC Li et al. (2026) trains an action-chunked Q function:

$$\mathcal{L}_Q = \mathbb{E}_{\substack{(s_t, a_{t:t+k}, s_{t+k}, g)\sim\mathcal{D} \\ a_{t+k:t+2k}\sim\pi(\cdot|s_{t+k}, g)}}\left[\left(Q\big(s_t, g, a_{t:t+k}\big) - \sum_{t'=0}^{k-1}\gamma^{t'} r(s_{t+t'}, g) - \gamma^k Q\big(s_{t+k}, g, a_{t+k:t+2k}\big)\right)^2\right].$$

It fits a flow-matching policy using behaviour-cloning:

$$\mathcal{L}_{\pi^{\text{BC}}} = \mathbb{E}_{\substack{(s_t, a_t, s_{t+k})\sim\mathcal{D},\ z\sim\mathcal{N}(0, I_d) \\ u\sim\text{Unif}([0,1]),\ a^u = (1-u)z + u\, a_{t:t+k}}}\left[\left\|v_{\pi_l}\big(s_t, s_{t+k}, u, a^u\big) - (a_{t:t+k} - z)\right\|_2^2\right].$$

The policy is defined using Rejection Sampling:

$$\pi(a_{t:t+k} \mid s_t, g) := \text{argmax}_{a_{t:t+k}\in\{a^i_{t:t+k}\}, \{a_{t:t+k}\}^i\sim\pi^{\text{BC}}(\cdot|s_t, g)} Q\big(s_t, g, a^i_{t:t+k}\big).$$

G.2.2 Dataset Sampling for Non-Hierarchical and Action Chunking Baselines

The dataset sampling intends to mirror the hierarchical ones as closely as possible, to enable the isolation of the hierarchical option space on performance as opposed to dataset sampling.

Non-hierarchical: IQL The flat baseline has no option structure, so there is no distinction between a near and a distant goal and the option window collapses to the current state. Goals are sampled according to:

$$p_{\text{flat}}(g \mid s_t) = w^{\text{cur}}\,\delta_{s_t}(g) + w^{\text{traj}}\,p^{\text{traj}}_{\text{flat}}(g \mid s_t) + w^{\text{rand}}\,\text{Unif}_{\mathcal{D}}(g),$$

where the second component samples the goal uniformly from the remainder of the trajectory beyond the current state,

$$p^{\text{traj}}_{\text{flat}}(g \mid s_t) = \mathcal{U}\big(\{\,\min(t+1,\,T),\ \ldots,\ T\,\}\big).$$

and, like for hierarchical high-level dataset sampling, the third component samples goals uniformly from the entire dataset. We use the same weighting as for the hierarchical high-level dataset sampling as opposed to the low-level one, since the single non-hierarchical must cover the full range of state-goal possibilities with one distribution, in contrast to the hierarchical case where the near and far regimes can be split across two levels.

Action Chunking: QC The action chunking baseline is flat but temporally extended, so it inherits the high-level goal distribution of Equation 8 and additionally requires the length-k action sequence beginning at s_t,

$$a_{t:t+k} = \big(a_{\min(t,\,T)},\ \ldots,\ a_{\min(t+k-1,\,T)}\big),$$

where indices are clipped to the end of the trajectory. If an agent reaches its target state before the end of the k length option chunk, actions past the target state are masked out and set to zero, so that the critic is not trained on behaviour occurring past termination.

Following original implementations of non-hierarchical methods, we use strict reward labelling.

G.3 HYPERPARAMETERS

We use the following default hyperparameters for all value networks (both Value and Q functions), flow-matching and Gaussian policies, and Rejection Sampling methods. Default learning rate, batch size, target update rate, layer normalisation, non-linearities, and representation latent dimensions follow Park et al. (2025). According to Park et al. (2023), we use 128 for HIQL. Rejection Sampling N follows Park et al. (2026). Default network sizes and actor flow steps were chosen according to available compute time and memory. For Gaussian policies, following Park et al. (2023), we use a network that outputs a mean and constant standard deviation. We use the number of steps from (Park et al., 2023), apart from for QC, where we use their suggestion of 5.

Table 5: Hyperparameters.

Hyperparameter	**Value**
Optimizier	Adam
Learning Rate	0.0003
Batch Size	1024
Target Update Rate	0.005
Representation Latent Dimension	10 (default), 128 (HIQL)
Representation MLP	[512, 512, 512]
Value Layer Normalisation	True
Value Non-linearity	GELU
Value Expectile ν (IQL)	0.7
Value MLP	[512, 512, 512, 512]
Value Layer Normalisation	True
Value Non-linearity	GELU
Actor MLP	[512, 512, 512, 512]
Actor Layer Normalisation	True
Actor Non-linearity	GELU
Actor Flow Steps	6
Actor Rejection Sampling N	32
Actor HIQL α	3.0
Actor IQL α	1.0 (default), 0.1 (`humanoidmaze`), 0.3 (`antmaze`)
Subgoal Steps k	25 (default), 100 (`humanoidmaze`), 5 (QC)
Dataset	(default), (IQL), (QC)
Discount γ	0.99 (default), 0.995/0.999 (HIQL & IQL `antmaze` / `humanoidmaze`)

		Non-Hierarchical		Hierarchical					
Environment	Task	IQL	QC	HIQL	OPAL	SHARSA	DSHARSA	ASO	QSO
antmaze-giant-stitch-v0	Task 1	0 [0, 0]	0 [0, 0]	35 [29, 39]	35 [33, 37]	59 [42, 71]	**72** [63, 86]	57 [47, 67]	57 [50, 63]
	Task 2	0 [0, 0]	0 [0, 0]	66 [60, 70]	**83** [77, 90]	74 [65, 87]	67 [60, 73]	67 [64, 69]	75 [69, 79]
	Task 3	0 [0, 0]	0 [0, 0]	42 [38, 45]	62 [52, 71]	53 [39, 66]	72 [66, 78]	66 [58, 72]	**73** [66, 81]
	Task 4	0 [0, 0]	0 [0, 0]	57 [50, 62]	17 [12, 22]	65 [57, 72]	**67** [57, 77]	65 [60, 72]	52 [43, 61]
	Task 5	0 [0, 0]	0 [0, 0]	69 [53, 86]	69 [65, 74]	**90** [85, 95]	82 [80, 85]	84 [70, 94]	82 [78, 87]
	Overall	0 [0, 0]	0 [0, 0]	54 [48, 60]	53 [52, 55]	68 [61, 76]	**72** [67, 77]	68 [66, 70]	68 [66, 70]
antmaze-teleport-stitch-v0	Task 1	35 [27, 44]	27 [18, 33]	44 [40, 48]	**61** [50, 72]	18 [12, 28]	28 [17, 38]	54 [46, 65]	34 [28, 40]
	Task 2	47 [42, 52]	38 [32, 43]	44 [41, 48]	85 [81, 91]	77 [70, 83]	**87** [83, 93]	75 [70, 80]	75 [73, 78]
	Task 3	57 [52, 62]	40 [33, 47]	34 [27, 42]	**87** [82, 90]	80 [76, 83]	74 [71, 78]	82 [78, 88]	68 [53, 79]
	Task 4	51 [40, 60]	42 [36, 50]	26 [13, 41]	50 [47, 54]	55 [51, 61]	51 [45, 58]	**59** [51, 65]	50 [45, 57]
	Task 5	48 [38, 57]	37 [27, 48]	32 [26, 38]	**61** [52, 71]	33 [26, 42]	48 [38, 58]	58 [50, 67]	46 [43, 48]
	Overall	47 [44, 51]	37 [32, 40]	36 [34, 39]	**69** [64, 74]	53 [52, 54]	58 [55, 62]	66 [62, 69]	54 [52, 57]
humanoidmaze-giant-stitch-v0	Task 1	0 [0, 0]	0 [0, 0]	5 [0, 10]	73 [63, 83]	12 [8, 18]	17 [12, 22]	22 [18, 28]	**82** [77, 89]
	Task 2	3 [1, 6]	0 [0, 0]	18 [8, 27]	78 [65, 89]	41 [33, 51]	47 [32, 69]	32 [18, 43]	**98** [95, 100]
	Task 3	1 [0, 2]	0 [0, 0]	6 [4, 7]	85 [72, 92]	42 [28, 61]	61 [58, 65]	55 [48, 62]	**91** [88, 93]
	Task 4	0 [0, 0]	0 [0, 0]	2 [0, 3]	78 [73, 83]	11 [10, 12]	8 [3, 12]	19 [17, 22]	**85** [82, 87]
	Task 5	0 [0, 0]	0 [0, 0]	1 [0, 2]	95 [92, 98]	43 [32, 61]	17 [5, 28]	92 [84, 98]	**97** [94, 99]
	Overall	1 [1, 1]	0 [0, 0]	6 [4, 8]	82 [78, 85]	30 [27, 33]	30 [27, 32]	44 [38, 48]	**90** [88, 92]

Table 6: **Full results on selected OGBench locomotion environments using default datasets.** Success rate (%) on each evaluation task after 1M offline training steps. Mean and 95% bootstrap confidence interval over 4 seeds.

