DISCOVERY OF STICKY AND RESPONSIBLE OPTIONS FOR FROZEN-OPTION TRANSFER

Anonymous authors
Paper under double-blind review

ABSTRACT

A fundamental problem in hierarchical reinforcement learning is finding a small number of reusable and easily composable behaviors that transfer with little or no fine-tuning to solve novel tasks. Option and skill discovery frameworks are two prominent approaches that address this problem. However, option discovery typically focuses on learning specialized behaviors for a training task, leading to a low diversity of behaviors with limited transferability, and unsupervised skill discovery learns diverse behaviors that can be misaligned with downstream tasks. To address these issues, we introduce Task-Agnostic Sticky and Responsible Options (TASRO), an option-learning framework that combines (1) Markovian option switching at every time step, (2) an objective promoting persistent, diverse, and specialized options, and (3) information asymmetry, where the high-level controller observes both task and task-agnostic information while the low-level options observe only task-agnostic information. We evaluate under a strict frozen-option transfer protocol: pretrain options on a goal-reaching training task, freeze them, and relearn only the controller on downstream tasks. Across our navigation, locomotion, and manipulation benchmarks, TASRO improves frozen-option transfer over the tested skill-discovery and task-exposed option baselines.

1 INTRODUCTION

Hierarchical reinforcement learning (HRL) seeks a small set of reusable, composable behaviors that can be transferred to new tasks with *little or no fine-tuning* (Sutton et al., 1999; Dayan & Hinton, 1992; Dietterich, 2000; Botvinick et al., 2009; Mann & Mannor, 2014; Konidaris & Barto, 2007; Eysenbach et al., 2019; Frans et al., 2018). For example, an agent that can crawl, walk, run, and jump in arbitrary directions, for varying durations, and flexibly switch among these behaviors is better equipped to navigate and survive in a complex forest. These behaviors are most useful if they are pretrained and do not need to be relearned or fine-tuned in the forest. However, discovering a small library of behaviors that is genuinely useful for transfer remains difficult: the behaviors should be temporally extended, composable, reusable across related tasks, and diverse enough to support new objectives (Sutton et al., 1999; Garcia et al., 2020; Igl et al., 2020; Eysenbach et al., 2019).

Options are a natural starting point for learning reusable behaviors in HRL because they represent temporally extended closed-loop sub-policies selected by a higher-level controller (Sutton et al., 1999). Subsequent work learned such options end-to-end from task reward, notably the option-critic architecture (Bacon et al., 2017), its continuous-action extension (Klissarov et al., 2017), and related hierarchical policy-search approaches (Daniel et al., 2012). However, when both options and the high-level controller are given the full task information (Bacon et al., 2017; Klissarov et al., 2017), options tend to overfit to the training task, so that high performance in the training task does not imply that the learned options are reusable downstream. Fine-tuning can compensate by adapting the options downstream (Li et al., 2020), but *frozen-option transfer* keeps the library fixed and trains only a new downstream controller (Igl et al., 2020; Veeriah et al., 2021; Garcia et al., 2020). This is a valuable empirical test of whether the learned options are genuinely reusable, and a practically relevant one, since deployed systems often cannot afford extensive retraining of low-level behavior. Two major directions for frozen-option transfer are pretraining options via motion imitation or dedicated training tasks and freezing them for transfer (Peng et al., 2019; Qureshi et al., 2020), or discovering reusable options from multi-task training distributions (Igl et al., 2020; Veeriah et al., 2021; Garcia et al., 2020). Although extremely promising, these approaches require either

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.

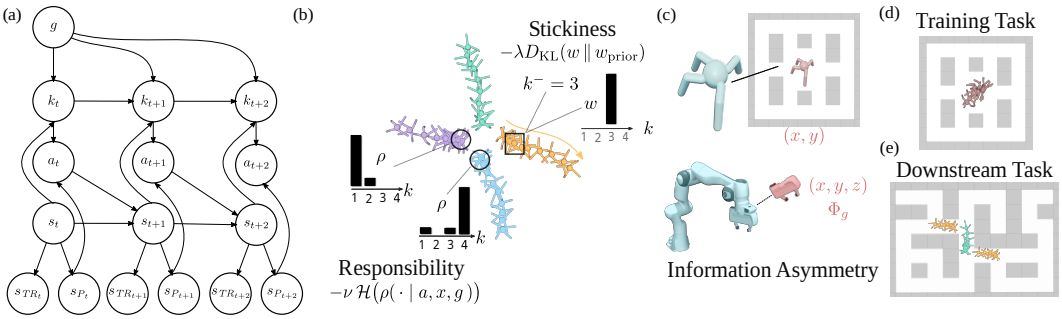


Figure 1: **Overview of TASRO. (a) Markovian option process:** The controller selects each option from the goal g, current state s_t, and previous option k_{t-1}, so persistence and switching are Markov decisions on (s_t, k_{t-1}). The state $s = (s_P, s_{TR})$ is decomposed into task-relevant variables s_{TR} and proprioceptive variables s_P. **(b) Learned option library** ($K = 4$). The sticky-prior KL penalty on the gate distribution $w(k|x, g)$, which selects option k from context x and goal g, encourages reselecting the previous option and thus temporal persistence. The responsibility-entropy penalty term on $\rho(k|x, a, g)$, the posterior probability that option k generated action a, encourages sharper specialization by making each action more confidently attributable to a single option. **(c) Information asymmetry**. Options π_k observe only proprioceptive state s_P and the previously active option k^-. The controller in addition observes the task-relevant state s_{TR} and the goal g. For example, in AntMaze, s_{TR} is the agent's current xy position and g is the target xy position; in Panda arm manipulation, s_{TR} includes the current gripper xyz position and gripper orientation Φ_g, while g is the desired box pose, defined by its target xyz position and a $6D$ orientation representation. **(d) Training task**. Options are learned on a single goal-conditioned task. **(e) Frozen-option transfer**. The option library is frozen and recomposed by a new controller trained on a downstream task.

high quality demonstrations, which may not be available, or learning across many tasks, which can largely impair sample efficiency during learning.

Skill discovery is an alternative to reward-trained options for learning reusable behaviors. This line of work learns skills from intrinsic diversity or coverage objectives (Eysenbach et al., 2019; Park et al., 2024). For instance, DIAYN distinguishes skills through visited states (Eysenbach et al., 2019), while METRA spreads them using metric structure over state space (Park et al., 2024). This can produce broad, task-agnostic behavioral coverage useful for downstream transfer through fine-tuning, planning, or frozen reuse. Because pretraining is unsupervised, however, the resulting repertoire is optimized for diversity or coverage rather than task usefulness; for an industrial robot arm, reusable workspace motions may matter more than covering the entire reachable state space. Most latent-conditioned skill-discovery methods hold one skill fixed over a rollout or fixed horizon and compose the learned skills only downstream, so their discovery phase does not optimize the transitions needed for flexible downstream composition. Some other work (Habib et al., 2025) discovers specialized behavioral components by maximizing mixture entropy while minimizing component entropy and supports frozen-component transfer; however, it does not generate temporally extended behaviors, so the learned tokens remain relatively short-lived. See the additional literature review in Appendix A.1.

In this paper we address the problem of frozen-option transfer under a single option-learning training task, and provide the controller with full option-switching awareness to increase downstream flexibility and option composability. We introduce Task-Agnostic Sticky and Responsible Options (TASRO), an HRL framework where (1) the high-level controller selects low-level options at every time step through a Markovian option-switching structure, (2) options are promoted to be persistent, diverse, and specialized (sticky and responsible options), and (3) the high-level controller has both task-agnostic and task-dependent information, while the low-level options only have task-agnostic information (information asymmetry). Our main result is to show that TASRO outperforms option-learning baselines (even with options having full task information) and unsupervised skill-discovery baselines across navigation, locomotion, and manipulation domains in frozen-option transfer. The novelty of TASRO lies in integrating these three design choices into a transition-aware option architecture for strict frozen-option transfer; we describe their roles below:

(1) *Markovian Option Switching*. Under standard call-and-return execution, the high-level controller is consulted only when the current option terminates, yielding a semi-Markov high-level process (Sutton et al., 1999). TASRO instead defines a *Markovian option process* by deciding at every time step whether to retain the current option or switch to a particular successor option, conditioned on the state and previously active option (Fig. 1a). The controller uses the full option-augmented state, while the selected option uses the same state or a masked version thereof. The controller therefore learns persistence and switching directly, without separate termination and state-only reselection components; all options remain available at every state, so no initiation sets are needed. Moreover, per-step option switching provides the agent with experience of transitions between options during training, aiding stability when composing them to solve downstream tasks with unstable agents.

(2) *Sticky and Responsible Options*. To promote option temporal extension, we use a sticky prior on the controller, which encourages reselecting the same option across consecutive steps (Fig. 1b). This temporal persistence allows the controller to plan at a longer time scale, facilitating learning. To promote option diversity and specialization, we combine mixture entropy, which promotes diverse action paths, with negative responsibility entropy, which makes actions more confidently attributable to individual options (Fig. 1b). Together, these three terms shape persistent, diverse, and specialized options (Eq. 3). Their integration with Markovian option switching and information asymmetry is designed for strict frozen-option transfer.

(3) *Information Asymmetry and Frozen-Option Transfer*. To aid frozen-option transferability, during training-task learning we adopt an asymmetric information structure across hierarchy levels (Fig. 1c): options receive only task-agnostic state information, while the controller observes task-dependent variables as well (Heess et al., 2016). In our environments, options have only proprioceptive information, such as joint angles, with no information about spatial location in navigation tasks or gripper orientation in manipulation tasks; the controller has full information about all those variables, in addition to goal or reward information. Options are trained jointly with the controller on a single goal-reaching training task (Fig. 1d); on downstream tasks, only the controller is learned, while options are frozen (Fig. 1e). Our ablations show that asymmetry is central but not sufficient: TASRO's switching and regularization also help organize a reusable library (Sec. 3.3).

2 Objective for Sticky and Responsible Options

We aim to learn task-agnostic options that generate diverse, reusable, and easy-to-compose behaviors for downstream tasks. We consider a Markov decision process $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where $\mathcal{S}$ is a continuous state space, $\mathcal{A}(s)$ is a state-dependent continuous action space, $p(s'|s,a)$ is the transition kernel, $r(s,a)$ is a fixed reward function, and $\gamma \in [0,1)$ is the discount factor. To introduce information asymmetry, we decompose the state as $s = (s_P, s_{TR})$, where s_P denotes task-agnostic proprioceptive information and s_{TR} denotes task-relevant or reward information.

To make option switching Markovian, we define the option-augmented state $x \equiv (s, k^-) = (s_P, s_{TR}, k^-)$, which corresponds to the state observed by the controller, where $k^- \in \{1, \dots, K\}$ is the option used in the previous time step, and the option state $x_P \equiv (s_P, k^-)$, which corresponds to the state observed by the options. Our framework can also handle the case where options see the full augmented state x, but here we focus on the masked case. We define the mixture policy

$$\pi_m(a|x) = \sum_{k=1}^{K} w(k|x)\pi_k(a|x_P) \tag{1}$$

as a convex combination of K option policies $\pi_k(a|x_P)$ with high-level controller $w(k|x) \geq 0$ and $\sum_k w(k|x) = 1$. At each time step, the controller samples an option $k \sim w(\cdot|x)$, the selected option samples an action $a \sim \pi_k(\cdot|x_P)$, the environment transitions according to $p(s'|s,a)$, and the next option-augmented state is $x' = (s', k)$.

Our goal is to optimize both the options $\{\pi_k\}$ and the controller w so as to maximize the value function

$$V(x) = \mathbb{E}_{\substack{k_t \sim w(\cdot|x_t),\ a_t \sim \pi_{k_t}(\cdot|x_{P,t}),\\ s_{t+1} \sim p(\cdot|s_t, a_t)}} \left[\sum_{t=0}^{\infty} \gamma^t \left(r(s_t, a_t) + R(x_t; \pi, w) \right) \,\middle|\, x_0 = x \right], \tag{2}$$

with the three-factor regularizer

$$R(x;\pi,w) \equiv \underbrace{\alpha\, \mathcal{H}(\pi_m(\cdot|x))}_{\text{diversity}} - \underbrace{\nu\, \mathbb{E}_{a\sim\pi_m(\cdot|x)}[\mathcal{H}(\rho(\cdot|x,a))]}_{\text{specialization}} - \underbrace{\lambda\, D_{\mathrm{KL}}(w(\cdot|x)\|w_{\mathrm{prior}}(\cdot|x))}_{\text{persistence (stickiness)}} . \quad (3)$$

Here $\mathcal{H}(\pi(\cdot|y)) = -\int da\, \pi(a|y) \log \pi(a|y)$ is the differential action entropy of a policy π, and $\mathcal{H}(\rho(\cdot|x,a)) = -\sum_k \rho(k|x,a) \log \rho(k|x,a)$ is the entropy of the responsibility function, defined as

$$\rho(k|x,a) = w(k|x)\, \pi_k(a|x_P)/\pi_m(a|x) , \quad (4)$$

which is the probability that a performed action a at option-augmented state x has been generated from option k. We define a sticky prior over options with $\Omega^{\mathrm{stick}}_{k^-} > 0$

$$w_{\mathrm{prior}}(k|x=(s,k^-)) = \exp\big(\Omega^{\mathrm{stick}}_{k^-}\mathbf{1}\{k=k^-\}\big) \,/\, \sum_j \exp\big(\Omega^{\mathrm{stick}}_{k^-}\mathbf{1}\{j=k^-\}\big) . \quad (5)$$

The goal defined by Eqs. 2,3 thus consists in maximizing the sum of future rewards, while shaping the hierarchy through three regularization terms—the weighting coefficients in Eq. 3 are non-negative. Each of the regularizers plays a role in generating option diversity, specialization, or temporal persistence (stickiness). In order of appearance in Eq. 3: (a) the mixture entropy promotes option diversity by promoting diverse action paths; (b) the negative option responsibility entropy favors option specialization, as this entropy is small if just a small range of actions is consistent with choosing a specific option; and (c) the negative KL divergence promotes reusing the previously active option through the sticky prior in Eq. 5, and therefore it produces temporal persistence.

Using mixture entropy to promote diversity has been used before (Habib et al., 2025), but not in the option framework. Analytical responsibilities and their weighted updates follow probabilistic option inference and HiREPS (Daniel et al., 2016; 2012); see Appendix A.1. The sticky-prior term is, to our knowledge, novel in the context of option discovery. We also note that the degree of option diversity is shaped by the information asymmetry between the controller and the options: because options have only partial information, their ability to specialize to the training task is constrained.

Eq. 2 can be recursively expressed in the form of the Bellman equation

$$V(x) = R(x;\pi,w) + \sum_k w(k|x) \int da\, \pi_k(a|x_P) Q(x,k,a) \quad (6)$$

$$Q(x,k,a) \equiv r(s,a) + \gamma \int ds'\, p(s'|s,a) V(x'), \qquad x' = (s',k) . \quad (7)$$

where $V(x')$ denotes the value at the next option-augmented state $x' = (s',k)$. Here, $Q(x,k,a)$ is the option-action value of choosing option k and action a at option-augmented state x. It includes the immediate reward and then evaluates the continuation at $x' = (s',k)$, so future keep-or-switch decisions remain part of the value function. Therefore, the objective can be expressed as maximizing the value function over option policies $\pi = \{\pi_1, ..., \pi_K\}$ and the controller w.

At fixed x, using the mixture-entropy identity $\mathcal{H}(\pi_m) = \mathbb{E}_{k\sim w}\mathcal{H}(\pi_k) + \mathcal{H}(w) - \mathbb{E}_{a\sim\pi_m}\mathcal{H}(\rho)$ and expanding the KL, Eq. 3 becomes

$$\begin{aligned} R(x;\pi,w) = {} & \phi_1\, \mathbb{E}_{k\sim w(\cdot|x)}\big[\mathcal{H}(\pi_k(\cdot|x_P))\big] - \phi_2\, \mathbb{E}_{a\sim\pi_m(\cdot|x)}\left[\mathcal{H}(\rho(\cdot|x,a))\right] \\ & + \phi_3\, \mathcal{H}(w(\cdot|x)) + \lambda \sum_k w(k|x) \log w_{\mathrm{prior}}(k|x) . \end{aligned} \quad (8)$$

where $\mathcal{H}(w(\cdot|x)) = -\sum_k w(k|x) \log w(k|x)$ is the entropy of the controller, with $\phi_1 := \alpha$, $\phi_2 := \alpha + \nu$, and $\phi_3 := \alpha + \lambda$. Thus, the three effective coefficients are coupled: $\phi_2 - \phi_1 = \nu$ and $\phi_3 - \phi_1 = \lambda$. Appendix A.2 rewrites the controller and responsibility entropies using $I(K;A \mid x)$ to compare with prior objectives.