G.4 Further Results

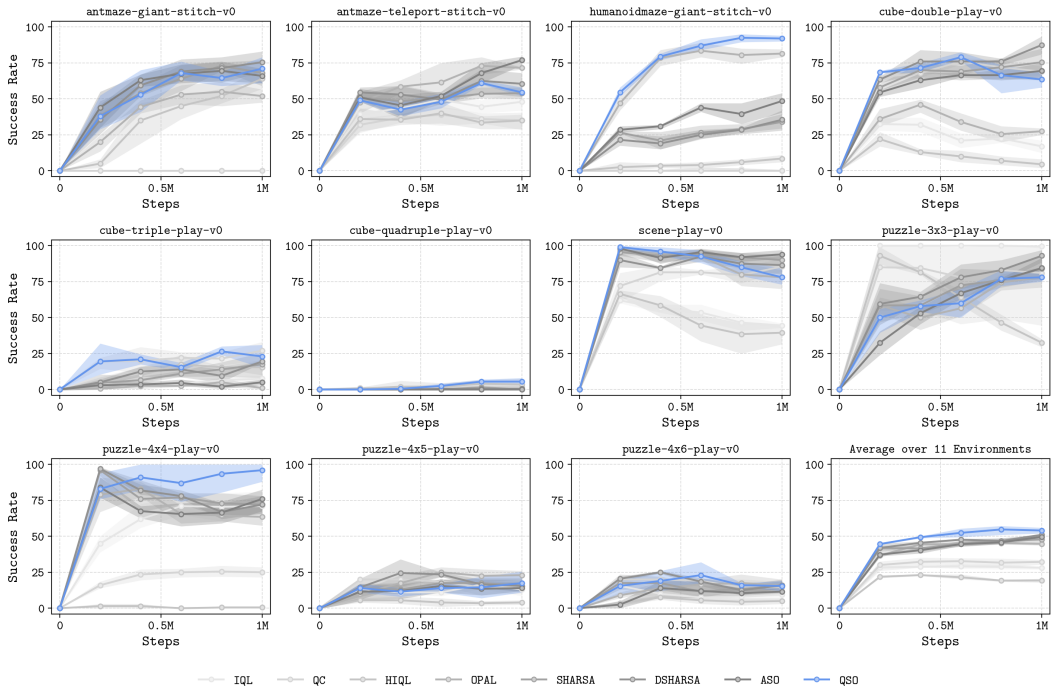

Figure 3: **Training on selected OGBench environments using the default datasets.** Success rate averaged over 5 evaluation tasks. Mean and 95% bootstrap confidence interval over 4 seeds.

		Non-Hierarchical		Hierarchical					
Environment	Task	IQL	QC	HIQL	OPAL	SHARSA	DSHARSA	ASO	QSO
cube-double-play-v0	Task 1	62 [53, 70]	90 [86, 93]	26 [20, 32]	82 [78, 87]	95 [92, 98]	98 [97, 100]	96 [92, 99]	**99** [98, 100]
	Task 2	11 [3, 18]	88 [83, 96]	2 [0, 5]	12 [4, 18]	82 [78, 86]	**93** [91, 96]	88 [87, 90]	68 [53, 78]
	Task 3	20 [16, 23]	88 [85, 90]	4 [1, 8]	11 [6, 15]	89 [87, 92]	**91** [86, 95]	85 [83, 88]	68 [62, 76]
	Task 4	2 [0, 5]	42 [38, 45]	2 [0, 3]	15 [8, 22]	41 [33, 48]	**59** [48, 68]	20 [13, 27]	53 [50, 57]
	Task 5	6 [3, 8]	54 [45, 63]	2 [0, 3]	25 [18, 32]	53 [43, 67]	58 [50, 64]	48 [40, 57]	**61** [55, 69]
	Overall	20 [17, 23]	72 [70, 74]	7 [5, 9]	29 [26, 33]	72 [68, 77]	**80** [75, 84]	68 [65, 70]	70 [65, 74]
cube-triple-play-v0	Task 1	29 [27, 32]	57 [48, 68]	**58** [51, 65]	8 [5, 12]	45 [29, 58]	44 [40, 48]	16 [13, 18]	45 [42, 48]
	Task 2	1 [0, 2]	**31** [18, 42]	7 [7, 7]	1 [0, 2]	15 [11, 21]	16 [11, 23]	2 [1, 3]	28 [27, 30]
	Task 3	0 [0, 0]	**20** [17, 23]	12 [10, 13]	2 [1, 3]	5 [3, 7]	5 [0, 10]	1 [0, 2]	17 [12, 22]
	Task 4	0 [0, 0]	4 [2, 7]	3 [0, 8]	2 [0, 5]	3 [1, 6]	6 [2, 9]	0 [0, 0]	**15** [12, 18]
	Task 5	0 [0, 0]	**6** [4, 7]	1 [0, 2]	1 [0, 2]	2 [0, 5]	1 [0, 2]	0 [0, 0]	3 [3, 3]
	Overall	6 [6, 6]	**24** [19, 27]	16 [14, 18]	3 [2, 4]	14 [10, 18]	14 [13, 16]	4 [4, 4]	22 [20, 23]
cube-quadruple-play-v0	Task 1	0 [0, 0]	4 [3, 6]	**11** [5, 19]	0 [0, 0]	2 [0, 3]	2 [0, 3]	0 [0, 0]	9 [8, 10]
	Task 2	0 [0, 0]	4 [3, 6]	4 [2, 7]	0 [0, 0]	1 [0, 2]	1 [0, 2]	0 [0, 0]	**10** [7, 13]
	Task 3	0 [0, 0]	0 [0, 0]	**1** [0, 2]	0 [0, 0]	0 [0, 0]	0 [0, 0]	0 [0, 0]	**1** [0, 2]
	Task 4	0 [0, 0]	0 [0, 0]	0 [0, 0]	0 [0, 0]	0 [0, 0]	0 [0, 0]	0 [0, 0]	**2** [0, 5]
	Task 5	0 [0, 0]	0 [0, 0]	0 [0, 0]	0 [0, 0]	0 [0, 0]	0 [0, 0]	0 [0, 0]	**1** [0, 2]
	Overall	0 [0, 0]	2 [1, 2]	3 [1, 5]	0 [0, 0]	0 [0, 1]	0 [0, 1]	0 [0, 0]	**4** [4, 5]
scene-play-v0	Task 1	86 [78, 93]	**100** [100, 100]	55 [40, 65]	99 [98, 100]	**100** [100, 100]	**100** [100, 100]	**100** [100, 100]	**100** [100, 100]
	Task 2	54 [49, 58]	98 [97, 99]	59 [52, 71]	92 [88, 98]	98 [97, 100]	96 [93, 98]	98 [95, 100]	**99** [98, 100]
	Task 3	64 [57, 72]	97 [94, 99]	22 [20, 23]	80 [67, 88]	98 [97, 100]	**100** [100, 100]	**100** [100, 100]	82 [78, 88]
	Task 4	33 [23, 43]	65 [60, 70]	47 [38, 55]	76 [68, 80]	85 [75, 92]	84 [82, 87]	**86** [78, 93]	75 [65, 82]
	Task 5	3 [1, 6]	52 [46, 56]	22 [13, 30]	70 [59, 78]	78 [72, 85]	64 [52, 77]	**85** [81, 91]	69 [60, 80]
	Overall	48 [45, 51]	82 [81, 83]	41 [35, 46]	84 [78, 87]	92 [90, 93]	89 [87, 91]	**94** [91, 95]	85 [82, 90]
puzzle-3x3-play-v0	Task 1	**100** [100, 100]	94 [82, 100]	63 [57, 68]	94 [90, 98]	**100** [100, 100]	**100** [100, 100]	99 [98, 100]	**100** [100, 100]
	Task 2	**100** [100, 100]	91 [75, 100]	58 [42, 73]	72 [59, 81]	74 [55, 93]	85 [79, 91]	78 [70, 87]	81 [73, 88]
	Task 3	**100** [100, 100]	74 [25, 100]	32 [30, 35]	58 [52, 62]	67 [58, 76]	78 [70, 87]	70 [68, 72]	66 [59, 73]
	Task 4	**99** [98, 100]	72 [24, 98]	36 [32, 41]	59 [57, 64]	69 [58, 80]	76 [67, 86]	52 [40, 62]	48 [38, 58]
	Task 5	**100** [100, 100]	72 [25, 100]	49 [43, 55]	65 [55, 75]	78 [68, 90]	84 [78, 90]	79 [75, 84]	64 [62, 67]
	Overall	**100** [100, 100]	81 [46, 99]	48 [46, 49]	70 [65, 74]	78 [71, 87]	85 [81, 90]	76 [72, 80]	72 [69, 75]
puzzle-4x4-play-v0	Task 1	85 [77, 92]	94 [90, 98]	0 [0, 0]	70 [63, 81]	85 [82, 88]	76 [68, 83]	81 [77, 85]	**96** [92, 100]
	Task 2	47 [33, 60]	0 [0, 0]	1 [0, 2]	63 [48, 78]	72 [66, 78]	67 [62, 70]	68 [61, 75]	**92** [80, 100]
	Task 3	78 [64, 90]	32 [20, 43]	0 [0, 0]	65 [56, 78]	74 [63, 83]	78 [75, 80]	60 [51, 76]	**92** [80, 100]
	Task 4	72 [64, 78]	0 [0, 0]	1 [0, 2]	52 [50, 53]	71 [62, 80]	68 [57, 78]	71 [68, 75]	**89** [75, 100]
	Task 5	72 [56, 88]	0 [0, 0]	0 [0, 0]	72 [67, 78]	68 [53, 82]	73 [68, 82]	67 [62, 73]	**92** [78, 100]
	Overall	71 [61, 78]	25 [23, 27]	0 [0, 1]	64 [61, 68]	74 [67, 79]	72 [69, 75]	69 [65, 75]	**92** [81, 100]
puzzle-4x5-play-v0	Task 1	68 [53, 76]	**81** [63, 97]	18 [13, 23]	43 [39, 47]	46 [38, 55]	45 [40, 52]	46 [36, 60]	36 [29, 42]
	Task 2	0 [0, 0]	0 [0, 0]	0 [0, 0]	**36** [27, 46]	17 [10, 23]	27 [18, 35]	19 [7, 32]	20 [15, 25]
	Task 3	0 [0, 0]	0 [0, 0]	0 [0, 0]	7 [2, 12]	20 [12, 27]	**21** [18, 23]	6 [3, 8]	3 [0, 10]
	Task 4	0 [0, 0]	0 [0, 0]	1 [0, 2]	**30** [23, 37]	1 [0, 2]	0 [0, 0]	1 [0, 2]	17 [5, 28]
	Task 5	0 [0, 0]	0 [0, 0]	0 [0, 0]	**2** [0, 3]	1 [0, 2]	0 [0, 0]	0 [0, 0]	1 [0, 2]
	Overall	14 [11, 15]	16 [13, 19]	4 [3, 5]	**24** [22, 25]	17 [15, 20]	18 [18, 19]	14 [11, 18]	15 [11, 20]
puzzle-4x6-play-v0	Task 1	32 [20, 45]	**80** [68, 92]	20 [18, 22]	39 [33, 43]	39 [30, 49]	33 [27, 38]	38 [33, 43]	45 [41, 51]
	Task 2	10 [3, 17]	0 [0, 0]	5 [2, 8]	34 [31, 38]	21 [15, 28]	30 [27, 34]	14 [5, 24]	**37** [28, 50]
	Task 3	0 [0, 0]	0 [0, 0]	0 [0, 0]	8 [5, 10]	7 [3, 11]	**14** [7, 22]	4 [2, 7]	8 [3, 13]
	Task 4	0 [0, 0]	0 [0, 0]	0 [0, 0]	**2** [0, 5]	1 [0, 2]	1 [0, 2]	0 [0, 0]	1 [0, 2]
	Task 5	0 [0, 0]	0 [0, 0]	0 [0, 0]	0 [0, 0]	0 [0, 0]	0 [0, 0]	0 [0, 0]	0 [0, 0]
	Overall	8 [5, 10]	16 [14, 18]	5 [4, 6]	17 [15, 18]	14 [10, 16]	16 [14, 17]	11 [8, 15]	**18** [15, 22]
Average over 55 Tasks		29 [27, 30]	32 [29, 34]	20 [20, 20]	45 [44, 46]	46 [44, 49]	49 [48, 49]	47 [46, 48]	**54** [51, 56]