2.1 Relation to Classical Options

Classical options are defined by an initiation set $I_k \subseteq \mathcal{S}$, an option policy π^{classic}_k, and a termination function $\beta_k : \mathcal{S} \to [0,1]$ (Sutton et al., 1999). Under standard call-and-return execution, the high-level controller is consulted only when the current option terminates. However, this controller has

an equivalent per-step formulation: denoting the probability of initiating an option given the state by $\mu(k|s)$, the per-step controller can be written as

$$w_{\text{classic}}(k|s,k^-) = (1-\beta_{k^-}(s))\,\mathbf{1}\{k=k^-\} + \beta_{k^-}(s)\mu(k|s). \tag{9}$$

This induced controller shows how initiation and termination appear in per-step form. Thus, classical call-and-return options factor switching into termination through β_{k^-} and reselection through μ. The reselection policy $\mu(k|s)$ does not condition on the previous option k^-. In contrast, our formulation removes the restricted factorization in Eq. 9 and learns $w(k|s,k^-)$ directly. Thus, in our Markovian framework, $w(k^-|s,k^-)$ is the probability of persistence, and $w(k|s,k^-)$ for $k \neq k^-$ is the probability of switching from k^- to k; these probabilities do not need to be expressible as Eq. 9. Therefore, the probability of entering option k may depend on the option from which the controller is switching, which is not captured by a single state-only initiation set I_k together with a reselection policy $\mu(k|s)$. Our temporal persistence term in Eq. 3 favors that the controller has mostly transitions within options and thus high-level control remains sparse, but its transitions are much more flexible than the constrained version in Eq. 9.

The same distinction as the one described before applies to the option policies. Classical option policies have the form $\pi_k^{\text{classic}}(a|s)$, whereas our options may use the option-augmented input, $\pi_k(a|s,k^-)$. Thus both option selection and option execution can condition on the previously active option. The option-critic line of work (Bacon et al., 2017; Klissarov et al., 2017), and subsequent end-to-end option learning, keep the constrained factorization of Eq. 9. This approximation preserves the classical call-and-return structure, but limits the flexibility of option switching because the reselection policy μ does not condition on k^-. Our controller instead treats keep-or-switch decisions as ordinary Markov decisions on the option-augmented state. This information is important for stable transitions between options, especially in unstable virtual robots.

2.2 Practical training algorithm

This section turns the objective in Eq. 2 into a practical training algorithm. Our algorithm does not optimize Q, π, and w all at once. We instead use a policy-iteration style alternation. At each policy-improvement step, we hold the critic Q and responsibilities ρ fixed. The responsibility ρ is not a separate learned model; it is recomputed from the current controller and option policies using Eq. 4. With these quantities fixed, Proposition 1 in Appendix A.2 lets us update each option independently and then update the controller. Each subproblem is entropy-regularized and concave under $\phi_1 > 0$ for the option updates and $\phi_3 > 0$ for the controller update, so its maximizer is a softmax over actions or options (Haarnoja et al., 2018); Appendix A.2 derives the exact forms. In continuous action spaces, their normalizers are action integrals with no closed form, so we approximate them with sampled candidate actions.

Our algorithm is an off-policy actor–critic (Konda & Tsitsiklis, 2003) trained from a replay buffer $\mathcal{D}$ of past transitions (Mnih et al., 2015). We write $Q_\xi, \pi_\theta, w_\psi$ generically; Appendix A.2 gives the clipped twin-critic and target-network implementation. Each training iteration updates the critic first, then the option policies, and finally the controller. The objective in Eq. 2 is written for a general reward $r(s,a)$. In our experiments, $g \in \mathcal{G}$ denotes the current goal (e.g., a navigation target or manipulation target), which we make explicit by using the notation $r(s,a,g)$. The controller and critic are goal-conditioned, $w_\psi(k|x,g)$ and $Q_\xi(x,k,a,g)$, while the option policies remain goal-agnostic, $\pi_{\theta,k}(a|x_P)$. As defined in Section 2, $x = (s,k^-)$ is the option-augmented state and x_P is the masked option input. Thus options learn reusable low-level behaviors from x_P, while the controller and critic use (x,g) to compose options for the current task.

Critic update. The critic provides the value Q used by the option and controller updates. We train it by Bellman regression on replay transitions (x,k,a,r,x',c,g), where $c \in \{0,1\}$ is the nonterminal mask. The one-step Bellman target is

$$\widehat{V}(x',g) = \mathbb{E}_{k'\sim w_\psi, a'\sim\pi_{\theta,k'}}\left[Q_\xi(x',k',a',g)\right] + R(x',g;\pi_\theta,w_\psi), \quad y = r + \gamma c\widehat{V}(x',g). \tag{10}$$

Here $R(x',g;\pi_\theta,w_\psi)$ is the regularizer in Eq. 3, evaluated at the next option-augmented state and current goal. The critic is fit by minimizing $\mathcal{L}_Q = \mathbb{E}_{\mathcal{D}}(Q_\xi(x,k,a,g) - y)^2$.

Option update. For fixed Q, controller w, and responsibilities ρ, each option has its own policy-improvement problem. For each option, this gives a target that puts probability on actions that are high-value under Q and already attributed to option k by ρ. The exact target, denoted by a star, is

$$\pi_k^\star(a|x,g) \propto \rho(k|x,a,g)^{\phi_2/\phi_1} \exp(Q(x,k,a,g)/\phi_1)\,. \tag{11}$$

Here $\rho(k|x,a,g)$ is the responsibility from Eq. 4, the posterior probability that option k generated action a. From Eq. 8, $\phi_1 = \alpha$ controls how concentrated the option is on high-Q actions, while $\phi_2/\phi_1 = 1 + \nu/\alpha$ controls how strongly the option specializes on actions for which it is already responsible. Appendix A.2 derives this closed-form target. The target $\pi_k^\star(\cdot|x,g)$ is a full-context density, but the parametric option $\pi_{\theta,k}(\cdot|x_P)$ only observes x_P. Fitting one masked-input network to these targets across replay states enforces the information asymmetry during training. We can update $\pi_{\theta,k}$ toward this target either directly using reparameterized actions and pathwise gradients, or by fitting it through weighted likelihood to sampled candidate actions. We use the reparameterized update in our experiments because we found it more stable; Appendix A.2 gives both updates.

Controller update. After option updates, the controller learns to compose them at (x,g). For fixed Q and analytical option targets, the decomposition in Appendix A.2 assigns each option the score

$$M_k(x,g) := \mathbb{E}_{a\sim\pi_k^\star(\cdot|x,g)}\left[Q(x,k,a,g) + \phi_2 \log \rho(k|x,a,g)\right] + \phi_1\, \mathcal{H}(\pi_k^\star(\cdot|x,g)). \tag{12}$$

This score is the option-local objective value at the analytical target, so it summarizes option k's usefulness at (x,g). The optimal controller is a softmax over these scores. With $\lambda > 0$, the sticky prior favors reselecting the previous option k^-. The exact controller target is

$$w^\star(k|x,g) \propto \exp(M_k(x,g)/\phi_3)\, w_{\text{prior}}(k|x)^{\lambda/\phi_3}. \tag{13}$$

From Eq. 8, $\phi_3 = \alpha + \lambda$ is the controller softness coefficient: larger values spread probability more broadly across options, while smaller values make the controller more committed to high-score options. The ratio λ/ϕ_3 controls the strength of the sticky-prior effect. Appendix A.2 derives this closed-form target. Computing $M_k(x,g)$ exactly is intractable because it requires integrating over all continuous actions for each option. This option score can be written as $M_k(x,g) = \phi_1 \log F_k(x,g)$, where $F_k(x,g)$ is the normalizer of the exact option target in Eq. 11. We estimate $F_k(x,g)$ by importance sampling as detailed in Appendix A.2, giving a sampled option score $\widehat{M}_k(x,g)$. We then replace $M_k(x,g)$ by $\widehat{M}_k(x,g)$ in Eq. 13 and fit w_ψ to the resulting sampled controller target by KL minimization.

3 Experiments

Our experiments test whether the learned options can be reused in downstream tasks. We adopt the same frozen-option protocol in every domain: after training-task learning, the option library is fixed and only a new controller and critic are trained for downstream transfer. The downstream learner selects options proportionally to $\exp(Q(s,g,k^-,k)/\tau)$, with τ tuned automatically (Haarnoja et al., 2017). During transfer, the controller holds each frozen option for H primitive steps, preserving temporal abstraction: $H = 10$ for the long-horizon navigation and Barkour tasks, and $H = 5$ for the shorter Panda task. Full environment and training details are deferred to Appendices A.3 and A.4. We choose four domain families because each tests a different kind of reuse (Fig. 6). AntMaze tests whether $K = 4$ direction-like options suffice to reach goals in new environments. HumanoidMaze adds the challenge of balancing a less stable body. Barkour tests whether a compact $K = 3$ locomotion library trained on a simplified route covers more variable downstream waypoints, using the Google Barkour v0 model (Zakka et al., 2022) and Barkour benchmark (Caluwaerts et al., 2023). Panda uses the Franka Emika Panda Description from MuJoCo Menagerie (Zakka et al., 2022) to test transfer between standard but different manipulation tasks: $K = 16$ options learned on pick-and-orient are recomposed for unseen table pushing.

3.1 Baselines and evaluation

We choose baselines to answer five different questions about frozen-option transfer. ***(1) Does standard option learning transfer well when options see the full task information?*** Option-Critic (OC)

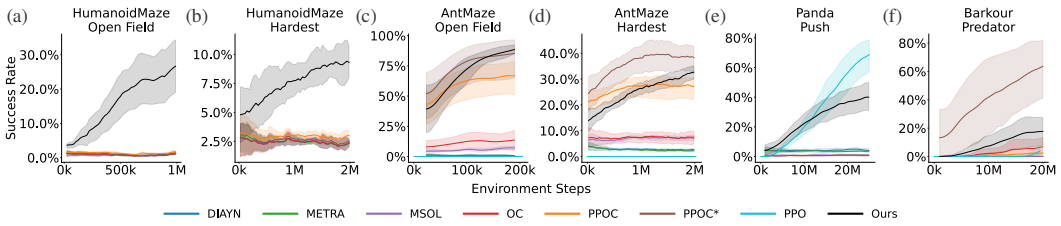


Figure 2: Frozen-option transfer results across four domains. (a,b) HumanoidMaze open field and hardest. (c,d) AntMaze open field and hardest. (e) Panda push. (f) Barkour predator. Mean over 5 agents; 50 randomized episodes per evaluation. Shading denotes 95% bootstrap CIs.

(Bacon et al., 2017) and PPOC (Klissarov et al., 2017) test this case. Both learn options from the training task while giving the options the full state and goal. ***(2)*** *Is masking task information from the options enough by itself?* PPOC* tests this case. It uses the same PPO-style option-learning baseline as PPOC, but its options observe only proprioceptive state. This tests whether masking alone is enough, without TASRO's previous-option conditioning, persistence bias, or responsibility-based specialization. ***(3)*** *Does an existing method designed for multitask frozen-option transfer work in our single task training setting?* MSOL (Igl et al., 2020) tests this case. The original method learns options from multiple tasks and was evaluated with discrete actions. Here we use the same single task training protocol as the other methods, adapt MSOL to continuous control, and freeze its prior goal-agnostic option library for transfer. ***(4)*** *Can unsupervised skill discovery produce reusable frozen behaviors?* DIAYN (Eysenbach et al., 2019) and METRA (Park et al., 2024) test this case. ***(5)*** *How does transfer compare to learning from scratch in the downstream task?* Flat PPO tests this case. It trains a flat policy directly on the downstream task, without using frozen options. This shows whether frozen options help early in training compared with learning the downstream task from scratch. Appendix A.4 gives full configurations, adaptations, and budgets. Appendix A.6 shows training-task learning before freezing.

3.2 Transfer results

We evaluate frozen-option transfer in the four domains (Fig. 2). HumanoidMaze is the clearest test of option reusability due to the agent's unstable body dynamics. Here, TASRO largely outperforms all tested methods, including a flat policy (Fig. 2a,b). Consistent with this performance, forced-option rollouts, which keep one option continuously active throughout an episode, show stable, diverse, and specialized TASRO behaviors, while the baselines show less diversity and specialization (Fig. 3a).

In AntMaze, the quadruped's greater stability makes navigation more tractable. TASRO and PPOC* are the two strongest frozen-option methods, with TASRO attaining the higher open-field mean and PPOC* the higher hardest-layout mean (Fig. 2c,d). However, TASRO learns more directionally distinct behaviors (Fig. 3b). With $H = 10$ and episodes shortened from 1000 to 500 steps, TASRO retains $86.0 \pm 5.5\%$ success, while PPOC* drops to $55.9 \pm 13.8\%$, consistent with PPOC* relying more on long episodes than precise directional specialization. Separately, varying H shows a broad TASRO performance plateau from 5 to 15 steps, with PPOC* also competitive but more variable across seeds (Appendix A.9). The task-exposed PPOC variant also performs well here, reflecting AntMaze's relative ease.

Panda arm manipulation tests a different form of reuse: transferring options between two different arm-manipulation objectives. The training task is pick-and-orient and the downstream task is table pushing, but the options see the same proprioceptive view in both tasks. Under frozen-option transfer, TASRO is the only frozen library that makes consistent progress, reaching about $40.0 \pm 9.6\%$ success (Fig. 2e). TASRO's advantage over frozen-library baselines persists across the horizon sweep, with its best performance at $H = 5$ and lower success at longer horizons (Appendix A.9). Flat PPO is the only baseline that does well on this task, reaching $68.6 \pm 10.8\%$ when trained end-to-end, but it is not a frozen-library method. OC/PPOC are omitted here because pick-to-push changes the goal distribution seen by their goal-conditioned options.

Barkour is the main exception. With $K = 3$ options trained on a fixed route, TASRO reaches $17.7 \pm 9.6\%$ predator-transfer success, compared with $63.6 \pm 19.7\%$ for PPOC* (Fig. 2f). To test

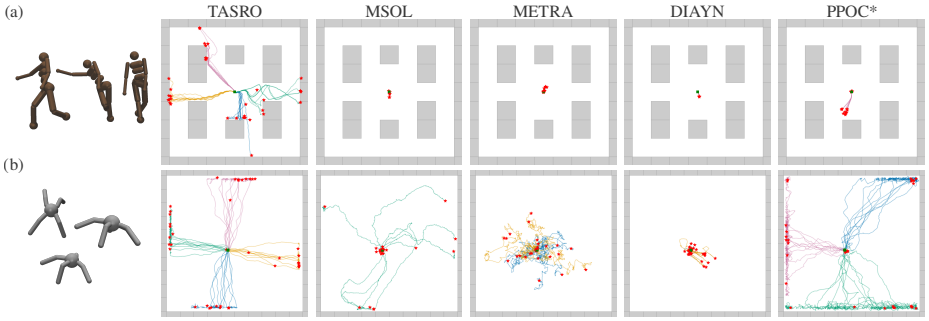


Figure 3: Forced-option trajectories in maze domains. Each option is held active for an entire episode. (a) HumanoidMaze training layout. (b) AntMaze open-field transfer; additional AntMaze seeds are shown in Fig. 12. Red points indicate episode termination (timeout or loss of balance).

whether this gap reflects insufficient library capacity, we increase K from 3 to 8. TASRO develops a broader, actively used repertoire (see Fig. 9 in Appendix A.7), yet PPOC* retains higher mean transfer success at the shorter horizon $H = 5$, 59.6% versus 42.0%, although confidence intervals overlap (Table 6). Thus, $K = 3$ alone does not explain the gap, and library breadth does not by itself ensure downstream utility.

Training-task success alone does not predict frozen reuse: TASRO, PPOC, and PPOC* all learn the AntMaze training task, while TASRO learns the Barkour training route but performs worse than PPOC* after the libraries are frozen (Appendix A.6).

Taken together, TASRO is the only frozen-library method that avoids a degenerate option library across all four domain families. PPOC* is the closest competitor, but its options do not support useful transfer in Panda arm manipulation or HumanoidMaze. The forced-option plots show more distinct TASRO behaviors in both maze domains (Fig. 3a,b). In AntMaze, TASRO actively uses its library, reaching 3.95 effective options out of four during downstream learning. During evaluation, options remain active for 30.5 primitive steps on average, about three times the mandatory horizon $H = 10$, combining temporal persistence with within-episode switching. By contrast, PPOC, DIAYN, and METRA use fewer effective options and rarely switch within episodes (Appendices A.8 and A.11).

3.3 Ablations

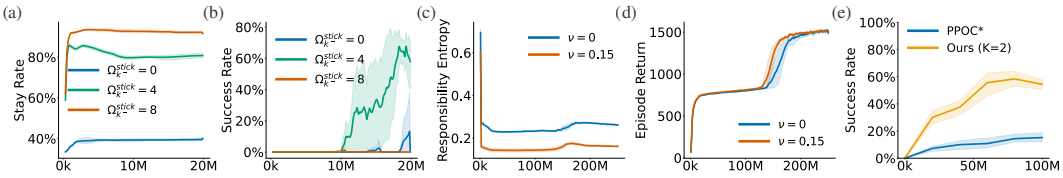


Figure 4: Training-task design checks. (a,b) Barkour stay rate, $\Pr(k = k^-)$, and success across sticky-prior strengths $\Omega^{\text{stick}}_{k^-}$. (c,d) Panda responsibility entropy and return for $\nu = 0$ versus $\nu = 0.15$. (e) AntMaze success with $K = 2$.

Our training-task ablations test four mechanisms: temporal persistence, responsibility, information asymmetry, and conditioning on k^- (Fig. 4). First, the sticky prior controls temporal persistence. In Barkour, increasing $\Omega^{\text{stick}}_{k^-}$ raises the stay rate, but training success is highest at intermediate stickiness, showing that both too little and too much persistence can hinder learning (Fig. 4a,b). Second, the responsibility term should sharpen option assignment. In the Panda training task, using $\nu = 0.15$ lowers responsibility entropy from 0.26 to 0.16 (Fig. 4c). Both settings eventually learn the training pick task, but the responsibility-weighted run learns faster and reaches a slightly higher final return (Fig. 4d).