Table 7: **Full results on selected OGBench manipulation environments using default datasets.** Success rate (%) on each evaluation task after 1M offline training steps. Mean and 95% bootstrap confidence interval over 4 seeds. Last row averages each seed over all 55 tasks, locomotion and manipulation (Table 6).

G.5 Analysis and Ablations

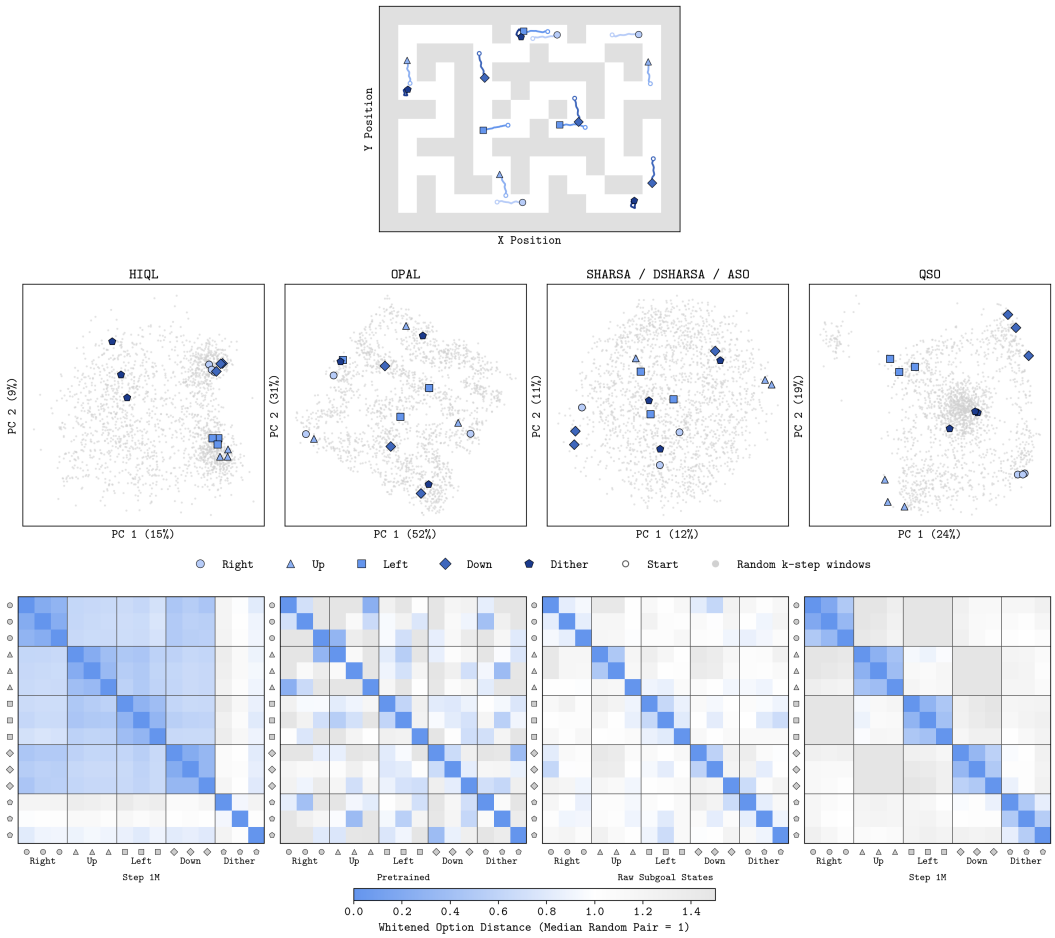

Figure 4: **PCA** (**top**) of each algorithm's action abstractions for `antmaze-giant-stitch-v0` fitted on random trajectory sequences, showing different groups of three k-step trajectory sequences with semantically similar behaviour (randomly selected from the validation dataset). **L2 embedding distances** (**bottom**) for those groups in units of median distance between two random trajectory sequences. QSO is the only algorithm that consistently groups functionally equivalent behaviours while separating the behaviour groups.