Third, masking s_{TR} should prevent options from encoding the training maze directly. With default option input, AntMaze options observe s_P and k^-, but not xy position. As expected, the forced options look position-blind: when an option is forced into a wall, it keeps pushing rather than navigating the maze (Fig. 5a). When s_{TR} is given to options, they avoid walls directly (Fig. 5b). With $K = 6$, extra options specialize without collapsing (Fig. 5c).

Fourth, while PPOC* works well with $K = 4$, with $K = 2$ it produces poorly organized trajectories: PPOC* tends to make each option broad and stochastic, because random movement can cover more targets than two narrow behaviors (Fig. 5d). In contrast, TASRO trajectories generated under the same current option can differ depending on which option was active before: rows fix the previous option k^-, and columns fix the current option k (Fig. 5e). This gives the controller extra flexibility.

With $K = 2$, TASRO reaches 54.4% training success versus 15.2% for PPOC* (Fig. 4e), despite PPOC* using the same information asymmetry. Thus, asymmetry is central but not sufficient to organize the library, consistent with TASRO's transition flexibility (Fig. 5e). In frozen transfer, removing asymmetry produces the largest drop, from $91.2 \pm 4.2\%$ to $34.8 \pm 4.4\%$, while increasing ϕ_1 from 0.1 to 0.6 softens the option targets and raises success from 38.0% to 95.6%. Responsibility specialization and k^- conditioning beyond the sticky prior also appear beneficial, while performance is relatively insensitive to controller softness (Appendix A.10).

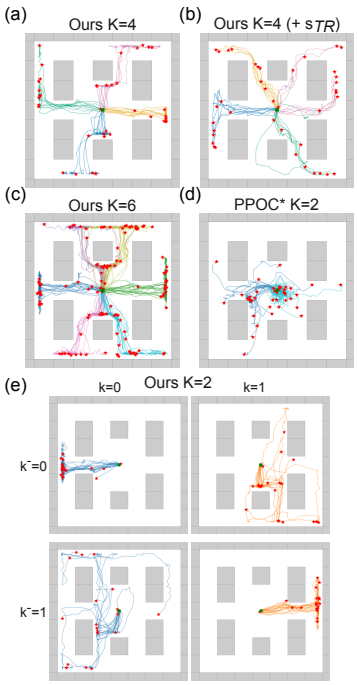


Figure 5: Training-task AntMaze option visualizations.

4 Conclusion and Limitations

We introduced TASRO to test whether frozen-option transfer improves when options encode reusable task-agnostic behaviors and the controller handles goal-conditioned composition. The main finding is that TASRO is the only frozen-library method that avoids a degenerate option library across all four domain families. This supports a specific view of HRL: option discovery should optimize for recomposability under a new controller, not only for training-task performance. TASRO implements this view through information asymmetry, a responsibility-weighted regularizer, a sticky prior, and a fully option-switching Markovian structure. Unlike prior option-augmented frameworks that keep reselection independent of the previously active option, TASRO makes switching itself a Markov decision on (s_t, k_{t-1}), so both continuing and switching can depend on the current state and previous option. The transfer results back this design across navigation, locomotion, and manipulation, with the largest gains in harder settings where reuse demands both stable low-level behavior and flexible recomposition. The ablations support the mechanism: hiding task-relevant information from the options prevents them from absorbing the training task, and conditioning on the previously active option produces more organized behavior under small option libraries.

TASRO is not limited to frozen reuse: since it learns options and controller jointly, the same machinery could support downstream fine-tuning, as in HiPPO (Li et al., 2020). We chose frozen transfer because it more strictly tests whether learned low-level behaviors are reusable without modification; frozen reuse and unrestricted downstream learning answer different questions. The central limitation is choosing a training task that exposes behaviors useful downstream. AntMaze and HumanoidMaze test closely aligned tasks; Panda and Barkour test less aligned transfer. TASRO transfers in Panda, but Barkour shows that option diversity and active use alone do not guarantee better transfer. The main open problem is thus choosing or designing a training task for a downstream task family. Selecting K and stickiness remains heuristic and requires empirical tuning. Overall, TASRO points toward HRL systems whose low-level behaviors are pretrained for recomposability, inspected separately, and reused through new goal-conditioned controllers.

AI USE STATEMENT

In accordance with the ICLR 2027 AI policy, we used a large language model only to proofread the manuscript, that is, to correct grammar and improve the clarity of text written by the authors. All research ideas, theory, experiments, and analysis are the authors' own. The authors reviewed all edits and take full responsibility for the content of this paper.

REPRODUCIBILITY STATEMENT

Complete derivations and proofs are provided in Appendix A.2, including the exact policy targets and frozen-responsibility surrogate analysis. Environment definitions, transfer protocols, training procedures, hyperparameters, baseline configurations, seed counts, and evaluation details are documented in Appendices A.3–A.11. Anonymized source code and the execution environment are provided in the supplementary material.

REFERENCES

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pp. 5048–5058, 2017. URL https://papers.nips.cc/paper/2017/hash/453fadbd8a1a3af50a9df4df899537b5-Abstract.html.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, volume 31, pp. 1726–1734, 2017. doi: 10.1609/aaai.v31i1.10916.

Michał Bortkiewicz, Władysław Pałucki, Vivek Myers, Tadeusz Dziarmaga, Tomasz Arczewski, Łukasz Kuciński, and Benjamin Eysenbach. Accelerating goal-conditioned RL algorithms and research. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2408.11052.

Matthew M. Botvinick, Yael Niv, and Andrew G. Barto. Hierarchically organized behavior and its neural foundations: A reinforcement-learning perspective. *Cognition*, 113(3):262–280, 2009. doi: 10.1016/j.cognition.2008.08.011.

Ken Caluwaerts, Atil Iscen, J. Chase Kew, Wenhao Yu, Tingnan Zhang, Daniel Freeman, Kuang-Huei Lee, Lisa Lee, Stefano Saliceti, Vincent Zhuang, Nathan Batchelor, Steven Bohez, Federico Casarini, Jose Enrique Chen, Omar Cortes, Erwin Coumans, Adil Dostmohamed, Gabriel Dulac-Arnold, Alejandro Escontrela, Erik Frey, Roland Hafner, Deepali Jain, Bauyrjan Jyenis, Yuheng Kuang, Edward Lee, Linda Luu, Ofir Nachum, Ken Oslund, Jason Powell, Diego Reyes, Francesco Romano, Feresteh Sadeghi, Ron Sloat, Baruch Tabanpour, Daniel Zheng, Michael Neunert, Raia Hadsell, Nicolas Heess, Francesco Nori, Jeff Seto, Carolina Parada, Vikas Sindhwani, Vincent Vanhoucke, and Jie Tan. Barkour: Benchmarking animal-level agility with quadruped robots, 2023. URL https://arxiv.org/abs/2305.14654.

Christian Daniel, Gerhard Neumann, and Jan Peters. Hierarchical relative entropy policy search. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 22 of *Proceedings of Machine Learning Research*, pp. 273–281. PMLR, 2012. URL https://proceedings.mlr.press/v22/daniel12.html.

Christian Daniel, Herke van Hoof, Jan Peters, and Gerhard Neumann. Probabilistic inference for determining options in reinforcement learning. *Machine Learning*, 104(2–3):337–357, 2016. doi: 10.1007/s10994-016-5580-x.

Peter Dayan and Geoffrey E. Hinton. Feudal reinforcement learning. In *Advances in Neural Information Processing Systems 5 (NIPS)*, volume 5, pp. 271–278. Morgan-Kaufmann, 1992. URL https://proceedings.neurips.cc/paper/1992/hash/d14220ee66aeec73c49038385428ec4c-Abstract.html.

Thomas G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000. doi: 10.1613/jair.639.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://openreview.net/forum?id=SJx63jRqFm.

Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. Meta learning shared hierarchies. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://openreview.net/forum?id=SyX0IeWAW.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1587–1596. PMLR, 2018. URL https://proceedings.mlr.press/v80/fujimoto18a.html.

Alexandre Galashov, Siddhant M. Jayakumar, Leonard Hasenclever, Dhruva Tirumala, Jonathan Schwarz, Guillaume Desjardins, Wojciech M. Czarnecki, Yee Whye Teh, Razvan Pascanu, and Nicolas Heess. Information asymmetry in KL-regularized RL. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://openreview.net/forum?id=S1lqMn05Ym.

Francisco M. Garcia, Chris Nota, and Philip S. Thomas. Learning reusable options for multi-task reinforcement learning, 2020. URL https://arxiv.org/abs/2001.01577.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1352–1361. PMLR, 2017. URL https://proceedings.mlr.press/v70/haarnoja17a.html.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

Yamen Habib, Dmytro Grytskyy, and Rubén Moreno-Bote. Unsupervised action-policy quantization via maximum entropy mixture policies with minimum entropy components. In *18th European Workshop on Reinforcement Learning (EWRL)*, 2025. URL https://openreview.net/forum?id=HhbHw2yInZ. EWRL 2025 poster.

Nicolas Heess, Greg Wayne, Yuval Tassa, Timothy Lillicrap, Martin Riedmiller, and David Silver. Learning and transfer of modulated locomotor controllers, 2016. URL https://arxiv.org/abs/1610.05182.

Maximilian Igl, Andrew Gambardella, Jinke He, Nantas Nardelli, N. Siddharth, Wendelin Böhmer, and Shimon Whiteson. Multitask soft option learning. In *Proceedings of the Thirty-Sixth Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 124 of *Proceedings of Machine Learning Research*, pp. 969–978. PMLR, 2020. URL https://proceedings.mlr.press/v124/igl20a.html.

Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214, 1994. doi: 10.1162/neco.1994.6.2.181.

Martin Klissarov, Pierre-Luc Bacon, Jean Harb, and Doina Precup. Learning options end-to-end for continuous action tasks, 2017. URL https://arxiv.org/abs/1712.00004.

Vijay R. Konda and John N. Tsitsiklis. On actor-critic algorithms. *SIAM Journal on Control and Optimization*, 42(4):1143–1166, 2003. doi: 10.1137/S0363012901385691.

George Konidaris and Andrew G. Barto. Building portable options: Skill transfer in reinforcement learning. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 895–900, 2007. URL https://www.ijcai.org/Proceedings/07/Papers/144.pdf.

Alexander C. Li, Carlos Florensa, Ignasi Clavera, and Pieter Abbeel. Sub-policy adaptation for hierarchical reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=ByeWogStDS.

Chang Li, Dongjin Song, and Dacheng Tao. HiT-MDP: Learning the SMDP option framework on MDPs with hidden temporal embeddings. In *International Conference on Learning Representations*, 2023. URL https://songdj.github.io/publication/iclr-23-a/iclr-23-a.pdf.

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. In *International Conference on Learning Representations (ICLR)*, 2016. URL https://arxiv.org/abs/1509.02971.

Timothy A. Mann and Shie Mannor. Scaling up approximate value iteration with options: Better policies with fewer iterations. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, volume 32 of *Proceedings of Machine Learning Research*, pp. 127–135. PMLR, 2014. URL https://proceedings.mlr.press/v32/mann14.html.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. doi: 10.1038/nature14236.

Nirbhay Modhe, Prithvijit Chattopadhyay, Mohit Sharma, Abhishek Das, Devi Parikh, Dhruv Batra, and Ramakrishna Vedantam. IR-VIC: Unsupervised discovery of sub-goals for transfer in RL. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2022–2028, 2020. doi: 10.24963/ijcai.2020/280. URL https://www.ijcai.org/proceedings/2020/280.

Seohong Park, Jongwook Choi, Jaekyeom Kim, Honglak Lee, and Gunhee Kim. Lipschitz-constrained unsupervised skill discovery. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=BGvt0ghNgA.

Seohong Park, Oleh Rybkin, and Sergey Levine. METRA: Scalable unsupervised RL with metric-aware abstraction. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=c5pwL0Soay.

Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. MCP: Learning composable hierarchical control with multiplicative compositional policies. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pp. 3681–3692, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/95192c98732387165bf8e396c0f2dad2-Abstract.html.

Boris T. Polyak and Anatoli B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992. doi: 10.1137/0330046.

Ahmed H. Qureshi, Jacob J. Johnson, Yuzhe Qin, Taylor Henderson, Byron Boots, and Michael C. Yip. Composing task-agnostic policies with deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=H1ezFREtwH.

Sasha Salter, Kristian Hartikainen, Walter Goodwin, and Ingmar Posner. Priors, hierarchy, and information asymmetry for skill transfer in reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=0v4VkCSkHNm.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1312–1320. PMLR, 2015. URL https://proceedings.mlr.press/v37/schaul15.html.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=HJgLZR4KvH.

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. doi: 10.1016/S0004-3702(99)00052-1.

Dhruva Tirumala, Alexandre Galashov, Hyeonwoo Noh, Leonard Hasenclever, Razvan Pascanu, Jonathan Schwarz, Guillaume Desjardins, Wojciech Marian Czarnecki, Arun Ahuja, Yee Whye Teh, and Nicolas Heess. Behavior priors for efficient reinforcement learning. *Journal of Machine Learning Research*, 23(221):1–68, 2022. URL https://jmlr.org/papers/v23/20-1038.html.

Vivek Veeriah, Tom Zahavy, Matteo Hessel, Zhongwen Xu, Junhyuk Oh, Iurii Kemaev, Hado van Hasselt, David Silver, and Satinder Singh. Discovery of options via meta-learned subgoals. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, pp. 29861–29873, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/fa246d0262c3925617b0c72bb20eeb1d-Abstract.html.

Markus Wulfmeier, Dushyant Rao, Roland Hafner, Thomas Lampe, Abbas Abdolmaleki, Tim Hertweck, Michael Neunert, Dhruva Tirumala, Noah Siegel, Nicolas Heess, and Martin Riedmiller. Data-efficient hindsight off-policy option learning. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11340–11350. PMLR, 2021. URL https://proceedings.mlr.press/v139/wulfmeier21a.html.

Kevin Zakka, Yuval Tassa, and MuJoCo Menagerie Contributors. MuJoCo menagerie: A collection of high-quality simulation models for MuJoCo. GitHub repository, 2022. URL http://github.com/google-deepmind/mujoco_menagerie.

Jesse Zhang, Haonan Yu, and Wei Xu. Hierarchical reinforcement learning by discovering intrinsic options. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=r-gPPHEjpmw.

Shangtong Zhang and Shimon Whiteson. DAC: The double actor-critic architecture for learning options. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://papers.neurips.cc/paper/8475-dac-the-double-actor-critic-architecture-for-learning-options.pdf.

A Appendix

A.1 Related work

Prior work on temporal abstraction has made it possible to learn reusable behavior libraries, but it has left open a specific design question that matters for whether those behaviors remain useful after transfer: what task information should low-level options be allowed to observe? Existing methods have explored three broad responses to this question: some leave low-level inputs largely unrestricted, some restrict low-level information to promote reusable behavior, and others remove task reward entirely during skill pretraining.

The options framework formalizes temporally extended behavior through intra-option policies, a policy over options, and termination rules (Sutton et al., 1999). End-to-end option learning, including the option-critic architecture and its continuous-action extensions, trains these components jointly to maximize task reward (Bacon et al., 2017; Klissarov et al., 2017). What these methods leave unconstrained is the input design of the low level: by default, option policies may observe the same goals or task-defining state variables available to the rest of the agent. This makes it easy for the low level to encode task-specific solutions directly, so the hierarchy does not itself protect option reuse. DAC studies options through augmented MDPs, while HiT-MDP directly models option selection conditioned on the previously active option (Zhang & Whiteson, 2019; Li et al., 2023). Their reported transfer evaluations continue training after a task switch, whereas we freeze options learned on one training task and train only a new controller. TASRO additionally conditions the selected option's actions on the previous option under a restricted state view. Thus, HiT-MDP primarily addresses the MDP representation and optimization of temporal abstraction, while TASRO studies how to structure option libraries for strict frozen recomposition.

Work on portable or shared options targets transfer more explicitly, but often through agent-centric structure, multi-task data, or other sources of cross-task signal rather than through low-level input restriction within a single source problem. Portable options use agent-centric representations and initiation sets to make skills transportable across tasks (Konidaris & Barto, 2007). Other methods assume access to task distributions, multi-task data, or trajectories from previously solved tasks and learn shared option structure from that crosstask signal (Igl et al., 2020; Veeriah et al., 2021; Garcia et al., 2020). MSOL (Igl et al., 2020), in particular, learns transferable soft options in a multitask setting, with evaluation in discrete-action domains.

Restricted low-level information has also been studied more broadly through information asymmetry, where a lower-level or prior policy is intentionally given a narrower observation than the task-facing controller (Galashov et al., 2019; Tirumala et al., 2022; Salter et al., 2023). HO2 also studies this asymmetry by providing task information only to its controller in some experiments (Wulfmeier et al., 2021). Within hierarchical control, the closest structural relatives are methods that also separate goal-conditioned composition from lower-level control. MCP freezes task-agnostic motor primitives pretrained from motion-capture imitation and relearns a goal-conditioned composer at transfer time, but it composes primitives multiplicatively at each primitive step rather than learning temporally extended options from a single source reinforcement-learning problem (Peng et al., 2019). Composing Task-Agnostic Policies assumes a given set of pretrained task-agnostic primitive policies and learns to compose them for new tasks, but it does not discover the primitive library jointly from a single source reinforcement-learning problem (Qureshi et al., 2020). HiPPO uses masked lower-level sub-policies but fine-tunes both hierarchy levels on downstream tasks rather than treating the low level as a frozen reusable library (Li et al., 2020). HIDIO is also close in motivation, but it learns task-agnostic options as continuous latent codes rather than as a small discrete option library, and it does not evaluate frozen-option transfer with a newly trained controller (Zhang et al., 2021). Our training task is goal-conditioned in the standard reinforcement-learning sense (Schaul et al., 2015; Andrychowicz et al., 2017), but the distinguishing choice is not goal conditioning itself; it is restricting that signal to the high-level controller and critic while studying frozen reusable discrete options from a single source problem.