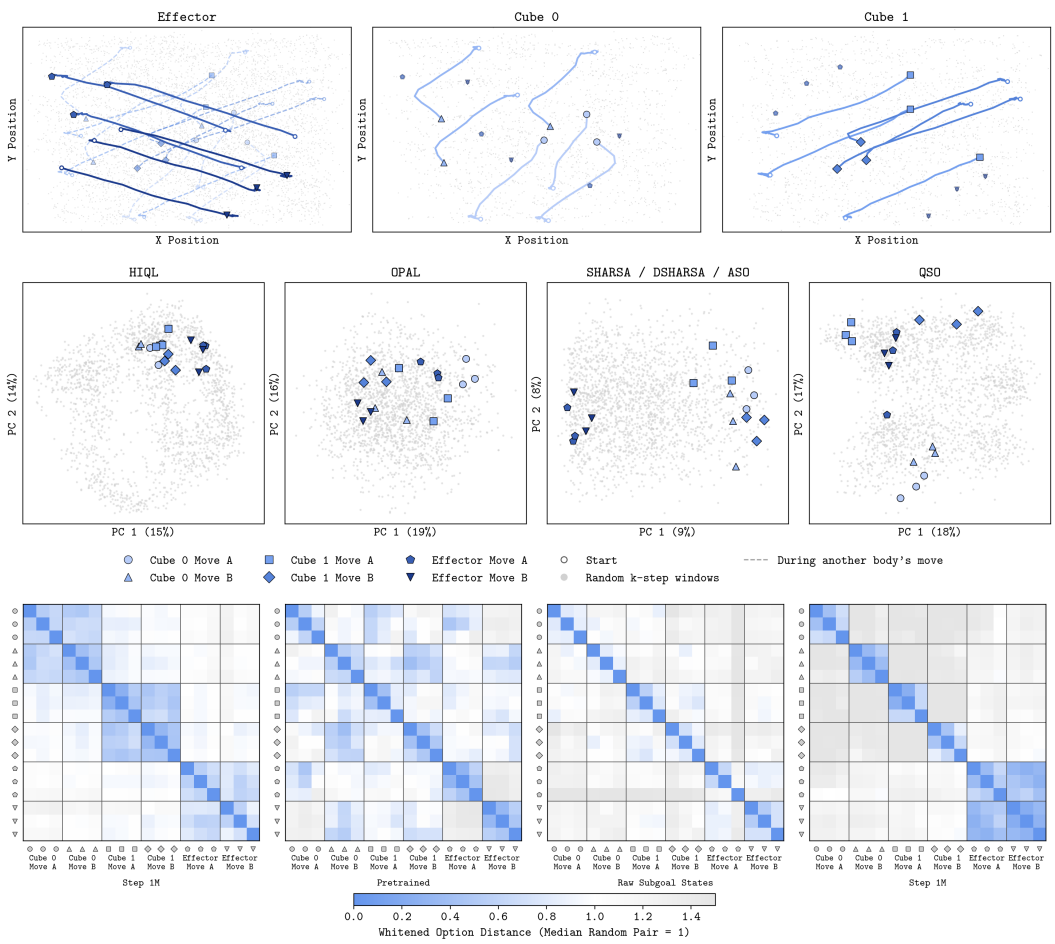

Figure 5: **PCA** (**top**) of each algorithm's action abstractions for `cube-double-play-v0` fitted on random trajectory sequences, showing different groups of three k-step trajectory sequences with semantically similar behaviour (randomly selected from the validation dataset). **L2 embedding distances** (**bottom**) for those groups in units of median distance between two random trajectory sequences. QSO is the only algorithm that consistently groups functionally equivalent behaviours while separating the behaviour groups.

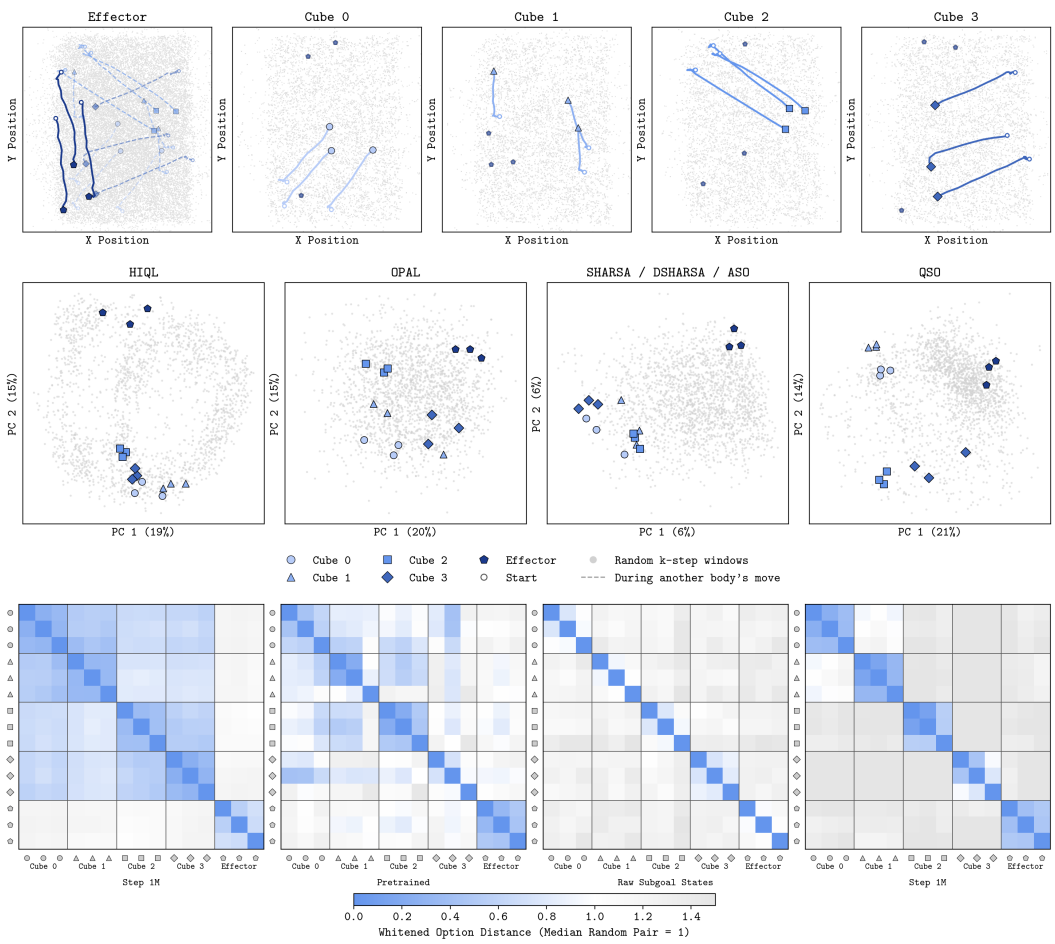

Figure 6: **PCA** (**top**) of each algorithm's action abstractions for `cube-quadruple-play-v0` fitted on random trajectory sequences, showing different groups of three k-step trajectory sequences with semantically similar behaviour (randomly selected from the validation dataset). **L2 embedding distances** (**bottom**) for those groups in units of median distance between two random trajectory sequences. QSO is the only algorithm that consistently groups functionally equivalent behaviours while separating the behaviour groups.

	Reward Labelling		High-level Policy			High-level Value			Option Encoder			Low-level Policy		
Agent	High	Low	Fast	Slow	Interaction/ Noise	Fast	Slow	Interaction	Fast	Slow	Interaction	Fast	Slow	Interaction/ Noise
antmaze-giant-stitch-v0														
QSO	E	S	35 [31, 39]	14 [13, 16]	50 [47, 54]	6 [4, 7]	88 [86, 90]	7 [6, 7]	45 [42, 47]	45 [42, 47]	11 [10, 11]	59 [57, 60]	4 [4, 4]	37 [36, 38]
	E	E	28 [26, 30]	18 [17, 19]	54 [52, 55]	6 [4, 7]	88 [86, 90]	6 [6, 7]	23 [21, 25]	72 [70, 74]	5 [5, 6]	55 [54, 57]	5 [4, 5]	40 [39, 41]
	S	S	34 [30, 38]	14 [13, 15]	52 [49, 56]	8 [6, 10]	84 [81, 87]	8 [7, 9]	46 [43, 48]	45 [43, 48]	9 [9, 9]	54 [53, 56]	5 [4, 5]	41 [40, 42]
	S	E	29 [26, 32]	19 [18, 20]	52 [51, 54]	8 [6, 11]	84 [81, 87]	8 [7, 8]	28 [25, 30]	68 [65, 70]	5 [5, 5]	56 [54, 57]	4 [4, 5]	40 [39, 41]
ASO	E	S	21 [18, 24]	13 [12, 14]	67 [64, 69]	5 [4, 6]	88 [86, 90]	7 [7, 8]	--	--	--	53 [51, 54]	5 [5, 5]	42 [41, 44]
	E	E	21 [18, 24]	13 [12, 14]	67 [64, 69]	5 [4, 6]	88 [86, 90]	7 [7, 8]	--	--	--	53 [51, 54]	5 [5, 5]	42 [41, 44]
	S	S	19 [16, 22]	13 [12, 14]	68 [66, 70]	8 [6, 10]	83 [80, 85]	9 [8, 10]	--	--	--	54 [53, 55]	5 [4, 5]	41 [40, 43]
	S	E	19 [16, 22]	13 [12, 14]	68 [66, 70]	8 [6, 10]	83 [80, 85]	9 [8, 10]	--	--	--	54 [53, 55]	5 [4, 5]	41 [40, 43]
cube-double-play-v0														
QSO	E	S	73 [71, 74]	5 [5, 5]	22 [21, 24]	3 [2, 4]	74 [70, 77]	23 [20, 26]	79 [77, 80]	15 [14, 16]	6 [6, 7]	77 [75, 78]	4 [4, 5]	19 [18, 20]
	E	E	25 [23, 27]	23 [21, 24]	52 [51, 53]	3 [3, 4]	75 [72, 78]	21 [19, 24]	8 [8, 9]	84 [83, 85]	8 [7, 8]	71 [69, 72]	8 [7, 8]	22 [21, 23]
	S	S	71 [70, 72]	7 [6, 7]	22 [21, 23]	4 [3, 5]	70 [67, 72]	26 [24, 28]	76 [75, 78]	15 [14, 16]	9 [8, 9]	81 [80, 83]	5 [4, 5]	14 [13, 15]
	S	E	18 [17, 20]	39 [37, 41]	42 [41, 44]	4 [3, 5]	73 [71, 76]	23 [21, 25]	8 [7, 9]	83 [82, 84]	9 [9, 9]	75 [74, 77]	5 [4, 5]	20 [19, 21]
ASO	E	S	26 [25, 27]	52 [51, 53]	22 [21, 23]	3 [3, 4]	75 [71, 78]	22 [19, 25]	--	--	--	88 [87, 89]	2 [2, 2]	10 [9, 11]
	E	E	26 [25, 26]	52 [51, 53]	23 [22, 24]	3 [3, 4]	74 [70, 78]	22 [19, 26]	--	--	--	88 [87, 89]	2 [2, 2]	10 [9, 11]
	S	S	26 [25, 27]	52 [51, 53]	22 [21, 23]	4 [3, 5]	66 [63, 69]	30 [28, 32]	--	--	--	88 [87, 89]	3 [2, 3]	9 [8, 10]
	S	E	26 [25, 27]	52 [51, 53]	22 [21, 23]	4 [3, 5]	66 [63, 69]	30 [28, 32]	--	--	--	88 [87, 89]	3 [2, 3]	9 [8, 10]

Table 8: **QSO and ASO high and low-level decision process state abstractions.** Percentage share of the variance of each output explained by slow-moving dimensions (state and goal XY coordinates for `antmaze-giant-stitch-v0`; state and goal cube positions and orientations for `cube-double-play-v0`) and by the fast-moving dimensions (respectively state and goal XY height, orientation, joints and velocities; and state and goal effector, gripper, joints and velocities). The reward labelling columns indicate whether the high and low-level decision processes are trained using environment (E) or strict (S) reward labelling. Inputs are built on grids that combine the slow-moving dimensions of 16 random validation state-goal (or state-target) pairs with the fast-moving dimensions of 16 others, each under 8 flow-noise samples. For QSO, the state-target pairs are encoded before passing them to the low-level policy; the option encoder columns give the variance of these encoded options (ASO has no encoder). For ASO's high-level policy and both ASO's and QSO's low-level policies, each dimension of the output is divided by the larger of its standard deviations over the dataset and over the sampled absolute subgoal options or actions. The variance is attributed exactly using two-way ANOVA, so that the attributions sum to 100. Mean and 95% bootstrap confidence interval (2000 resamples of the 64 grids) for one seed after 1M training steps.

Further discussion on the coherence of state abstractions. The high-level policy remains dependent on fast-moving dimensions (with 35% of the output variance being explained by fast-moving dimensions in antmaze-giant and 73% in cube-double). We attribute this to our method of Q-shaping: because the option is shaped to retain information about the required immediate action, if the target's fast-moving dimensions form part of the representation that the option encodes, the high-level policy inherits this dependence. The following ablation confirms this hypothesis. We ablate using the environment success criterion (which only focusses on slow-moving dimensions) rather than the stricter success criterion to also label the reward in the dataset for the low-level decision process. Doing so makes the option encoder and hence also high-level policy substantially less dependent on fast-moving dimensions: percentage shares for the option encoder drop from 45% to 23% and from 79% to 8% respectively, while for the high-level policy, they drop from 35% to 28% and 73% to 25%. Because we see no loss in downstream performance (Figure 13) and although we use the stricter success criterion for labelling reward in the low-level decision process across all experiments, we hypothesise that the environment success criterion might be preferable at both levels.



Figure 7: **QSO's sampled options gradually shift across the option space during deployment.** Distance between two consecutively sampled options (every 10 steps), whitened against the distance between two randomly sampled options. **Dotted blue** shows the distance between two random trajectory sequence embeddings whitened against the distance between two randomly sampled options; this being approximately 1 confirms that the small consecutive distances reflect genuinely slow variation rather than a collapsed policy. Mean and 95% confidence interval over 4 seeds, cut off where the third quartile of successful episodes has ended.

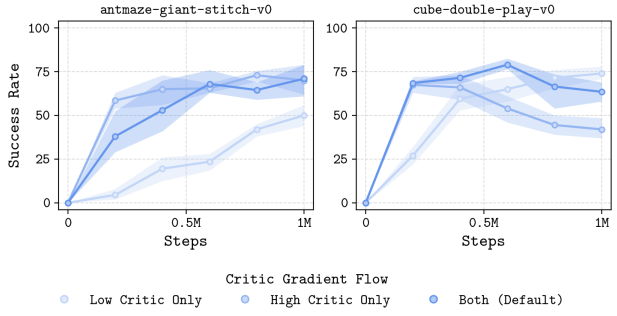

Figure 8: High and low-level critic **Q-function shaping ablation**.