A more radical response to the same problem is to remove task reward entirely during pretraining. DIAYN maximizes mutual information between skill identity and visited states to induce behavioral diversity (Eysenbach et al., 2019); LSD adds a Lipschitz constraint to encourage more dynamic skills (Park et al., 2022); DADS emphasizes skills with predictable outcomes that support downstream planning (Sharma et al., 2020); and METRA uses metric-aware latent abstractions to

scale unsupervised skill discovery (Park et al., 2024). These methods can produce broad behavioral coverage, but the learned behaviors are not optimized for any particular downstream task during pretraining. Our method keeps task reward, but restricts where task information is allowed to enter the hierarchy. In our comparison, DIAYN and METRA learn latent-conditioned policies without task reward and hold one skill fixed throughout each pretraining rollout; task reward is introduced only afterward to compose the frozen skills. TASRO instead uses source-task reward while jointly learning its options and a task-aware controller that may retain or switch them at every step. Thus, source training shapes both the behaviors and their transitions, although both kinds of library serve the same frozen low-level role downstream.

Responsibility-weighted updates arise naturally when option identity is treated as a latent variable. More generally, the responsibility in Eq. 4 is the standard mixture-of-experts posterior probability that a particular expert generated an observed action (Jordan & Jacobs, 1994). Probabilistic option inference treats option identity and termination as latent variables, recovers option components from trajectories with EM-style inference, and extends the same framework to reinforcement learning through weighted policy-search updates (Daniel et al., 2016). HiREPS similarly learns hierarchical mixtures through responsibility-weighted updates under a KL trust region (Daniel et al., 2012). TASRO builds on this responsibility-based option-inference lineage: it adopts the analytical posterior as its action-to-option attribution mechanism and combines it with task-masked option policies, direct per-step control conditioned on the previous option, a sticky prior, and strict frozen-option transfer. Appendix A.2 gives the associated conditional mutual-information interpretation and its relation to IR-VIC.

Each of these families addresses part of the challenge but leaves a specific aspect unresolved: end-to-end option learning leaves low-level inputs unconstrained; transfer-oriented methods rely on external primitive sources, multi-task structure, action-space composition, or different transfer regimes; unsupervised skill discovery removes task reward altogether; and probabilistic methods provide the optimization framework without enforcing the same structural input split.

A.2 Derivations and Update Details

This section derives the TASRO update equations from the regularized Bellman equation in Eq. 6. We first rewrite the local Bellman maximization problem at a fixed option-augmented state as a controller objective plus one objective for each option. We then derive the exact fixed-Q, fixed-responsibility targets for the option policies and the controller, show how the frozen-responsibility update defines a local surrogate, and finally explain how the practical algorithm approximates the continuous action integrals in the closed-form targets with Monte Carlo sampling.

All derivations are stated for a fixed goal g. To keep notation light, we omit g from conditioning arguments in this section. Thus $w(k|x)$, $Q(x,k,a)$, $\rho(k|x,a)$, and $\pi_k^\star(a|x)$ denote the goal-conditioned quantities $w(k|x,g)$, $Q(x,k,a,g)$, $\rho(k|x,a,g)$, and $\pi_k^\star(a|x,g)$. The closed-form option target is derived at the option-augmented state x, using these quantities. The information asymmetry enters when the parametric option is optimized, since $\pi_{\theta,k}(a|x_P)$ observes only x_P. The derivation describes one policy-improvement step: hold the critic Q and responsibility ρ fixed, update the option policies, and then update the controller using the updated option scores.

Local fixed-Q objective. Fix an option-augmented state $x = (s, k^-)$ and a critic $Q(x,k,a)$. The local objective optimized by the policy-improvement step is the Bellman improvement term corresponding to Eq. 6:

$$\mathcal{J}_x(\pi, w) = R(x; \pi, w) + \sum_{k=1}^{K} w(k|x) \int da\, \pi_k(a|x) Q(x,k,a). \tag{14}$$

Here $\pi = \{\pi_1, \ldots, \pi_K\}$ is the option library and w is the controller. The critic is held fixed in this policy-improvement step; critic learning is handled separately by the Bellman regression described in Section 2.2.

Responsibility identity. The mixture policy is

$$\pi_m(a|x) = \sum_{j=1}^{K} w(j|x) \pi_j(a|x). \tag{15}$$

For any action a with $\pi_m(a|x) > 0$, the responsibility of option k is

$$\rho(k|x,a) = \frac{w(k|x)\pi_k(a|x)}{\pi_m(a|x)}. \tag{16}$$

Equivalently,

$$\pi_m(a|x)\rho(k|x,a) = w(k|x)\pi_k(a|x). \tag{17}$$

This identity converts expectations under the mixture policy into controller-weighted expectations under individual options and is used throughout the derivation.

Mixture-entropy decomposition. Using Eq. 17, the mixture entropy can be written in terms of controller entropy, option entropies, and responsibility entropy:

$$\mathcal{H}(\pi_m(\cdot|x)) = \mathcal{H}(w(\cdot|x)) + \sum_{k=1}^{K} w(k|x)\mathcal{H}(\pi_k(\cdot|x)) - \mathbb{E}_{a\sim\pi_m(\cdot|x)}\left[\mathcal{H}(\rho(\cdot|x,a))\right]. \tag{18}$$

The action entropies are differential entropies over the continuous action space, while $\mathcal{H}(w(\cdot|x))$ and $\mathcal{H}(\rho(\cdot|x,a))$ are discrete entropies over options.

Effective coefficients. After applying the mixture-entropy identity, the regularizer is governed by three effective coefficients:

$$\phi_1 := \alpha, \qquad \phi_2 := \alpha + \nu, \qquad \phi_3 := \alpha + \lambda. \tag{19}$$

The coefficient ϕ_1 controls option action-level softness: larger ϕ_1 gives softer option targets, while smaller ϕ_1 makes options more sharply value-seeking. The coefficient ϕ_2 controls responsibility-based specialization: larger ϕ_2 pulls an option more strongly toward actions already attributed to that option. The coefficient ϕ_3 controls controller softness: larger ϕ_3 spreads probability more broadly across options, while smaller ϕ_3 makes the controller more sharply committed to a single option. We assume $\phi_1 > 0$, $\phi_2 \geq 0$, and $\phi_3 > 0$. Under these assumptions, the option and controller subproblems are concave after Q and ρ are fixed, and the frozen-responsibility bound below is nonnegative. The next proposition uses these coefficients to separate the local fixed-Q, fixed-responsibility objective into a controller term and one term for each option.

Objective decomposition. Substituting Eq. 18 into the regularizer $R(x;\pi,w)$ in Eq. 3 groups all terms into either a controller-only term or an option-local term. Here $\pi_k(\cdot|x)$ denotes the target distribution for option k at the option-augmented state x; the masked parametric policy $\pi_{\theta,k}(\cdot|x_P)$ enters only in the sampled updates below.

Proposition 1 (Decomposition of the regularized Bellman objective)**.** *For fixed option-action values $Q(x,k,a)$, let $\rho = \rho^{\pi,w}$ denote the responsibility induced by (π,w) via Eq. 4. The local objective in Eq. 14 then decomposes as*

$$\mathcal{J}_x(\pi,w) = \mathcal{L}_w(w|x) + \sum_{k=1}^{K} w(k|x)\mathcal{L}_k^{\rho}(\pi_k|x), \tag{20}$$

where the controller term is

$$\mathcal{L}_w(w|x) = \phi_3\,\mathcal{H}(w(\cdot|x)) + \lambda\sum_{k=1}^{K} w(k|x)\log w_{\text{prior}}(k|x), \tag{21}$$

and the option-local term for option k is

$$\mathcal{L}_k^{\rho}(\pi_k|x) = \phi_1\,\mathcal{H}(\pi_k(\cdot|x)) + \int da\,\pi_k(a|x)\left[Q(x,k,a) + \phi_2\log\rho(k|x,a)\right]. \tag{22}$$

Proof. Write $w_k = w(k|x)$, $\pi_k(a) = \pi_k(a|x)$, and $\rho_k(a) = \rho(k|x,a)$. Since $\pi_m(a|x) = \sum_j w_j\pi_j(a)$, the responsibility identity gives

$$\pi_m(a|x)\rho_k(a) = w_k\pi_k(a).$$

For points where $w_k\pi_k(a) > 0$, this implies

$$\log \pi_m(a|x) = \log w_k + \log \pi_k(a) - \log \rho_k(a).$$

Points where $w_k\pi_k(a) = 0$ contribute zero to the integrals below under the standard convention $0 \log 0 = 0$. Multiplying by $w_k\pi_k(a)$, integrating over actions, and summing over options gives

$$\begin{aligned}\mathcal{H}(\pi_m(\cdot|x)) &= -\sum_k w_k \int da\, \pi_k(a) \log \pi_m(a|x) \\ &= -\sum_k w_k \log w_k - \sum_k w_k \int da\, \pi_k(a) \log \pi_k(a) + \sum_k w_k \int da\, \pi_k(a) \log \rho_k(a).\end{aligned}$$

The first term is $\mathcal{H}(w(\cdot|x))$, the second term is $\sum_k w_k \mathcal{H}(\pi_k(\cdot|x))$, and the last term is $-\mathbb{E}_{a\sim\pi_m(\cdot|x)}[\mathcal{H}(\rho(\cdot|x,a))]$, because

$$\mathbb{E}_{a\sim\pi_m}[\mathcal{H}(\rho(\cdot|x,a))] = -\sum_k w_k \int da\, \pi_k(a) \log \rho_k(a).$$

This proves Eq. 18.

Substituting this identity into Eq. 3, before expanding the KL term, gives

$$\begin{aligned}R(x;\pi,w) = \alpha\mathcal{H}(w(\cdot|x)) + \alpha \sum_k w_k \mathcal{H}(\pi_k(\cdot|x)) \\ - (\alpha+\nu)\mathbb{E}_{a\sim\pi_m}[\mathcal{H}(\rho(\cdot|x,a))] - \lambda D_{\mathrm{KL}}\left(w(\cdot|x) \| w_{\mathrm{prior}}(\cdot|x)\right).\end{aligned}$$

Expanding the KL term then gives

$$\begin{aligned}R(x;\pi,w) = (\alpha+\lambda)\mathcal{H}(w(\cdot|x)) + \lambda \sum_k w_k \log w_{\mathrm{prior}}(k|x) \\ + \alpha \sum_k w_k \mathcal{H}(\pi_k(\cdot|x)) - (\alpha+\nu)\mathbb{E}_{a\sim\pi_m}[\mathcal{H}(\rho(\cdot|x,a))],\end{aligned}$$

where we used

$$-\lambda D_{\mathrm{KL}}(w(\cdot|x)\|w_{\mathrm{prior}}(\cdot|x)) = \lambda\mathcal{H}(w(\cdot|x)) + \lambda \sum_k w_k \log w_{\mathrm{prior}}(k|x).$$

Using the responsibility identity once more,

$$-\mathbb{E}_{a\sim\pi_m}[\mathcal{H}(\rho(\cdot|x,a))] = \sum_k w_k \int da\, \pi_k(a|x) \log \rho(k|x,a).$$

Adding the fixed-Q term from Eq. 14 and collecting coefficients with $\phi_1 = \alpha$, $\phi_2 = \alpha + \nu$, and $\phi_3 = \alpha + \lambda$ yields Eq. 20 with the terms in Eqs. 21 and 22. □

Closed-form option target. For fixed Q, fixed ρ, and $\phi_1 > 0$, the option-local objective in Eq. 22 is maximized by the density

$$\pi_k^\star(a|x) = \frac{\rho(k|x,a)^{\phi_2/\phi_1} \exp(Q(x,k,a)/\phi_1)}{\int da'\, \rho(k|x,a')^{\phi_2/\phi_1} \exp(Q(x,k,a')/\phi_1)}. \tag{23}$$

To see this, maximize Eq. 22 over densities $\pi_k(\cdot|x)$ subject to $\int da\, \pi_k(a|x) = 1$. The Lagrangian is

$$\begin{aligned}\mathfrak{L}(\pi_k,\zeta) = -\phi_1 \int da\, \pi_k(a|x) \log \pi_k(a|x) \\ + \int da\, \pi_k(a|x)\left[Q(x,k,a) + \phi_2 \log \rho(k|x,a)\right] \\ + \zeta\left(\int da\, \pi_k(a|x) - 1\right).\end{aligned}$$

The stationarity condition with respect to $\pi_k(a|x)$ is

$$-\phi_1\left(\log \pi_k(a|x) + 1\right) + Q(x,k,a) + \phi_2 \log \rho(k|x,a) + \zeta = 0.$$

Solving for the maximizing density gives

$$\pi_k^\star(a|x) \propto \rho(k|x,a)^{\phi_2/\phi_1} \exp(Q(x,k,a)/\phi_1),$$

and normalizing over actions gives Eq. 23. This target is defined at the option-augmented state x. The practical sampled updates below optimize the corresponding masked parametric policy $\pi_{\theta,k}(\cdot|x_P)$.

Closed-form controller target. After maximizing $\mathcal{L}_k^\rho(\pi_k|x)$ for each option, the controller subproblem depends on option k only through the achieved value of this term. Define this value as

$$M_k(x) := \mathcal{L}_k^\rho(\pi_k^\star|x). \tag{24}$$

Equivalently,

$$M_k(x) = \phi_1 \mathcal{H}(\pi_k^\star(\cdot|x)) + \mathbb{E}_{a\sim\pi_k^\star(\cdot|x)}\left[Q(x,k,a) + \phi_2 \log \rho(k|x,a)\right]. \tag{25}$$

This score uses the same fixed responsibility ρ as the option update; responsibility is not recomputed inside this policy-improvement step. The only remaining optimization variable is the controller $w(\cdot|x)$, which maximizes

$$\max_w \ \phi_3 \,\mathcal{H}(w(\cdot|x)) + \lambda \sum_k w(k|x) \log w_{\text{prior}}(k|x) + \sum_k w(k|x) M_k(x). \tag{26}$$

For $\phi_3 > 0$, the solution is

$$w^\star(k|x) = \frac{\exp(M_k(x)/\phi_3) w_{\text{prior}}(k|x)^{\lambda/\phi_3}}{\sum_j \exp(M_j(x)/\phi_3) w_{\text{prior}}(j|x)^{\lambda/\phi_3}}. \tag{27}$$

This follows by optimizing Eq. 26 over the option simplex. The Lagrangian is

$$\begin{aligned}\mathfrak{L}(w,\xi) = &-\phi_3 \sum_k w(k|x) \log w(k|x) + \lambda \sum_k w(k|x) \log w_{\text{prior}}(k|x) \\ &+ \sum_k w(k|x) M_k(x) + \xi \left(\sum_k w(k|x) - 1 \right).\end{aligned}$$

For each option k, stationarity gives

$$-\phi_3 \left(\log w(k|x) + 1\right) + \lambda \log w_{\text{prior}}(k|x) + M_k(x) + \xi = 0.$$

Thus

$$w(k|x) \propto \exp(M_k(x)/\phi_3) w_{\text{prior}}(k|x)^{\lambda/\phi_3},$$

and normalizing over options gives Eq. 27. Thus the controller selects options according to their scores $M_k(x)$, while the sticky prior increases the probability of reselecting the previous option through the factor $w_{\text{prior}}(k|x)^{\lambda/\phi_3}$.

The algorithm instead freezes ρ at the value induced by the current hierarchy. Proposition 2 shows that the resulting surrogate is a tight lower bound; at the exact distribution-space level, this freezing is therefore an EM-style minorization rather than an unaccounted approximation.

Surrogate improvement with frozen responsibilities. The fixed-responsibility update can be viewed as optimizing a surrogate for the local objective. Let $\rho^{\text{old}}(k|x,a)$ be a reference responsibility with full support over options, and define

$$\widetilde{\mathcal{J}}_x^{\rho^{\text{old}}}(\pi, w) := \mathcal{L}_w(w|x) + \sum_k w(k|x) \mathcal{L}_k^{\rho^{\text{old}}}(\pi_k|x), \tag{28}$$

where $\mathcal{L}_k^{\rho^{\text{old}}}$ is Eq. 22 with ρ replaced by ρ^{old}.

Proposition 2 (Frozen-responsibility lower bound)**.** *Assume $\phi_1 > 0$ and $\phi_2 \geq 0$. For any candidate hierarchy (π, w), let $\rho^{\pi,w}$ be the responsibility induced by that hierarchy. Then*

$$\mathcal{J}_x(\pi, w) - \widetilde{\mathcal{J}}_x^{\rho^{\text{old}}}(\pi, w) = \phi_2 \,\mathbb{E}_{a\sim\pi_m(\cdot|x)}\left[D_{\text{KL}}\left(\rho^{\pi,w}(\cdot|x,a) \| \rho^{\text{old}}(\cdot|x,a)\right)\right] \geq 0. \tag{29}$$

The bound is tight when the candidate hierarchy induces the frozen responsibility on the support of $\pi_m(\cdot|x)$.

Proof. The only difference between $\mathcal{J}_x$ and $\widetilde{\mathcal{J}}_x^{\rho^{\text{old}}}$ is the responsibility term in Eq. 22. Using $\pi_m(a|x)\rho^{\pi,w}(k|x,a) = w(k|x)\pi_k(a|x)$, the difference is

$$\begin{aligned}&\phi_2 \sum_k w(k|x) \int da\, \pi_k(a|x) \left[\log \rho^{\pi,w}(k|x,a) - \log \rho^{\text{old}}(k|x,a)\right] \\ &= \phi_2 \int da\, \pi_m(a|x) \sum_k \rho^{\pi,w}(k|x,a) \log \frac{\rho^{\pi,w}(k|x,a)}{\rho^{\text{old}}(k|x,a)},\end{aligned}$$

which is Eq. 29. The inequality follows from nonnegativity of KL divergence and $\phi_2 \geq 0$. □

Choose ρ^{old} as the responsibility induced by the current hierarchy $(\pi^{\mathrm{old}}, w^{\mathrm{old}})$. The surrogate is then tight at the current hierarchy:

$$\widetilde{\mathcal{J}}_x^{\rho^{\mathrm{old}}}(\pi^{\mathrm{old}}, w^{\mathrm{old}}) = \mathcal{J}_x(\pi^{\mathrm{old}}, w^{\mathrm{old}}).$$

In the finite-action tabular case, let $(\pi^{\mathrm{new}}, w^{\mathrm{new}})$ be the exact maximizer of the frozen surrogate, obtained by the closed-form option target in Eq. 23 followed by the controller target in Eq. 27. Combining the lower bound with exact surrogate maximization gives

$$\begin{aligned}\mathcal{J}_x(\pi^{\mathrm{new}}, w^{\mathrm{new}}) &\geq \widetilde{\mathcal{J}}_x^{\rho^{\mathrm{old}}}(\pi^{\mathrm{new}}, w^{\mathrm{new}}) \\ &\geq \widetilde{\mathcal{J}}_x^{\rho^{\mathrm{old}}}(\pi^{\mathrm{old}}, w^{\mathrm{old}}) = \mathcal{J}_x(\pi^{\mathrm{old}}, w^{\mathrm{old}}).\end{aligned}$$

Thus, in the finite-action tabular case, one exact update cannot decrease the local fixed-Q objective $\mathcal{J}_x$. This statement is for the exact finite-action update at fixed Q and fixed x. The practical update uses continuous actions, sampled action candidates, neural option policies and controllers, target networks, and replay. The result therefore explains the structure of the update, but it is not a monotonic-improvement guarantee for the implemented algorithm.

Auxiliary entropy corrections. The three terms in Eq. 3 define the conceptual objective. For the additional temperature flexibility used in our experiments, we extend this regularizer by

$$R_{\mathrm{ext}}(x; \pi, w) = R(x; \pi, w) - \beta \sum_k w(k|x)\mathcal{H}(\pi_k(\cdot|x)) - \eta\mathcal{H}(w(\cdot|x)), \qquad \beta, \eta \geq 0. \tag{30}$$

Repeating the decomposition above changes only the effective coefficients:

$$\phi_1 = \alpha - \beta, \qquad \phi_2 = \alpha + \nu, \qquad \phi_3 = \alpha + \lambda - \eta. \tag{31}$$

We require $\phi_1 > 0$, $\phi_2 \geq 0$, and $\phi_3 > 0$, or equivalently $\beta < \alpha$ and $\eta < \alpha + \lambda$ under the nonnegative coefficients in Eq. 3. The five raw coefficients therefore enter the update rules through four effective controls: $\phi_1, \phi_2, \phi_3, \lambda$.

The extension leaves the option and controller update rules unchanged. The option target remains Eq. 23, with ϕ_1 as its action temperature and ϕ_2 as its responsibility weight, and the controller target remains Eq. 27, with ϕ_3 as its selection temperature and λ as its sticky-prior weight. The frozen-responsibility bound also remains unchanged because $\phi_2 \geq 0$, and the sampled updates below retain the same forms. Thus the extension introduces no additional network or optimization step; it only allows the two temperatures to be tuned more independently. In the three-term objective, $\phi_1 = \alpha$, $\phi_2 = \alpha + \nu$, and $\phi_3 = \alpha + \lambda$. Adding β adjusts ϕ_1 without changing ϕ_2, while adding η adjusts ϕ_3 without changing λ.

The two corrections have different roles from the original regularizers. *Within-option consistency versus responsibility-based attribution.* The negative option-entropy term weighted by β acts within each option: it partially offsets the option entropy induced by mixture entropy, making the option's action distribution narrower and more consistent at the same input. This predictability can facilitate transfer because a downstream controller must compose fixed options based on their expected behavioral consequences. Excessive concentration, however, may reduce behavioral coverage. The term does not by itself make different options distinguishable, since all options may still concentrate on the same action. Responsibility regularization instead acts across options by favoring actions attributable to one option rather than several, but does not determine how stochastic an option is when selected alone. Hence ϕ_1 controls within-option action concentration, whereas ϕ_2 controls action-to-option attribution.

Controller decisiveness versus sticky retention. The controller temperature and sticky-prior weight play different roles, so interpreting both as persistence is imprecise. The negative controller-entropy term weighted by η partially offsets the controller entropy induced by mixture entropy and the sticky-prior KL, thereby lowering ϕ_3. A small ϕ_3 concentrates the controller on the highest-scoring choice, whereas a large one spreads probability across options. Being decisive is not the same as staying: a decisive controller may still make reliable switches, producing a repeatable sequence such as $k_1 \to k_2 \to k_3$. The sticky-prior weight λ, by contrast, favors retaining k^- and therefore promotes temporal continuity. Concretely, the sticky prior adds a stay margin $\lambda\Omega_{k^-}^{\mathrm{stick}}$ to the previous option's score; changing ϕ_3 sharpens or softens the resulting decision but does not create that margin.

Sampled approximation. The exact option target in Eq. 23 is a density over continuous actions. The practical update replaces this density by sampled actions. As above, the appendix suppresses the fixed goal argument g; in the implementation, the sampled quantities are evaluated at (x, g).

Sampled option update. The practical algorithm maintains two critics Q_{ξ_1}, Q_{ξ_2}, the option policies $\pi_{\theta,k}$, and the controller w_ψ, together with target copies denoted by bars. Thus $\bar{Q}_i$ are the target critics, $\bar{\pi}_k$ are the target options, $\bar{w}$ is the target controller, and $\bar{\pi}_m$ is the corresponding target mixture policy. Let

$$\bar{\rho}(k|x,a) = \frac{\bar{w}(k|x)\bar{\pi}_k(a|x_P)}{\bar{\pi}_m(a|x)}$$

be the responsibility computed under the target networks. Concretely, $\bar{\rho}$ is computed at the start of each policy-improvement step and held fixed while the sampled option and controller targets are formed. The reparameterized and projection option updates below, as well as the sampled controller target, instantiate fixed Q and ρ with $\min_i \bar{Q}_i$ and $\bar{\rho}$. All target-network parameters are held fixed during these updates. Using the minimum reduces overestimation bias (Fujimoto et al., 2018). The slow-moving target copies stabilize bootstrapping and provide the proposal actions used to estimate the projection and controller targets (Lillicrap et al., 2016); they are updated by Polyak averaging afterward. Using these target networks in Eq. 10 gives $\bar{y}$ by replacing $Q_\xi, \pi_\theta, w_\psi$ with $\min_i \bar{Q}_i, \bar{\pi}, \bar{w}$, respectively; the twin critics minimize $\mathcal{L}_Q = \mathbb{E}_{\mathcal{D}} \sum_{i=1}^{2} (Q_{\xi_i}(x,k,a) - \bar{y})^2$.

The exact option target in Eq. 23 is proportional to

$$\rho(k|x,a)^{\phi_2/\phi_1} \exp(Q(x,k,a)/\phi_1).$$

For each option k, both updates use the option-weighted replay distribution $\mathcal{D}_k(x) \propto \mathcal{D}(x)\bar{w}(k|x)$, implemented by normalizing the fixed target-controller probabilities over each minibatch. We consider two sampled updates toward this target.

Reparameterized update. $\mu_{\theta,k}(x_P)$ and $\sigma_{\theta,k}(x_P) > 0$ are the mean and standard deviation of the squashed-Gaussian option k. With $\epsilon^{(n)} \sim \mathcal{N}(0, I)$, its reparameterized action is

$$a_{k,\theta}^{(n)} = \tanh\Big(\mu_{\theta,k}(x_P) + \sigma_{\theta,k}(x_P) \odot \epsilon^{(n)}\Big). \tag{32}$$

Here $\pi_{\theta,k}(a|x_P)$ is the corresponding density, including the $\tanh$ change-of-variables correction. The option is updated by minimizing

$$\mathcal{L}_{\pi_k}^{\text{rep}} = -\mathbb{E}_{\mathcal{D}_k}\left[\frac{1}{N_{\text{rep}}} \sum_{n=1}^{N_{\text{rep}}} \Big(\min_i \bar{Q}_i(x,k,a_{k,\theta}^{(n)}) + \phi_2 \log \bar{\rho}(k|x,a_{k,\theta}^{(n)}) - \phi_1 \log \pi_{\theta,k}(a_{k,\theta}^{(n)}|x_P)\Big)\right]. \tag{33}$$

The critic and target-network parameters are held fixed during this update, while gradients pass through the sampled action into $\bar{Q}_i$ and $\bar{\rho}$. We use this update in our experiments because we found it more stable.

Projection update. Alternatively, the exact option target can be estimated using target-network quantities and samples from $\bar{\pi}_k(\cdot|x_P)$. With $a_k^{(n)} \sim \bar{\pi}_k(\cdot|x_P)$, the projection update assigns the importance-corrected weight

$$\exp b_k^{(n)} = \frac{\bar{\rho}(k|x,a_k^{(n)})^{\phi_2/\phi_1} \exp(\min_i \bar{Q}_i(x,k,a_k^{(n)})/\phi_1)}{\bar{\pi}_k(a_k^{(n)}|x_P)}. \tag{34}$$

Normalizing these weights over the N candidates gives

$$\omega_k^{(n)} = \frac{\exp b_k^{(n)}}{\sum_{m=1}^{N} \exp b_k^{(m)}}. \tag{35}$$

The option network is then trained by weighted maximum likelihood, using the same weights but evaluating the learned option on the option state x_P:

$$\mathcal{L}_{\pi_k}^{\text{proj}} = -\mathbb{E}_{\mathcal{D}_k}\left[\sum_{n=1}^{N} \omega_k^{(n)} \log \pi_{\theta,k}(a_k^{(n)}|x_P)\right]. \tag{36}$$

The total option-policy loss averages the selected loss over options:

$$\mathcal{L}_{\text{opt}} = \frac{1}{K}\sum_{k=1}^{K}\mathcal{L}_{\pi_k}. \tag{37}$$

Here $\mathcal{L}_{\pi_k}$ denotes either $\mathcal{L}^{\text{rep}}_{\pi_k}$ or $\mathcal{L}^{\text{proj}}_{\pi_k}$. For the same fixed target critics and target responsibility, both updates have the target-network instantiation of Eq. 23 as their unrestricted optimum. Within the parametric policy class, the reparameterized update directly optimizes the option-local objective, whereas the projection update fits the target by weighted likelihood. We use $N = 16$ when evaluating the projection update.

Sampled controller score. For either option update, the importance weights above also estimate the score used by the controller. To see this, define the normalizer of the exact option target:

$$F_k(x) := \int da\, \rho(k|x,a)^{\phi_2/\phi_1}\exp(Q(x,k,a)/\phi_1). \tag{38}$$

Then Eq. 23 can be written as

$$\pi^\star_k(a|x) = \frac{1}{F_k(x)}\rho(k|x,a)^{\phi_2/\phi_1}\exp(Q(x,k,a)/\phi_1).$$

Taking logs and rearranging gives

$$Q(x,k,a) + \phi_2\log\rho(k|x,a) = \phi_1\log\pi^\star_k(a|x) + \phi_1\log F_k(x).$$

Substituting this identity into Eq. 22 with $\pi_k = \pi^\star_k$ gives

$$\begin{aligned} M_k(x) &= \phi_1\mathcal{H}(\pi^\star_k(\cdot|x)) + \int da\,\pi^\star_k(a|x)\left[\phi_1\log\pi^\star_k(a|x) + \phi_1\log F_k(x)\right] \\ &= \phi_1\mathcal{H}(\pi^\star_k(\cdot|x)) - \phi_1\mathcal{H}(\pi^\star_k(\cdot|x)) + \phi_1\log F_k(x) = \phi_1\log F_k(x). \end{aligned}$$

The second line uses $\int da\,\pi^\star_k\log\pi^\star_k = -\mathcal{H}(\pi^\star_k)$, and the fact that $F_k(x)$ does not depend on a. Thus estimating $F_k(x)$ gives the option score $M_k(x) = \phi_1\log F_k(x)$. This is the score used by the controller target, which has the form

$$w^\star(k|x) \propto \exp(M_k(x)/\phi_3)w_{\text{prior}}(k|x)^{\lambda/\phi_3}.$$

Instantiating $F_k(x)$ with $\bar\rho$ and $\min_i \bar Q_i$, $\exp b^{(n)}_k$ is its importance-corrected integrand, so we use

$$\widehat{M}_k(x) = \phi_1\log\left(\frac{1}{N}\sum_{n=1}^{N}\exp b^{(n)}_k\right). \tag{39}$$

After applying the logarithm, $\widehat{M}_k(x)$ is a consistent sampled proxy for the corresponding target-network score, but not an unbiased finite-sample estimate.

Replacing $M_k(x)$ by $\widehat{M}_k(x)$ in Eq. 27 gives the sampled controller target

$$\widehat{w}^\star(k|x) = \frac{\exp(\widehat{M}_k(x)/\phi_3)w_{\text{prior}}(k|x)^{\lambda/\phi_3}}{\sum_j\exp(\widehat{M}_j(x)/\phi_3)w_{\text{prior}}(j|x)^{\lambda/\phi_3}}. \tag{40}$$

The controller network is trained to match this sampled target by minimizing

$$\mathcal{L}_{\text{ctrl}} = \mathbb{E}_{\mathcal{D}}\left[D_{\text{KL}}(\widehat{w}^\star(\cdot|x)\|w_\psi(\cdot|x))\right]. \tag{41}$$

Target-network update. After the critic, option-policy, and controller updates, every target parameter is updated by Polyak averaging (Polyak & Juditsky, 1992):

$$\bar\vartheta \leftarrow \tau_{\text{polyak}}\bar\vartheta + (1-\tau_{\text{polyak}})\vartheta, \qquad \vartheta \in \{\xi_1,\xi_2,\psi,\theta_1,\dots,\theta_K\}. \tag{42}$$

Thus each training iteration updates the critics first, then the option policies using either Eq. 33 or Eq. 36, then the controller, and finally all target networks.

Conditional mutual-information interpretation. At a fixed option-augmented state x, the hierarchy induces the joint distribution $p(k, a|x) = w(k|x)\pi_k(a|x_P)$. Its option posterior is exactly the responsibility $p(k|x, a) = \rho(k|x, a)$. Therefore,

$$\begin{aligned} I(K; A|x) &= \mathcal{H}(w(\cdot|x)) - \mathbb{E}_{a\sim\pi_m(\cdot|x)}\left[\mathcal{H}(\rho(\cdot|x, a))\right] \\ &= \mathcal{H}(\pi_m(\cdot|x)) - \mathbb{E}_{k\sim w(\cdot|x)}\left[\mathcal{H}(\pi_k(\cdot|x_P))\right]. \end{aligned} \tag{43}$$

If all options have the same action distribution, or if the controller always selects one option, then $I(K; A|x) = 0$. Increasing this quantity instead makes the action more informative about the selected option, although positive mutual information does not guarantee that all K options are used.

The controller- and responsibility-entropy terms in Eq. 8 can consequently be written exactly as

$$\begin{aligned} &- \phi_2\mathbb{E}_{a\sim\pi_m(\cdot|x)}\left[\mathcal{H}(\rho(\cdot|x, a))\right] + \phi_3\mathcal{H}(w(\cdot|x)) \\ &\quad = \phi_2 I(K; A|x) + (\phi_3 - \phi_2)\mathcal{H}(w(\cdot|x)). \end{aligned} \tag{44}$$

A pure conditional-MI bonus ties option utilization and action-to-option attribution to the same coefficient. TASRO instead controls attribution with ϕ_2 and controller softness with ϕ_3, while ϕ_1 separately controls within-option action softness. Conditional MI also neither determines whether actions are reward-relevant nor imposes the input restriction x_P: these roles are supplied by Q and the information asymmetry, respectively. Likewise, it does not favor retaining k^-; that preference comes from the sticky prior. IR-VIC instead uses conditional option–action information as a bottleneck for discovering subgoals (Modhe et al., 2020). Accordingly, we use conditional MI as a comparison lens: it connects TASRO to prior information-theoretic objectives while showing how TASRO controls action-to-option attribution and controller softness separately.