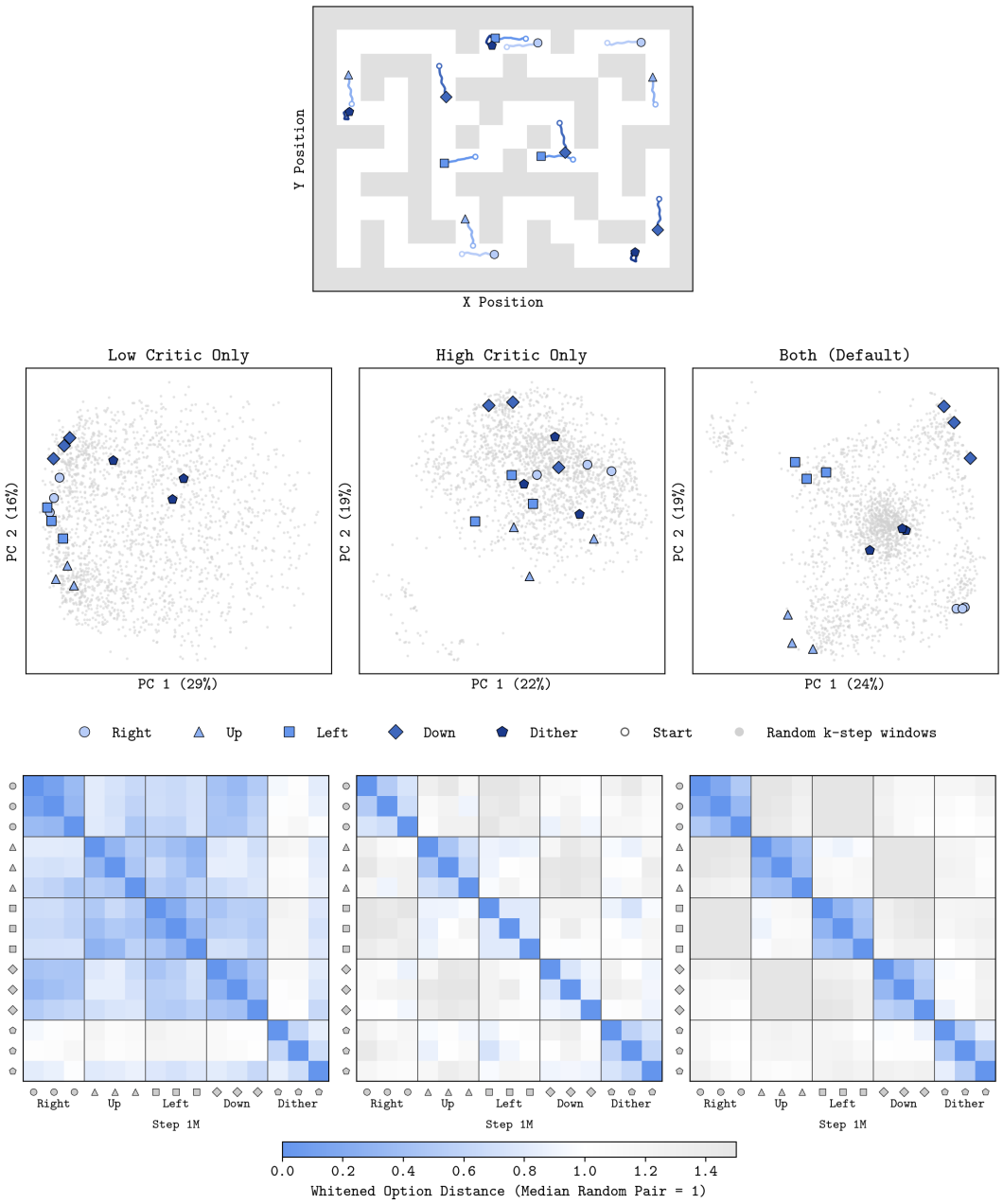

Figure 9: **PCA** (**top**) of QSO's action abstractions for gradient flow ablations for `antmaze-giant-stitch-v0` fitted on random trajectory sequences, showing different groups of three k-step trajectory sequences with semantically similar behaviour (randomly selected from the validation dataset). **L2 embedding distances** (**bottom**) for those groups in units of median distance between two random trajectory sequences. Both high and low-level gradient flows are required to group functionally equivalent behaviours while separating the behaviour groups.

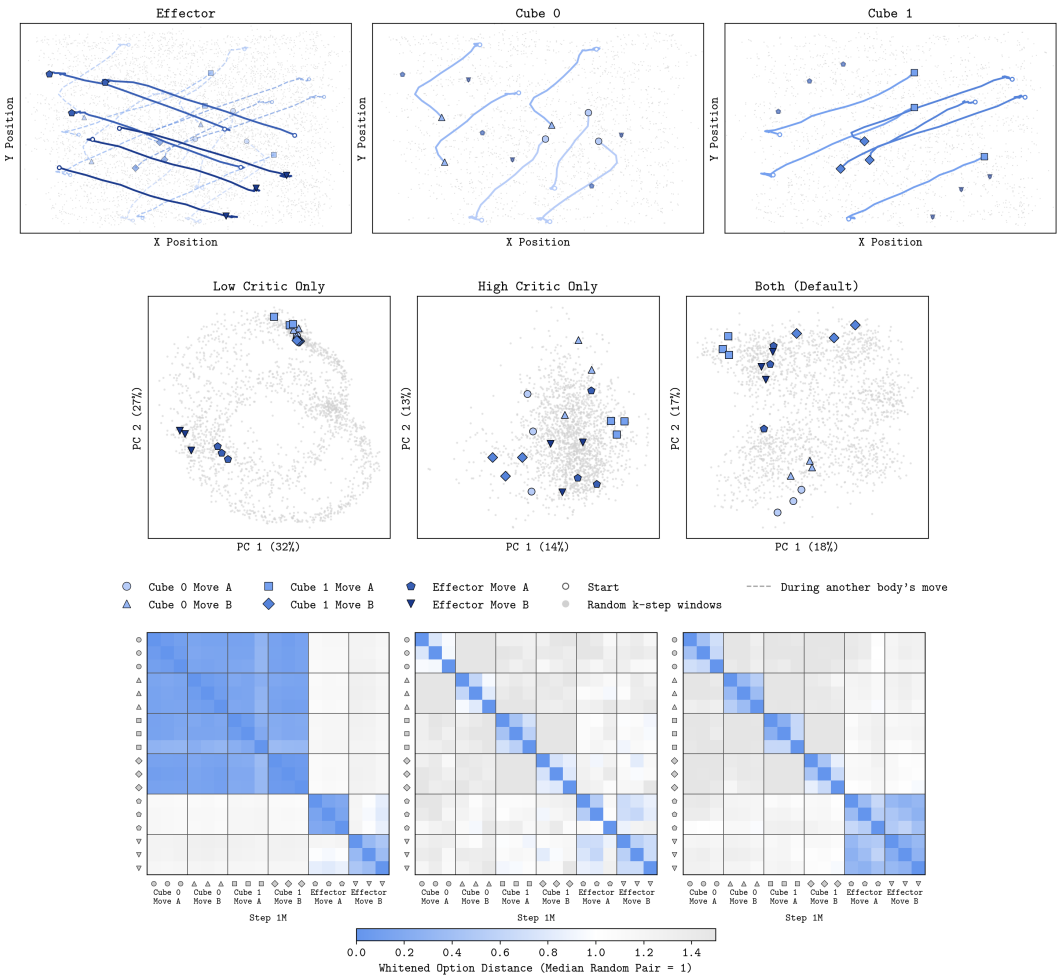

Figure 10: **PCA** (**top**) of QSO's action abstractions for gradient flow ablations for `cube-double-play-v0` fitted on random trajectory sequences, showing different groups of three k-step trajectory sequences with semantically similar behaviour (randomly selected from the validation dataset). **L2 embedding distances** (**bottom**) for those groups in units of median distance between two random trajectory sequences. Both high and low-level gradient flows are required to group functionally equivalent behaviours while separating the behaviour groups.

How sensitive is QSO to option representation input? Although our reasoning in Section 4 motivates representing options as relative achievements, we test whether this design choice improves downstream performance. Figure 11 compares performance with absolute achievement $\phi_\omega(s_{t+k})$ with relative achievement encodings $\phi_\omega(s_t, s_{t+k})$. Relative encoding substantially improves performance in `antmaze-giant-stitch-v0`, but has little effect in `cube-double-play-v0`. We hypothesise this to be because relative encoding provides the largest benefit when many absolute state-goal pairs (s, g_s) can be collapsed into the same context-dependent achievement, which is the case for the locomotion task, but not necessarily for the manipulation one, where the above state-abstraction analysis suggests that the option retains much more fine-grained information for immediate control.

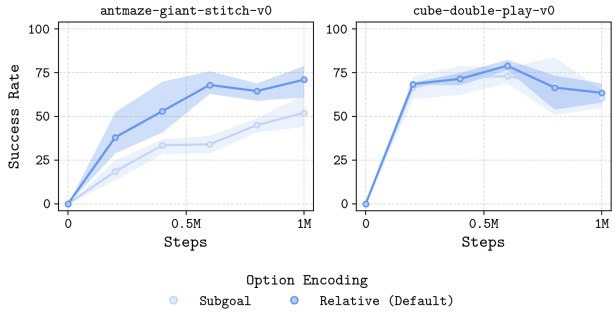

Figure 11: $\phi_\omega(s_{t+k})$ rather than $\phi_\omega(s_t, s_{t+k})$ **option encoding ablation**.

How sensitive is QSO to goal sampling? Both QSO and DSHARSA are relatively robust to the proportion of random high-level goals (Figure 12). In both `antmaze-giant-stitch-v0` and `cube-double-play-v0`, all three sampling ratios converge to similar performance in both algorithms.

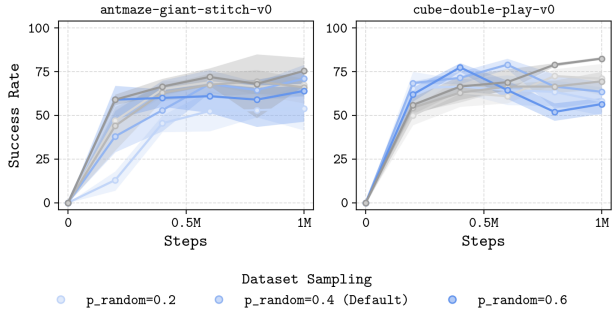

Figure 12: Random **goal sampling ablation** for the high-level decision process for QSO (**blue**) and ASO (**grey**).