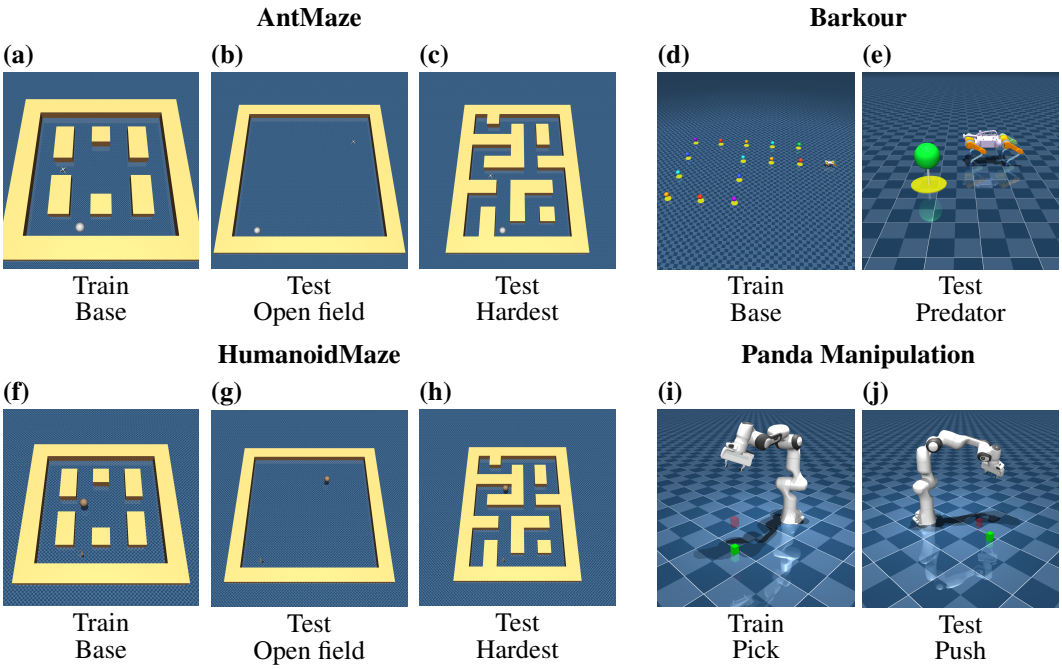


Figure 6: Source and target environments used in the transfer experiments. Within each domain, the left panel is the source training task and the remaining panels are target transfer tasks. The top row shows AntMaze and Barkour; the bottom row shows HumanoidMaze and Panda manipulation.

A.3 Environments and Transfer Protocol

All experiments use the same frozen-option transfer protocol. During training-task learning, the option policies $\pi_{\theta,k}$, controller w_ψ, and critics Q_{ξ_i} are trained jointly on a single training task. During downstream transfer, we freeze the learned option library and discard the training-task controller and critics. We then train a new downstream controller, critic, and temperature parameter on each downstream task. The downstream controller and critic receive the option-augmented state and goal (x, g), while each frozen option receives only the masked option input x_P. Thus options never receive the goal directly, and under the masking used in this paper they also do not observe the task-defining achieved-goal coordinates. The four domains below instantiate this protocol in navigation, locomotion, and manipulation.

A.3.1 AntMaze navigation

AntMaze (Bortkiewicz et al., 2025) is a quadruped navigation domain. The source task is simple-maze navigation, and the transfer tasks are open-field and hardest-maze goal reaching (Fig. 6a–c). The environment state s is 29-dimensional, and the goal g is the target xy position in world coordinates.

The controller observes the full augmented state $x = (s, k^-)$. Options act on $x_P = (s_P, k^-)$, where s_P is the 27-dimensional proprioceptive state with global xy position removed. For transfer, each frozen option library is reused by training a new controller and critic on each downstream layout.

Source and transfer episodes are both 1000 steps. The downstream controller selects an option every $H = 10$ primitive steps, long enough to produce measurable displacement but short enough for frequent route corrections. The dense reward is $20\Delta d + 10\mathbf{1}_{\text{success}} - 0.5\|a\|^2$, where Δd is progress toward the xy goal. Success is declared when the xy distance to the goal is below 0.5 meters. Episodes terminate on success or loss of balance, defined by torso height outside $[0.2, 1.0]$ or a flipped torso orientation.

A.3.2 HUMANOIDMAZE NAVIGATION

HumanoidMaze (Bortkiewicz et al., 2025) follows the same frozen-transfer protocol as AntMaze, but with higher-dimensional observations and less stable humanoid dynamics. The source task is simple-maze navigation (Fig. 6f), and the transfer tasks are open-field and hardest-maze goal reaching (Fig. 6g,h). The environment state s is 51-dimensional, consisting of 26 generalized coordinates and 25 generalized velocities; the extra position coordinate comes from the floating-base quaternion. The goal g is a target xyz position in world coordinates with fixed height $z = 1.25$.

The controller observes the full augmented state $x = (s, k^-)$. Options act on $x_P = (s_P, k^-)$, where s_P is the 49-dimensional proprioceptive state with global xy position removed. For transfer, each frozen option library is reused by training a new controller and critic on each downstream layout.

Source and transfer episodes are both 1000 steps. The downstream controller selects an option every $H = 10$ primitive steps. The dense reward is $300\Delta d + 10\mathbf{1}_{\text{success}} - 0.1\|a\|^2$, where Δd is progress toward the 3D target. Success is declared when the 3D target distance is below 0.5 meters. Episodes terminate on success or loss of balance, defined by torso height outside $[1.0, 2.0]$.

A.3.3 BARKOUR WAYPOINT LOCOMOTION

Barkour is a quadruped waypoint-navigation domain based on the Google Barkour model (Zakka et al., 2022; Caluwaerts et al., 2023). The source task uses a fixed base route, and the transfer task uses the predator waypoint layout (Fig. 6d–e). The environment state s is 48-dimensional, the goal g is the current waypoint xy position in world coordinates, and the action is a 12-dimensional joint-target command.

In the source task, the robot follows a fixed 14-waypoint route (Fig. 6d). In the predator transfer task, there is no fixed route. Whenever the robot reaches the current waypoint, defined as entering a 0.15-meter radius, a new waypoint is sampled relative to the robot's current pose. In our transfer configuration, the new waypoint is sampled uniformly at radius 2 meters, with direction drawn from a $\pm 170^\circ$ cone around the robot's forward yaw. This allows almost all headings except a narrow rear sector. The source route is a simplified waypoint-only mimic of the original Barkour agility setting (Caluwaerts et al., 2023): it keeps the Barkour robot, but removes the richer terrain, obstacle-course structure, and external kick disturbances.

The controller observes the full augmented state $x = (s, k^-)$ and the current waypoint goal. Options act on $x_P = (s_P, k^-)$, where s_P removes the robot's global xy position, leaving a 46-dimensional locomotion state. This option state contains global yaw, local velocity, angular rates, projected gravity, joint angles, joint velocities, and the previous action.

Source episodes are 1000 steps, and predator transfer episodes are 4000 steps. The longer transfer horizon accommodates the 10-waypoint random sequence. During downstream transfer, each frozen option library is reused by training a new controller and critic on the predator layout. The downstream controller selects an option every $H = 10$ primitive steps. The waypoint reward uses weights 10.0 for clipped progress, 0.2 for heading alignment, 20.0 for intermediate waypoint reaches, 100.0 for final success, 100.0 for proximity shaping, and -1.0 for falling; velocity-alignment, upright, action-magnitude, and action-rate terms are disabled in this setting. Episodes terminate on success after reaching 10 predator waypoints or on physical failure, defined by the torso flipping, any joint leaving its valid range, or torso height falling below 0.18 meters. This transfer task exposes waypoint directions much broader than the fixed source route.

A.3.4 PANDA MANIPULATION TRANSFER

Panda manipulation tests frozen-option transfer between two arm-manipulation objectives. The source task is single-phase pick-and-orient, and the transfer task is table pushing (Fig. 6i–j). Both tasks use the same 47-dimensional environment state s and an 8-dimensional accumulated delta joint-target command: 7 arm joints and 1 gripper command. The policy outputs a joint-target delta, and the environment integrates these deltas into the commanded joint targets. The state contains cube position (3), cube orientation as a 6D rotation representation (6), robot joint positions (9), robot joint velocities (9), gripper position (3), gripper orientation as a 6D rotation representation (6), cube–gripper relative position (3), and control residuals (8), for a total of 47 dimensions.

In the source pick-and-orient task, the robot must grasp a cube from the table and match a target pose in free space. The goal g is 9-dimensional: target cube position (3) and target cube orientation in a 6D rotation representation (6). In the transfer task table pushing, the robot must push the cube to a target position on the table. The goal g is the 3-dimensional target cube position. Thus the full pick observation has dimension $47+9=56$, while the full push observation has dimension $47+3=50$. The goal dimensionality differs across tasks, but the frozen options remain unchanged because they observe neither goal.

The controller observes the full augmented state $x = (s, k^-)$ and the task goal. Options act on $x_P = (s_P, k^-)$, where s_P is the same 26-dimensional proprio-control state in both tasks: 9 robot joint positions, 9 robot joint velocities, and 8 control residual features. The control residual is the difference between the commanded joint targets and the current robot joint positions. Thus the options do not see the cube pose, gripper pose, cube–gripper relative position, or goal. This identical option observation across source and transfer tasks enables frozen reuse.

Source episodes are 200 steps, and push transfer episodes are 300 steps. During downstream transfer, each frozen option library is reused by training a new controller and critic on table pushing. The downstream controller selects an option every $H = 5$ primitive steps, shorter than navigation horizons because manipulation episodes are shorter and require faster contact-level corrections. The pick reward weights are 4.0 for gripper-to-cube approach, 8.0 for cube-to-target pose matching, 0.25 for avoiding floor collision, and 0.3 for mild joint regularization. The push reward weights are 4.0 for gripper-to-cube approach, 8.0 for cube progress toward the target, and 0.25 for avoiding floor collision. Source success is declared when $\|p_{\text{box}} - p_{\text{target}}\| + 0.25\|R^{6D}_{\text{box}} - R^{6D}_{\text{target}}\| < 0.05$. Push success is declared when the cube xy distance to the target is below 0.05 meters. Success does not terminate an episode; episodes terminate only on workspace exit or simulator instability.

Existing assets and licenses. Our simulated robot assets are based on MuJoCo Menagerie (Zakka et al., 2022). In particular, the Franka Emika Panda model and the Google Barkour v0 model are distributed under the Apache License 2.0 in MuJoCo Menagerie. The AntMaze and HumanoidMaze environments are based on the JaxGCRL maze environments (Bortkiewicz et al., 2025), which are also distributed under the Apache License 2.0. We cite the Barkour benchmark setup (Caluwaerts et al., 2023); all third-party assets are used only for simulated control experiments.

A.4 Training and hyperparameter details

A.4.1 Training-Task Learning

Tables 1 and 2 summarize the TASRO training-task settings and objective coefficients for each domain. Here, $\dim(s_P)$ and $\dim(s)$ denote the masked option-state and full environment-state dimensions, respectively; both exclude k^-, and the goal is supplied separately to the controller.

Table 1: TASRO training-task settings by domain family.

Domain	Ep. len	$\dim(s_P)$	$\dim(s)$	K	γ	Steps	Batch	$\text{lr}_{Q/\pi/w}$
AntMaze	1000	27	29	4	0.99	100M	512	3×10^{-4}
HumanoidMaze	1000	49	51	4	0.99	200M	512	10^{-3}
Barkour	1000	46	48	3	0.99	200M	256	10^{-3}
Panda pick	200	26	47	16	0.97	350M	256	10^{-3}

Table 2: Effective TASRO objective coefficients used in training-task learning; see Eq. 31.

Domain	ϕ_1	ϕ_2	ϕ_3	λ	Ω_{stick}
AntMaze	0.5	0.7	0.6	0.4	2.0
HumanoidMaze	0.5	0.7	0.6	0.4	6.0
Barkour	0.3	0.7	0.6	0.3	4.0
Panda pick	0.2	0.7	0.5	0.3	4.0

Training uses the replay-based actor-critic implementation with a minimum replay fill of 100k transitions. The maximum replay size is 10M for AntMaze and Panda, 5M for HumanoidMaze,

and 20M for Barkour. Soft target updates use Polyak averaging with target-retention coefficient $\tau_{\text{polyak}} = 0.995$ (Polyak & Juditsky, 1992).

A.4.2 Downstream transfer training

In frozen-option transfer, the option library learned on the training task is kept fixed. For each downstream task, we discard the training-task controller and critic and train a new controller, critic, and temperature parameter. Within each domain, all frozen option libraries are evaluated with the same downstream training setup, so performance differences reflect how reusable the frozen options are.

The downstream training settings are summarized in Tab. 3. In all rows, the new controller and critic observe the option-augmented state and goal (x, g), while each frozen option keeps the same option input used during training-task learning. The downstream controller is an MLP with two 256-unit hidden layers. All downstream runs use batch size 256, critic learning rate 3×10^{-4}, and target-retention coefficient 0.995. The soft-Q temperature is initialized at $\tau_{\text{sql}} = 1.0$ and tuned automatically during downstream training.

Table 3: Downstream frozen-option transfer settings. The option library is fixed; only a new controller, critic, and temperature parameter are trained.

Domain	Downstream task	Steps	Ep. len	H
AntMaze	Open field	200k	1000	10
AntMaze	Hardest maze	2M	1000	10
HumanoidMaze	Open field	1M	1000	10
HumanoidMaze	Hardest maze	2M	1000	10
Barkour	Predator	20M	4000	10
Panda	Push	25M	300	5

A.5 Baseline Training-Task Settings

For baseline comparisons, we match the number of options K to the TASRO setting in each domain and evaluate all frozen option libraries with the same domain-specific transfer learner described above. Among the compared methods, only TASRO conditions the low-level options on the previous option index k^-; the baseline option policies follow their original interfaces.

Implementation and configuration protocol. To ensure a faithful comparison, we initialized each baseline from the objective, architecture, and hyperparameters reported in its original paper, and consulted released implementations when available. We preserved each method's objective, introducing only the adaptations required for continuous control, the common observation interface, and frozen-library evaluation. Because the methods depend on different hyperparameters, we tuned the quantities most relevant to each objective rather than applying a single shared search grid. The principal search ranges are reported below, and the complete configurations are included in the supplementary material.

Configurations and budgets were selected using only training-task evidence: task return and success for reward-trained methods, and intrinsic-objective and forced-skill diagnostics for DIAYN and METRA. We froze the final checkpoint at each stated budget; downstream scores were never used for selection. Hyperparameters were fixed across five seeds, and the frozen libraries were evaluated with the same downstream learner within each domain, so the downstream tasks serve as held-out tests of option reuse.

Option-Critic (OC), deep off-policy variant (Bacon et al., 2017). We began from the published Option-Critic objective and its reported settings for option selection, entropy regularization, and termination learning. The implementation retains learned intra-option policies and terminations and ϵ-greedy selection over state–option values. The original paper does not provide a deep Gaussian implementation for our continuous-action domains, so we developed a continuous-control variant with squashed-Gaussian intra-option policies, replay, and soft target networks. We tuned its network capacity, learning rates, regularization, exploration, and update schedule on training-task re-

Table 4: Training-task budgets for the five-seed experiments, in environment steps. Reward-trained methods receive the same full budget within each domain. DIAYN and METRA optimize intrinsic objectives and use the budgets selected from their intrinsic and forced-skill diagnostics. A dash means that the method is not included in that domain.

Method	AntMaze	HumanoidMaze	Barkour	Panda
OC	100M	200M	200M	–
PPOC	100M	200M	200M	–
PPOC*	100M	200M	200M	350M
DIAYN	50M	50M	50M	50M
METRA	50M	50M	50M	50M
MSOL	150M	200M	200M	350M

turn and success. The search compared two to four 256-unit hidden layers, minibatches of 128 and 256, option-policy learning rates $\{2, 3\} \times 10^{-4}$, critic learning rates from 2×10^{-4} to 10^{-3}, and termination learning rates from 5×10^{-5} to 10^{-3}. Entropy coefficients ranged from 0 to 0.2, termination margins from 0 to 1, final ϵ values from 0 to 0.1, target-retention coefficients were 0.999 or 0.995, and update multipliers ranged from 1 to 100. The selected configurations use three 256-unit layers, learning rates 2×10^{-4} for the option policies and critic and 5×10^{-5} for termination, target-retention coefficient 0.995, minibatch size 256, update multiplier 10, entropy coefficient 0.01, termination margin 0.01, and an ϵ-greedy schedule from 1.0 to 0.05. We label this method as a deep off-policy variant rather than presenting it as an unaltered implementation of the original online algorithm. Its low-level options receive the full state and goal, including current position.

PPOC and PPOC* (Klissarov et al., 2017). We began from the paper's continuous-control recipe: its PPO optimization and advantage-estimation settings, two-layer 64-unit tanh networks, state-independent action log-standard deviations, stochastic option selection, learned terminations, and deliberation cost. We retained the published PPOC objective and call-and-return structure while tuning network capacity, optimization, deliberation cost, and training horizon for our domains. Starting from the paper's two 64-unit layers, we compared three 256-unit layers and six 512-unit layers. We also compared 1, 4, and 10 PPO epochs, entropy coefficients 0 and 0.01, and deliberation costs 0, 0.01, 0.1, and 1. The learning rate 3×10^{-4}, clipping threshold 0.2, $\lambda_{\mathrm{GAE}} = 0.95$, and 10 minibatches per update were held fixed. The selected configurations use three 256-unit layers, except for six 512-unit layers in HumanoidMaze, together with 10 PPO epochs, 0.01 entropy coefficient, and 0.01 deliberation cost.

PPOC and PPOC* use the same learner, hyperparameters, and budget within each domain; they differ only in the option observation. PPOC's low-level options receive the full state and goal, whereas PPOC* exposes them only to task-agnostic proprioception s_P. Their comparison therefore isolates information masking without adding TASRO's previous-option conditioning, sticky prior, or responsibility-based specialization. OC and PPOC are not evaluated on Panda push because their goal-conditioned options are trained with a 9D pick-and-orient goal, whereas the downstream push task uses a 3D position goal.

DIAYN (Eysenbach et al., 2019). We began from DIAYN's published recipe: a uniform categorical skill prior, one skill sampled and held for each episode, the discriminator reward $\log q_\phi(z \mid s) - \log p(z)$, and SAC optimization (Haarnoja et al., 2018). The initial configuration follows the paper's SAC settings, including its increased network capacity and entropy scaling. We then used intrinsic-objective and forced-skill diagnostics to compare critic networks with two 256-unit layers and four 512-unit layers, update multipliers $\{1, 4, 16, 100, 128\}$, and target-entropy coefficients -1 and 0.5. We also compared the standard x, y discriminator input, the full state, and the masked state required by our protocol. The selected configuration uses two 256-unit layers for the policy, critic, and discriminator, learning rate 3×10^{-4} for all three, minibatch size 256, update multiplier 128, automatic entropy adjustment initialized at 0.1 with target entropy $-\dim(\mathcal{A})$, and target-retention coefficient 0.995. These adaptations preserve the DIAYN objective while matching the common frozen-library protocol.

METRA (Park et al., 2024). We began from the paper's published hyperparameters and used the authors' released implementation as an executable reference. We cross-checked the representation and dual losses, intrinsic reward, replay and update ordering, SAC optimization, and initialization. The starting configuration uses the reported learning rate 10^{-4}, minibatch size 256, two 1024-unit hidden layers, automatic entropy adjustment, initial dual value 30, and relaxation constant 10^{-3}. We compared discrete and continuous latents, latent dimensions 2 and 4 before setting the final discrete dimension to K, and networks with two 256-unit, two 1024-unit, or three 1024-unit hidden layers. Minibatch sizes were 256, 512, and 1024; policy, critic, and dual learning rates were 10^{-4} or 3×10^{-4}; and representation learning rates were 10^{-4}, 3×10^{-4}, or 10^{-3}. We also compared automatic and fixed entropy adjustment, initial temperature values from 0.01 to 1, initial dual values $\{0.01, 0.5, 1, 30\}$, representation-update counts $\{1, 10, 20, 50, 200\}$, and observation normalization on or off. The selected benchmark uses a discrete latent of size K, two 1024-unit layers, minibatch size 512, learning rate 10^{-4} for the policy, critic, representation, and dual updates, automatic entropy adjustment, initial dual value 30, and relaxation 10^{-3}. It retains METRA's metric-aware objective and SAC backbone while matching TASRO's task-agnostic low-level interface. The position-aware reference in Fig. 11 verifies that the implementation recovers METRA's characteristic directional organization.

MSOL (Igl et al., 2020). We began from MSOL's published posterior–prior factorization, KL-regularized reward, learned termination model, and annealed regularization and entropy schedules. The original experiments learn from multiple discrete-action tasks; we adapt the method to continuous control with Gaussian prior and posterior option policies and apply it to the common single training task. We retain prior distillation and learned terminations and tune the KL and entropy schedules, termination prior, network capacity, PPO updates, and training horizon. We compared shared networks with two or three 256-unit layers and four 512-unit layers, prior learning rates 10^{-5} and 10^{-4}, termination prior weights from 0.4 to 0.9, 1 to 4 PPO epochs, and minibatch sizes 256 and 1024. The KL coefficient was annealed from 0.01 to 0.1; the intra-option and termination entropy coefficients were annealed from 0.001 to 0.0003 and from 0.01 to 0.001; and the controller-entropy schedules compared starts of 0.01 and 0.015 with endpoints 0.003 and 0.005. The selected implementation uses four 512-unit shared layers followed by two 256-unit head layers, minibatch size 1024, one PPO epoch, termination prior weight 0.4, $\mathrm{lr}_{\text{posterior}} = 3 \times 10^{-4}$, $\mathrm{lr}_{\text{prior}} = 10^{-5}$, and $\gamma = 0.99$. Its controller-entropy coefficient is annealed linearly from 0.01 to 0.003 over 150M environment steps in AntMaze and HumanoidMaze and 100M steps in Barkour and Panda, and remains at 0.003 thereafter. After training, we freeze the goal-free prior option library.

Flat PPO is trained from scratch only on the downstream task and therefore has no training-task option library or entry in Table 4. Its policy and value networks have four and five 256-unit hidden layers, respectively, and are optimized with Adam at learning rate 3×10^{-4} and entropy cost 10^{-3}. It uses the same downstream budget, seeds, and evaluation protocol as the frozen-library methods.

A.6 Training-task learning curves

To contextualize frozen transfer, we report training-task learning curves. The figures include the reward-trained option methods, for which success and return have a common interpretation within each domain. Figure 7 shows success across all domains, while Fig. 8 provides a more graded view of learning progress through episode return in HumanoidMaze and Panda. Comparing these curves with downstream performance distinguishes differences that emerge during joint training from those that emerge when the frozen libraries are recomposed downstream. DIAYN and METRA are omitted because they optimize intrinsic objectives, and Flat PPO because it is trained only downstream; Panda includes only the option methods evaluated in downstream transfer.

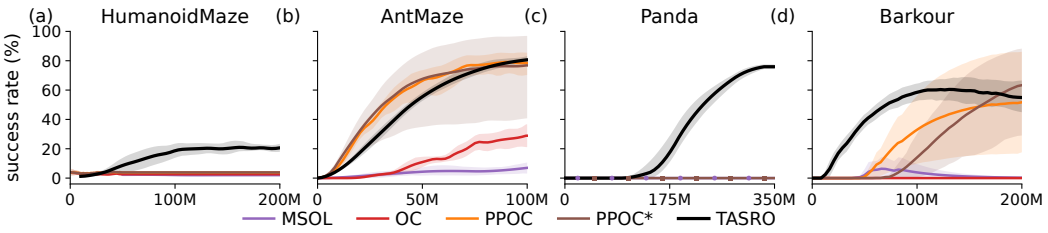


Figure 7: Training-task learning curves for reward-trained option methods.

In AntMaze, TASRO, PPOC, and PPOC* all attain high training-task success (Fig. 7b), yet their downstream ordering differs (Fig. 2c,d). PPOC is competitive during training but performs worse than TASRO and PPOC* after the options are frozen. TASRO has the highest mean in the open field, whereas PPOC* has the highest mean in the hardest layout. Thus, solving the training layout does not ensure that the learned options can be effectively recomposed downstream.

Success is binary and can hide partial learning. Figure 8 shows that in HumanoidMaze, OC, PPOC, and PPOC* move toward the target but not far enough to reach it; in Panda, PPOC* reaches the cube but plateaus before picking it up. TASRO continues to higher returns and is the only method to consistently complete either task, so its downstream advantage in these domains already emerges during training-task learning.

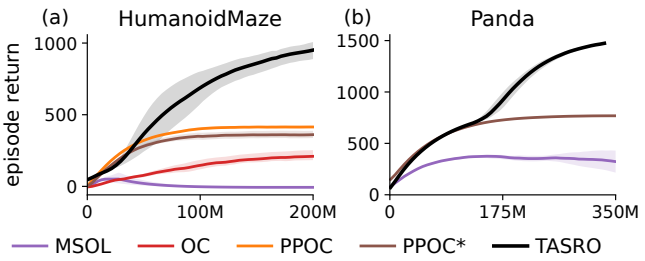


Figure 8: Training-task episode returns in HumanoidMaze and Panda; aggregation and shading match Fig. 7.

Barkour shows the opposite pattern. TASRO rapidly learns the fixed training route and maintains substantial success, yet performs worse than PPOC* on the predator downstream task. Its downstream weakness therefore does not result from failure to learn the training route. PPOC* attains both the strongest late-training performance and the strongest downstream result. Both methods learn $K = 3$ option libraries on the same fixed route. These results show that PPOC* learns options that are more useful for the varied headings required by the Barkour predator task (Sec. 3.2).

A.7 Scaling the Barkour option library

A larger nominal option count is useful only if the additional options become distinct behaviors that the controller actually composes. Barkour tests this directly. We train TASRO and PPOC* with $K \in \{3, 8\}$ under the same 200M-environment-step budget on the fixed waypoint-navigation training task, then ask (i) how many options the learned controller relies on and (ii) how distinct the individual options are when forced. Both measures show that TASRO's repertoire grows with K, whereas PPOC*'s does not. We then test whether this increased library capacity changes the original predator-transfer ordering.

We measure controller reliance using the effective number of options, $\exp(\mathcal{H}(p(k)))$, the perplexity of the option-occupancy distribution. This quantity has the intuitive interpretation of how many options the controller effectively uses.

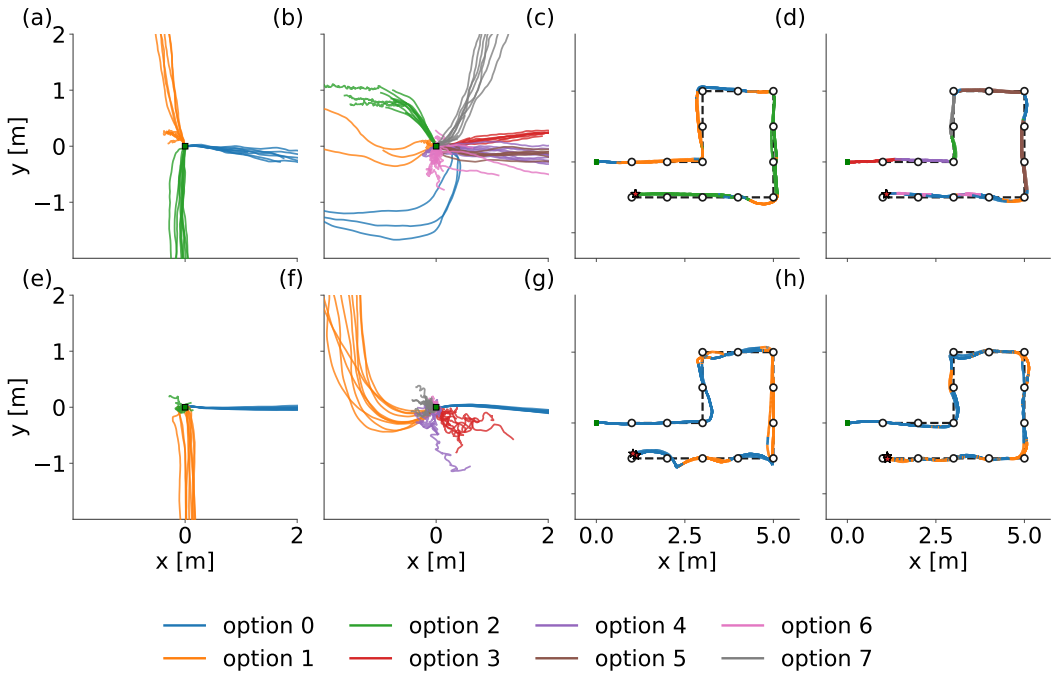


Figure 9: Barkour option behavior and training-task composition for representative libraries. The top row shows TASRO and the bottom row PPOC*. Panels (a,e) and (b,f) force the $K = 3$ and $K = 8$ options, respectively. Panels (c,g) and (d,h) show training-task composition at $K = 3$ and $K = 8$. Segments are colored by the executed option. The remaining markers show the route, waypoints, start, final waypoint, and successful termination.

Training task controller composition. Representative $K = 3$ and $K = 8$ libraries are composed over ten successful deterministic training-task episodes for TASRO at $K = 3$ (Fig. 9c) and $K = 8$ (Fig. 9d), and for PPOC* at $K = 3$ (Fig. 9g) and $K = 8$ (Fig. 9h). Option indices are local categorical labels and need not correspond across methods or values of K. Table 5 evaluates five independently trained seeds per method and K, with ten deterministic episodes per seed. A successful seed completes the fixed route in all ten episodes. For each successful seed, we combine option counts across episodes to compute the effective number of options and count those with nonzero occupancy, then report mean $\pm$ standard deviation across successful seeds.

Table 5: Barkour training-task library scaling.

Method	K	Successful seeds	Effective options	Options used
TASRO	3	5/5	2.74 ± 0.10	3.00 ± 0.00 / 3
TASRO	8	5/5	6.84 ± 0.48	7.75 ± 0.43 / 8
PPOC*	3	4/5	1.85 ± 0.12	2.00 ± 0.00 / 3
PPOC*	8	3/5	1.91 ± 0.04	2.00 ± 0.00 / 8

For TASRO, the effective number of options rises from 2.74 ± 0.10 to 6.84 ± 0.48 as K increases from 3 to 8, with occupancy spread broadly across the $K = 8$ library rather than concentrated on one option (Table 5 and Fig. 9d). PPOC* remains near two (1.85 ± 0.12 and 1.91 ± 0.04), and every successful controller uses exactly two options. At $K = 8$, those two options account for 65% and 35% of executed steps, while the remaining six are never selected.

Forced-option behavior. Forcing each option provides a controller-independent view and gives the same answer. The forced-option panels show ten stochastic rollouts per option in a $\pm$2m window around the common start, and trajectories may continue beyond this window (Fig. 9a,b,e,f).

TASRO's $K = 8$ library forms a graded, directionally diverse repertoire, with seven options moving the robot between 4.4m and 14.9m on average (Fig. 9b). PPOC* instead has two long-range options, while the rest remain near the start (Fig. 9f). Displacement is a coarse proxy because a useful option could instead turn or stabilize the robot, but here the low-displacement behaviors coincide with options that the controller does not select. Thus, the added capacity becomes usable, distinct behavior for TASRO, while PPOC* continues to solve the route with an effectively two-option library. All five TASRO seeds complete all ten evaluation episodes at both values of K, compared with four of five PPOC* seeds at $K = 3$ and three of five at $K = 8$.

Predator transfer at $K = 8$. We freeze each $K = 8$ library and train a new downstream controller for 20M environment steps. We use $H = 10$ as the primary setting and repeat the experiment at $H = 5$, with five library/controller seed pairs per setting.

Table 6: Barkour predator transfer with frozen $K = 8$ libraries. Final success is evaluated over 50 deterministic episodes per seed, with 95% bootstrap confidence intervals over five seeds. "Waypoints" is the mean number reached out of ten.

Method	H	Final success, % (95% CI)	Waypoints
TASRO	10	22.0 [3.2, 52.4]	4.44
PPOC*	10	47.2 [21.6, 72.8]	6.24
TASRO	5	42.0 [29.2, 62.4]	6.52
PPOC*	5	59.6 [32.0, 87.2]	7.44

Table 6 shows that PPOC* retains the higher mean predator-transfer success at both horizons: 47.2% versus 22.0% at $H = 10$, and 59.6% versus 42.0% at $H = 5$. Waypoint progress follows the same ordering, although the confidence intervals overlap. This occurs despite TASRO assigning nontrivial occupancy to essentially all eight options, whereas PPOC* concentrates 78% and 87% of executed steps on its top two options at $H = 10$ and $H = 5$, respectively. Thus, repertoire breadth does not by itself ensure downstream utility, and $K = 3$ alone does not explain the original Barkour ordering.

A.8 AntMaze option use and temporal composition

We next examine how the newly trained downstream controller uses each frozen AntMaze library in $K = 4$ open-field transfer across five seeds per method (Fig. 10). All methods use the same option horizon, $H = 10$ primitive steps. Option occupancy and stay rate aggregate 100 deterministic-policy episodes per seed (Fig. 10a,c), while library utilization uses the executed-option fractions logged during downstream controller training (Fig. 10b).

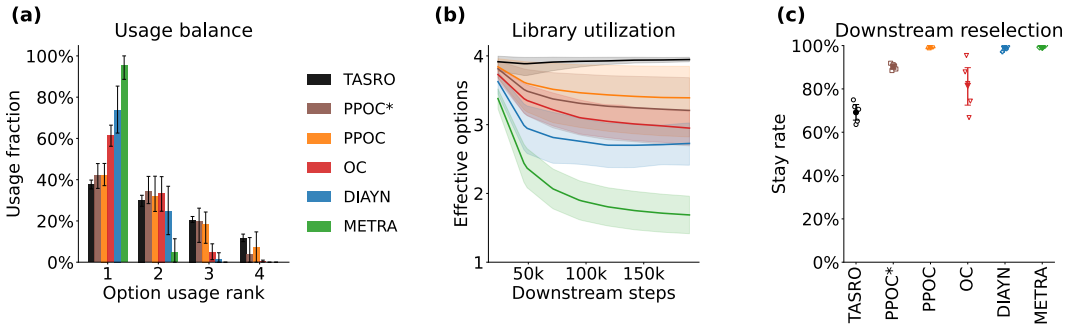


Figure 10: Option use and temporal composition in AntMaze open-field transfer. (a) Ranked option occupancy under deterministic evaluation; ranks order the four options from most to least used within each seed. (b) The effective number of executed options, $\exp(\mathcal{H}(p(k)))$, during downstream controller training. (c) Decision-level stay rate, $\Pr(k_t = k_{t-1})$, measured directly from option transitions. Means and 95% bootstrap confidence intervals are computed over five seeds.