How sensitive is QSO to reward labelling? Environment reward labelling for the high-level decision process substantially improves performance for both QSO and ASO in `cube-double-play-v0`, while on `antmaze-giant-stitch-v0` the effect is small (Figure 13). For ASO, this gain comes from the high-level critic: ranking samples from the environment-labelled high-level policy using the strict-labelled high-level critic recovers the full performance (of 70% rather than 20%), while the reverse swap results in a corresponding drop in performance. We hypothesise that the environment reward labelling enables the high-level decision process to form better state abstractions: consistently, the percentage share of the output variance for slow-moving dimensions in the high-level state-value function increases by almost 10% (66% to 75%) for `cube-double-play-v0` (Table 8). This is not the case for `antmaze-giant-stitch-v0`, where even strict labelling yields a value dominated by the slow-moving dimensions. Note that we analyse percentage shares of output variance for the state-value function rather than the critic, which would add a confounding factor of how to sample the subgoal/option for the crossed state-goal pairs.

For QSO, reward labelling also shapes the option space itself, as the option encoder is trained through the critics. Downstream performance is most sensitive to using high-level reward labelling, which, like for ASO, we attribute to the high-level critic. Interestingly, however, low-level environment labelling makes the encoder markedly less sensitive to fast-moving dimensions, dropping from 79% to 8% for `cube-double-play-v0` and 45% to 23% for `antmaze-giant-stitch-v0`; the high-level policy becomes correspondingly more dependent on slow-moving dimensions, respectively increasing from 5% to 23% and 14% to 18%. Hence, while the option space remains similarly clean under both variants (Figures 16 and 17), using low-level environment reward labelling could be considered better, as it retains less information about fast-moving state dimensions without sacrificing downstream performance.

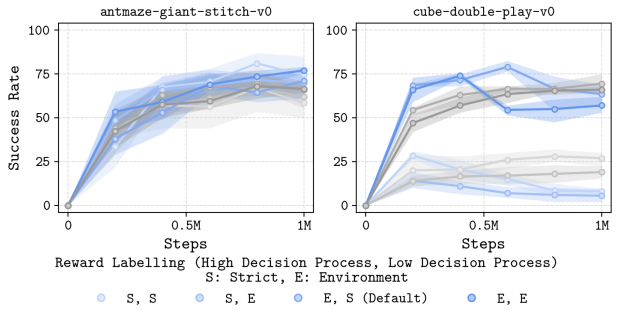

Figure 13: Strict versus environment dataset **reward labelling ablation** for QSO (**blue**) and ASO (**grey**).

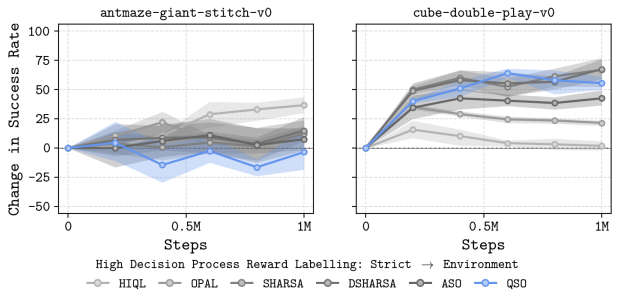

Figure 14: Change in performance when using the **environment success criterion rather than the stricter success criterion used by prior work** for reward labelling in the high-level decision process. Performance either does not change significantly, or improves for all algorithms.

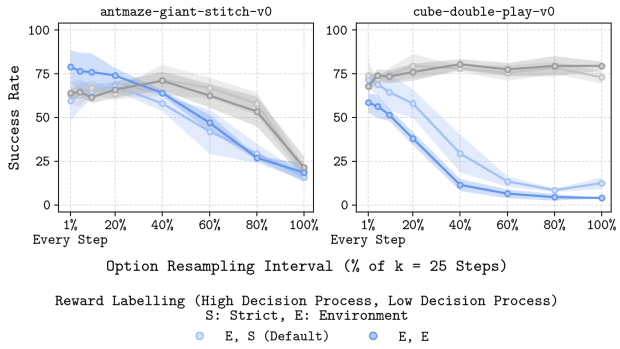

Figure 15: Performance with different **option resampling frequencies** for **low-level dataset reward labelling ablations** for QSO (**blue**) and ASO (**grey**).

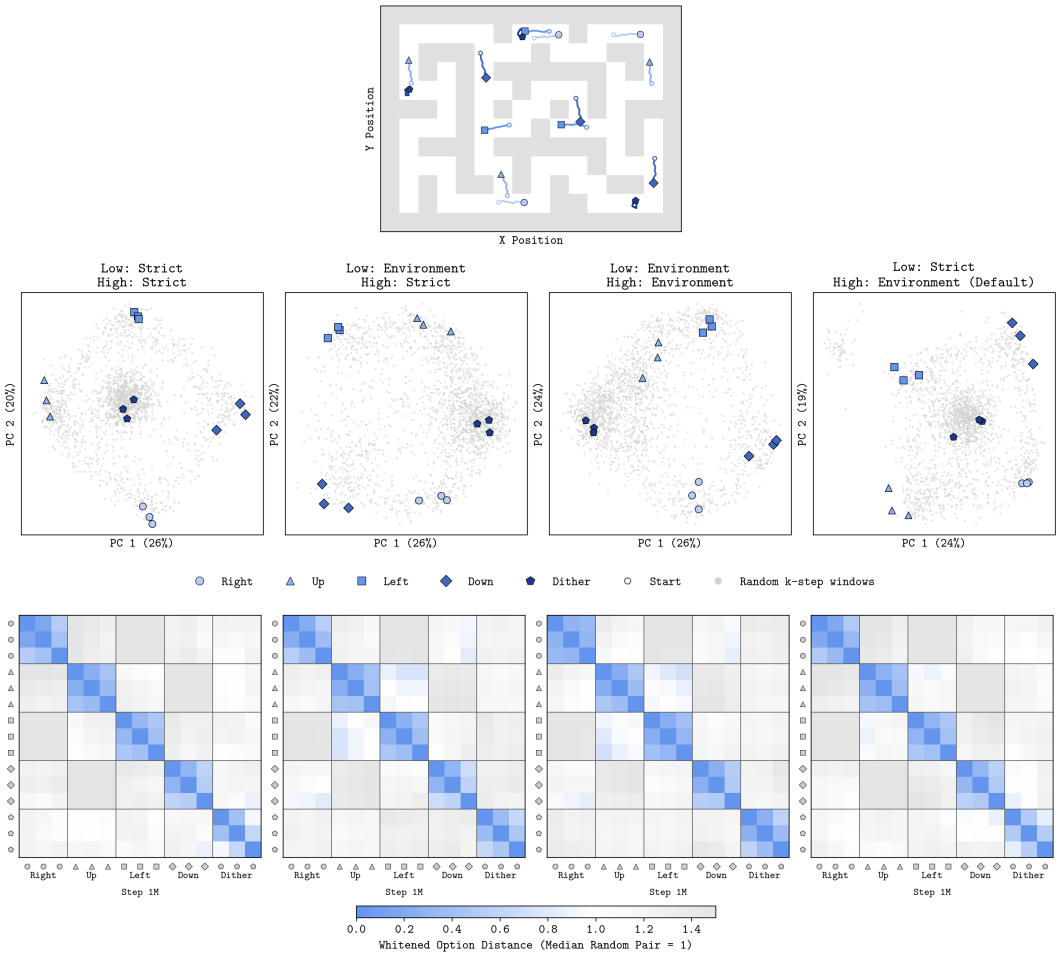

Figure 16: **PCA** (**top**) of QSO's action abstractions for reward labelling ablations for `antmaze-giant-stitch-v0` fitted on random trajectory sequences, showing different groups of three k-step trajectory sequences with semantically similar behaviour (randomly selected from the validation dataset). **L2 embedding distances** (**bottom**) for those groups in units of median distance between two random trajectory sequences. All reward labelling seems to encourage grouping functionally equivalent behaviours while separating the behaviour groups.