Broad library use. TASRO is the only method for which every option receives substantial occupancy under deterministic evaluation: its least-used option still accounts for 11.7% of primitive steps, while its most-used option accounts for 37.7% (Fig. 10a). This balance persists during downstream learning. TASRO finishes with 3.95 effective options out of four (95% CI $[3.91, 3.98]$), compared with 3.21 for PPOC*, 3.39 for PPOC, 2.95 for OC, 2.73 for DIAYN, and 1.69 for METRA (Fig. 10b). Thus, the downstream controller uses the TASRO library broadly rather than relying on one dominant option.

Persistent option use without controller collapse. TASRO keeps an option active for 30.5 primitive steps on average, approximately three option intervals, and retains the same option at 69.2% of decision boundaries (Fig. 10c). It therefore persists beyond the mandatory horizon while remaining able to switch within an episode. By contrast, PPOC, DIAYN, and METRA have stay rates of 99.6%, 98.9%, and 99.7%, respectively, together with mean dwell times between 560 and 817 primitive steps. Combined with their lower library utilization, these values indicate near-absorbing controller behavior rather than more effective temporal composition. We measure persistence by counting how often the controller reselects the same option at consecutive decision points, rather than using controller entropy as a proxy (Fig. 10c). Taken together with the forced-option trajectories in Fig. 3b, these diagnostics show that TASRO combines a broadly used option library with temporally extended but responsive downstream composition.

A.9 Sensitivity to the transfer option horizon

The transfer option horizon H is the number of primitive steps executed before the downstream controller may choose again. A short horizon permits frequent corrections, whereas a long horizon gives an option more time to produce a coherent behavior but delays the next opportunity to react. We repeat AntMaze open-field and Panda push transfer with $H \in \{1, 5, 10, 15\}$, changing only this horizon. Tables 7 and 8 report success over five seeds. Values are mean success with 95% bootstrap confidence intervals. The default $H = 10$ AntMaze and $H = 5$ Panda runs are reused from the main experiment, while the remaining runs use identical settings except for H. Bold marks the highest mean in each column, including ties. PPOC and OC are omitted from Panda because their goal-conditioned options are incompatible with the changed goal parameterization.

Table 7: AntMaze open-field success across transfer option horizons.

Method	$H = 1$	$H = 5$	$H = 10$	$H = 15$
TASRO	**76.8 ± 20.4**	**87.2 ± 9.4**	**88.5 ± 3.7**	**88.8 ± 5.8**
PPOC*	60.0 ± 26.4	86.8 ± 15.6	85.3 ± 15.6	**88.8 ± 13.6**
PPOC	22.0 ± 16.4	59.6 ± 16.8	66.6 ± 14.2	64.8 ± 15.0
OC	1.2 ± 1.8	15.2 ± 10.6	13.6 ± 8.1	12.4 ± 8.8
MSOL	1.6 ± 2.4	4.4 ± 2.2	7.1 ± 1.9	5.2 ± 3.0
METRA	0.8 ± 0.8	0.0 ± 0.0	0.7 ± 0.1	0.0 ± 0.0
DIAYN	0.0 ± 0.0	0.0 ± 0.0	0.7 ± 0.0	0.0 ± 0.0

Table 8: Panda push success across transfer option horizons.

Method	$H = 1$	$H = 5$	$H = 10$	$H = 15$
TASRO	**27.6 ± 8.8**	**40.0 ± 9.6**	**28.4 ± 9.6**	**15.6 ± 5.0**
PPOC*	0.0 ± 0.0	0.8 ± 0.5	1.2 ± 1.4	0.8 ± 0.8
MSOL	0.0 ± 0.0	0.9 ± 0.3	1.2 ± 1.4	1.6 ± 0.6
METRA	0.4 ± 0.6	3.5 ± 0.6	5.6 ± 1.2	6.0 ± 1.8
DIAYN	4.8 ± 3.2	4.1 ± 1.3	4.8 ± 2.4	6.4 ± 2.4

In AntMaze, TASRO is insensitive to the precise horizon once an option is given at least five steps: success is 87.2%, 88.5%, and 88.8% for $H = 5$, 10, and 15, respectively. It also retains the highest mean at $H = 1$, where the controller may reconsider its choice after every primitive step. The option-use results above make this stability intuitive. The horizon sets the earliest time at which the controller can change its choice, not the actual duration of the resulting behavior, because the controller may select the same option again. At the paper setting $H = 10$, TASRO reselects the same

option at 69.2% of decision boundaries (Fig. 10c), while each option remains active for 30.5 primitive steps on average. The controller therefore learns an effective temporal scale longer than the fixed horizon while retaining the ability to switch when the route changes. PPOC* is also competitive for $H \geq 5$, but with wider confidence intervals. The remaining methods perform substantially worse throughout the sweep. Panda exhibits a shorter, domain-appropriate timescale. TASRO performs best at $H = 5$ and declines for longer horizons, consistent with contact-rich manipulation requiring faster feedback. Importantly, it remains the only frozen library with substantial success across the sweep, so its advantage is not explained by the choice of H. Together, AntMaze and Panda show a broad navigation plateau and a shorter-horizon preference in manipulation.

A.10 AntMaze attribution with effective coefficients

Starting from the main TASRO setting $(\phi_1, \phi_2, \phi_3, \lambda) = (0.5, 0.7, 0.6, 0.4)$, we vary one effective coefficient at a time, retrain the AntMaze library, freeze the resulting options, and repeat open-field transfer with a new controller (Table 9). The remaining rows remove the structural choices. Without the sticky prior, $\lambda = 0$ and ϕ_3 also falls to 0.2. Without information asymmetry, the options receive the full $x = (s, k^-)$ instead of $x_P = (s_P, k^-)$. Without previous-option conditioning, the controller uses $w(k|s)$ and the options use $\pi_k(a|s_P)$; this also requires $\lambda = 0$ and gives $\phi_3 = 0.2$. All variants use $K = 4$, 10^8 training-task steps, 2×10^5 downstream steps, $H = 10$, and five seeds. Results are mean success with 95% bootstrap confidence intervals.

Table 9: AntMaze effective-coefficient and structural attribution.

Variant	ϕ_1	ϕ_2	ϕ_3	λ	Base maze (training task)	Open field (downstream task)
TASRO	0.5	0.7	0.6	0.4	84.0 ± 7.2	91.2 ± 4.2
Option softness						
$\phi_1 = 0.1$	0.1	0.7	0.6	0.4	45.2 ± 5.4	38.0 ± 16.8
$\phi_1 = 0.2$	0.2	0.7	0.6	0.4	62.8 ± 6.4	65.2 ± 6.6
$\phi_1 = 0.4$	0.4	0.7	0.6	0.4	78.0 ± 7.0	88.4 ± 4.8
$\phi_1 = 0.5$ (main)	0.5	0.7	0.6	0.4	84.0 ± 7.2	91.2 ± 4.2
$\phi_1 = 0.6$	0.6	0.7	0.6	0.4	$\mathbf{90.8 \pm 3.2}$	$\mathbf{95.6 \pm 2.8}$
Responsibility specialization						
$\phi_2 = 0.6$	0.5	0.6	0.6	0.4	78.0 ± 3.6	85.6 ± 4.2
$\phi_2 = 0.7$ (main)	0.5	0.7	0.6	0.4	84.0 ± 7.2	91.2 ± 4.2
$\phi_2 = 1.0$	0.5	1.0	0.6	0.4	82.0 ± 3.2	94.0 ± 4.6
Controller softness						
$\phi_3 = 0.2$	0.5	0.7	0.2	0.4	84.0 ± 6.6	88.0 ± 4.6
$\phi_3 = 0.6$ (main)	0.5	0.7	0.6	0.4	84.0 ± 7.2	91.2 ± 4.2
$\phi_3 = 1.0$	0.5	0.7	1.0	0.4	84.0 ± 2.4	87.2 ± 2.8
No sticky prior	0.5	0.7	0.2	0	80.4 ± 2.8	86.0 ± 6.0
No information asymmetry	0.5	0.7	0.6	0.4	76.4 ± 4.6	34.8 ± 4.4
No k^- conditioning	0.5	0.7	0.2	0	72.4 ± 8.0	80.8 ± 2.6

First, ϕ_1 should keep the options from becoming too sharply value-seeking. In AntMaze, increasing ϕ_1 from 0.1 to 0.6 raises training-task success from 45.2% to 90.8% and open-field transfer from 38.0% to 95.6%. Thus, within the tested range, softer option targets support both training-task learning and frozen reuse.

Second, ϕ_2 should favor actions that are more clearly attributable to each option. At $\phi_2 = 0.6$, 0.7, and 1.0, downstream success rises from 85.6% to 91.2% to 94.0%, whereas training-task success changes less consistently. This suggests that responsibility pressure can improve frozen reuse over these settings, although the confidence intervals overlap and we do not infer a general monotonic effect.

Third, ϕ_3 controls how decisively the controller selects an option, rather than directly favoring retention of the previous one. Deterministic training-task success remains 84.0% at $\phi_3 = 0.2$, 0.6, and 1.0. Under stochastic execution, success is $62.8 \pm 4.4\%$, $61.2 \pm 6.2\%$, and $65.2 \pm 8.6\%$, respectively. Downstream success also changes little, so performance is relatively insensitive to controller softness over this range.

The sticky prior instead encourages retaining k^-, while conditioning the controller and options on k^- allows their transitions to depend on the option being left. At the same $\phi_3 = 0.2$, using $\lambda = 0.4$

gives 88.0% downstream success, compared with 86.0% without the prior. With $\lambda = 0$ in both variants, removing k^- further lowers success to 80.8%. These smaller, overlapping differences suggest that the stay bias and previous-option input have complementary roles.

A complementary Barkour training-task ablation illustrates the importance of the sticky-prior strength. Increasing stickiness raises the option stay rate, but over the plotted 20M training steps, training-task performance is poor at $\Omega^{\text{stick}}_{k^-} = 0$ and 8, and highest at the intermediate setting 4 (Fig. 4a,b). Thus, both insufficient and excessive stickiness can hinder learning, so its strength must be tuned.

Finally, information asymmetry should prevent the options from using the training layout directly. Without this restriction, training-task success falls only from 84.0% to 76.4%, but downstream success falls from 91.2% to 34.8%. Thus the full-input hierarchy still learns the base maze, while its frozen options transfer substantially less well.

A.11 DIAYN AND METRA UNDER THE FROZEN-LIBRARY PROTOCOL

DIAYN (Eysenbach et al., 2019) and METRA (Park et al., 2024) learn unsupervised libraries matched to TASRO in size ($K = 4$ in the maze domains). Their skill policies observe proprioception without global position or task goals. DIAYN rewards states that identify the skill z, $r_{\text{DIAYN}} = \log q_\psi(z|s) - \log p(z)$, whereas METRA rewards representation displacement along z, $r_{\text{METRA}} = (\phi(s') - \phi(s))^\top z$. Both DIAYN's discriminator and METRA's representation ϕ still receive global x, y, preserving their spatial intrinsic-reward signals. A single skill is held fixed during each pretraining episode; the frozen policies are then composed by the common position- and goal-aware downstream controller. Their configurations were selected through the training-task sweeps detailed in Appendix A.5.

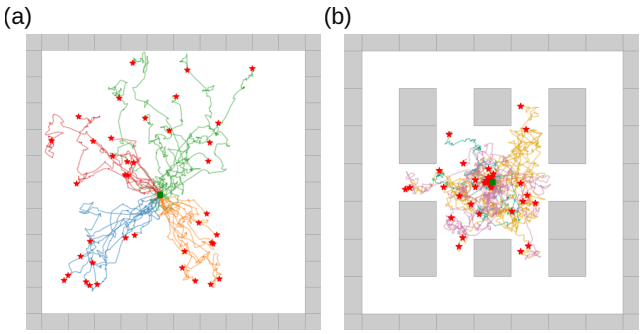


Figure 11: METRA skill organization in the reference and benchmark settings, both using the discrete $K = 4$ interface. Green squares and red stars mark starts and endpoints. (a) Reference reproduction: with global position available to the skill policy in an open field, METRA recovers four approximately orthogonal travel directions, matching Park et al. (2024). (b) Benchmark setting: global position is withheld from the skill policy but still provided to the representation ϕ, and the forced skills overlap in the base maze. All forced rollouts are shown.

METRA recovers the directional skills reported by Park et al. (2024) in the position-aware open-field reference, but its skills overlap under our maze benchmark (Fig. 11). The comparison illustrates two requirements for frozen reuse that are not captured by state-space diversity alone. Low-level behaviors must remain organized without access to global x, y, and the library must support transitions among those behaviors. These requirements interact in a maze. Consider a "go right" skill. It can work well in an open field, but eventually reaches a wall in a maze. Because the policy does not know its location and cannot switch to another skill during pretraining, it must keep pushing, stop making rightward progress, or learn a broader detour that weakens its consistent "go right" meaning. TASRO's options also lack x, y, but its position-aware controller can retain a right-moving option while it is useful and switch when a wall requires another behavior. The option can therefore preserve a consistent directional meaning without observing the maze location itself.

A.12 COMPUTE RESOURCES

As representative AntMaze runs, source training and downstream transfer were executed on a single workstation with one NVIDIA GeForce RTX 5090 GPU (34.2 GB GPU memory), 16 physical CPU cores (32 logical threads), 132.6 GB system RAM, and CUDA 13.0. Training AntMaze options for 100M environment steps used 128 parallel environments, batch size 512, 1000-step episodes, and $K = 4$ options; one run took 11,899 seconds, or about 3.31 GPU-hours. The AntMaze open-field transfer setting in Fig. 2c used 200k environment steps, 10 parallel environments, batch size 256, 1000-step episodes, and transfer option duration $H = 10$; one run took 1,357 seconds, or about 0.38 GPU-hours. The full project, including reported runs, baseline training, ablations, and preliminary sweeps not included in the paper, consumed approximately 800 single-GPU hours in total.

A.13 ADDITIONAL QUALITATIVE FORCED-OPTION PLOTS

Figure 12 shows the remaining four AntMaze open-field seeds for each method beyond the single representative seed used in the main paper. This appendix figure is intended to expose seed-to-seed variation in the qualitative option structure.

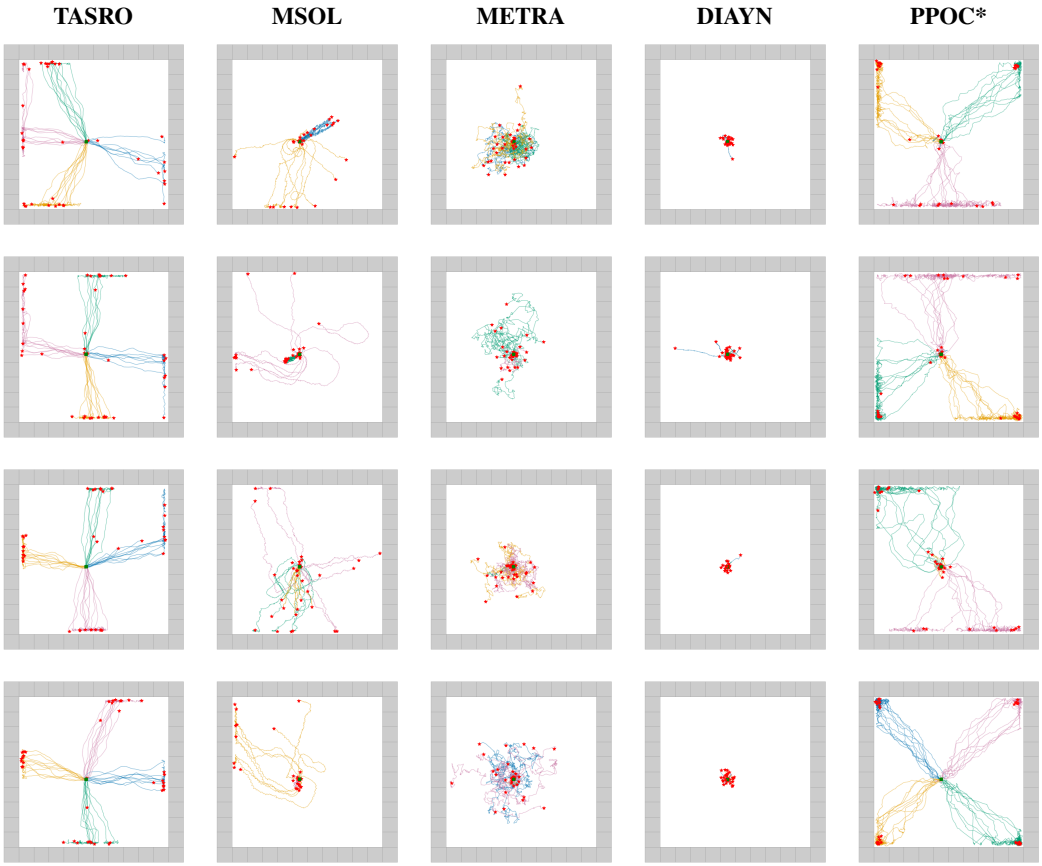


Figure 12: Additional AntMaze forced-option visualizations in the open-field transfer environment. Columns correspond to methods, and rows show the four additional seeds not included in Figure 3.