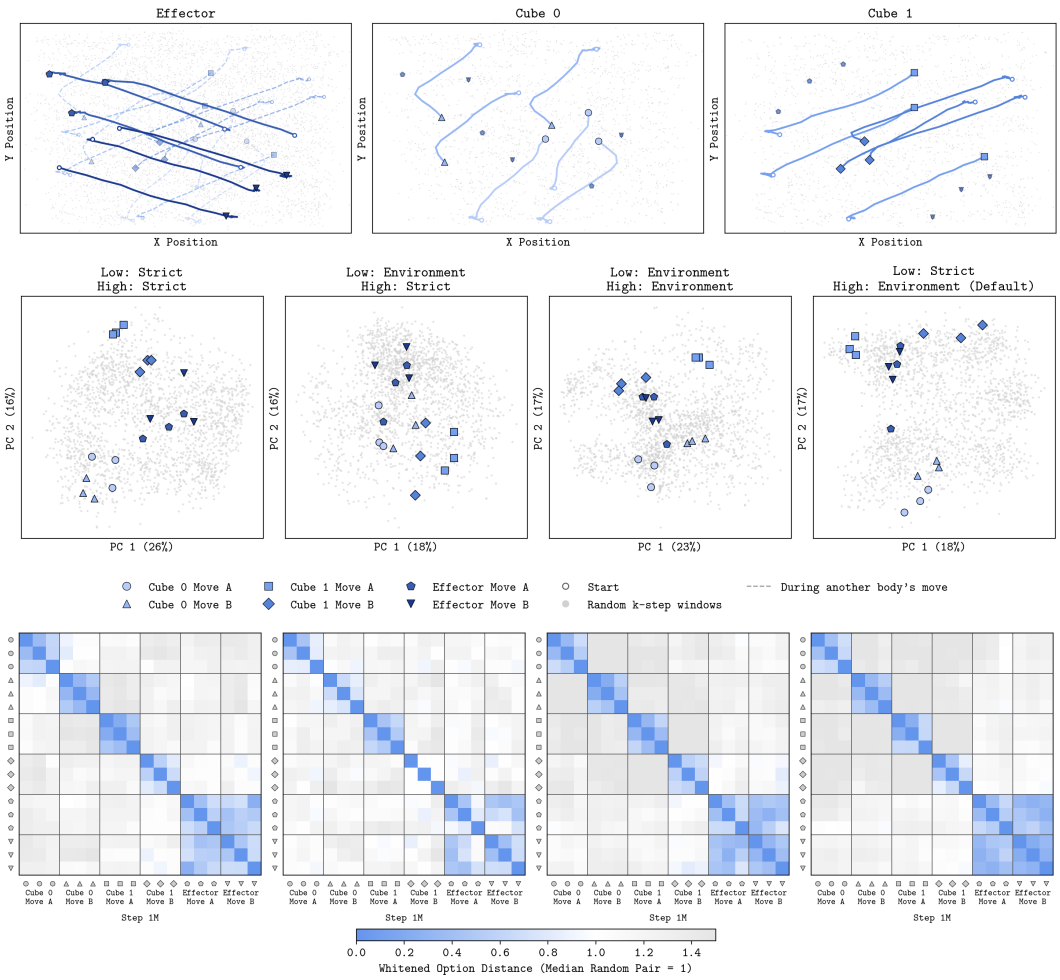

Figure 17: **PCA** (**top**) of QSO's action abstractions for reward labelling ablations for `cube-double-play-v0` fitted on random trajectory sequences, showing different groups of three k-step trajectory sequences with semantically similar behaviour (randomly selected from the validation dataset). **L2 embedding distances** (**bottom**) for those groups in units of median distance between two random trajectory sequences. All reward labelling seems to encourage grouping functionally equivalent behaviours while separating the behaviour groups.

How sensitive is QSO to policy extraction? Both advantage weighting and RS lead to more consistent performance for ASO and QSO (Figure 18). As motivated, advantage weighting increases the high-level policy's dependence on the goal: compared with BC, the goal's share of output variance increases from 1% to 14% for QSO and 1% to 5% for ASO in `antmaze-giant-stitch-v0`, and, respectively, from 0% to 1% and 1% to 3% in `cube-double-play-v0`. Note that RS on top of this further increases goal-dependency (Table 9).

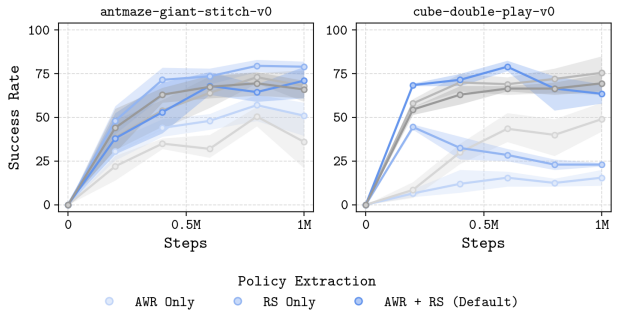

Figure 18: High-level **policy extraction ablation** for QSO (**blue**) and ASO (**grey**).

		Flow Policy			Deployed Policy (best of 32)		
Agent	Policy Extraction	State	Goal	Noise	State	Goal	Noise
antmaze-giant-stitch-v0							
QSO	BC	72 [69, 74]	1 [1, 2]	27 [25, 30]	60 [57, 62]	24 [23, 26]	16 [15, 17]
	AWR	64 [61, 67]	**14** [13, 15]	22 [21, 24]	54 [52, 57]	**34** [32, 36]	12 [11, 12]
SHARSA	BC	38 [35, 41]	1 [1, 2]	61 [58, 64]	34 [31, 36]	15 [15, 16]	51 [49, 53]
ASO	AWR	35 [32, 38]	**5** [5, 5]	60 [57, 62]	33 [31, 36]	**19** [18, 20]	48 [46, 50]
cube-double-play-v0							
QSO	BC	98 [98, 98]	0 [0, 0]	2 [2, 2]	97 [96, 97]	2 [2, 2]	2 [1, 2]
	AWR	93 [92, 93]	**1** [1, 1]	6 [6, 7]	88 [88, 89]	**6** [6, 7]	5 [5, 6]
SHARSA	BC	85 [85, 86]	1 [1, 1]	14 [13, 15]	80 [79, 81]	9 [8, 9]	11 [10, 13]
ASO	AWR	84 [83, 85]	**3** [3, 4]	13 [12, 13]	79 [78, 79]	**12** [11, 12]	10 [9, 10]

Table 9: **QSO, SHARSA and ASO high-level goal sensitivity with AWR and BC policy extraction.** Percentage share of the variance of the high-level policy's options explained by the state and goal. Inputs are built on grids that combine 16 random validation states with 16 random validation goals, each under 8 flow-noise samples. For SHARSA and ASO, each dimension of the output is divided by the larger of its standard deviations over the dataset and over the sampled absolute subgoal options. The variance is attributed exactly using two-way ANOVA, so that the attributions sum to 100 ("Goal" includes the state-goal interaction). The flow policy returns a single flow sample; the deployed policy returns the best of 32 flow samples under the high-level critic. Mean and 95% bootstrap confidence interval (2000 resamples of the 64 grids) for one seed after 1M training steps.

How sensitive is QSO to option dimension? QSO is largely insensitive to option dimension from 5 to 20, which all lead to similar downstream performance (Figure 19). We use a fixed default of 10.

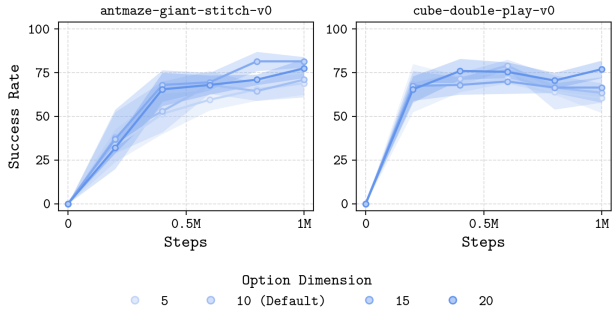

Figure 19: d **option-space dimensionality ablation**.

How sensitive is QSO to temporal horizon? Like ASO and prior work (Park et al., 2026; Ajay et al., 2020; Park et al., 2023; Li et al., 2026), QSO is sensitive to temporal abstraction horizon (Figure 20). For example, option horizons of 100 steps substantially degrades performance for both QSO and ASO in `antmaze-giant-stitch-v0`; this is unsurprising, however, considering that the maximum number of steps in the `antmaze-giant-stitch-v0` dataset is 200.

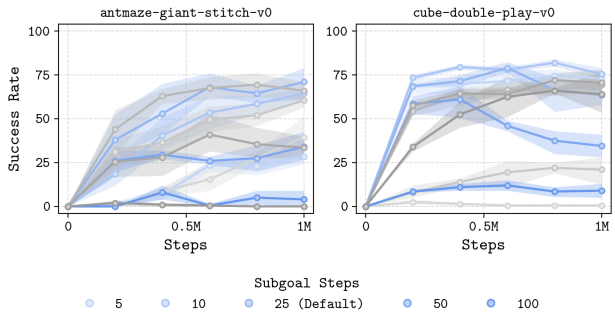

Figure 20: k **temporal horizon ablation** for QSO (**blue**) and ASO (**grey**).