# PLANNING GUIDED OPTION CRITIC

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

A symbolic skill specification describes when a skill may run, what it must achieve, and how it is rewarded, but says nothing about how to learn the skills or how to choose among them. We introduce Planning Guided Option Critic (PGOC), which learns the skills and an option-critic controller jointly from the specifications alone, deriving masks and plan-level rewards directly from them. Preconditions and continued-need predicates restrict which options may be selected; plan-progress rewards credit the controller for satisfying remaining requirements; and each completed skill bootstraps from the controller's continuation value, crediting a skill for the opportunities it creates rather than its position in a fixed plan. Across five training seeds on Craftax, PGOC with recurrent backbones reaches the Gnomish Mines in 62.8–64.5% of frozen evaluation episodes, versus 7.7–9.3% for SCALAR and at most 4.3% for PPO. Ablations that retain the critic while varying the selection rule decompose this gain by task structure. On Craftax, where a fixed plan already suffices, prescribing that plan raises Mines success further to 69.4%: the credit assignment, not the learned ordering, drives the fixed-plan task. On randomized TreasureDash, where the optimal plan depends on the encountered state, the learned controller is decisive, scoring 22.2 against 10.9 for the prescribed order and 13.3 for uniform selection. The same option critic thus supplies a single credit assignment for both kinds of task, while the value of learning the selection rule itself depends on the task.

## 1 INTRODUCTION

A long-horizon task can be decomposed into skills, each specified by when it may start, what it must achieve, and how it is rewarded. Turning those specifications into behavior then requires learning two things at once: a control policy for each skill, and a way to choose among them. The options framework makes the distinction explicit — an option is a temporally extended behavior, and a policy over options selects which one to execute (Sutton et al., 1999; Bacon et al., 2017) — but it does not say how the two should be learned together. The two problems are usually learned separately and sequenced in one of two ways. One tradition plans the order in advance (Ahn et al., 2022; Klissarov et al., 2024; Toro Icarte et al., 2022): the selector commits to a sequence before or independently of the encountered state. The other learns an observation-conditioned selector, as in the option-critic family and in hierarchical credit-assignment methods (Bacon et al., 2017; Matthews et al., 2026; de Vries et al., 2022). This split is a trade-off, not just a design preference: a learned selector pays estimation variance on the high-level policy, while a prescribed plan pays optimality whenever the state changes the best order.

Skill-specification pipelines make this split concrete. SCALAR, the most complete such system, grounds LLM-proposed skills — with preconditions, effects, and reward functions — in learned control policies, correcting the proposed requirements from trajectory feedback (Zabounidis et al., 2026). Its execution is the planned tradition in pure form: skills run in a planner-fixed order, and returns propagate along that sequence.

The planned tradition couples a skill's credit to its position in the plan rather than to its consequence. A completed skill's update bootstraps from the prescribed next skill's value and inherits its subsequent advantage, regardless of whether that next skill's reward reflects what the completed skill actually enabled. The plan also fixes the order, so the agent cannot choose among several skills that are simultaneously eligible: in SCALAR's Gnomish Mines sequence, the agent searches for the entrance only after completing the required combat, even when exploration has already revealed both the



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.

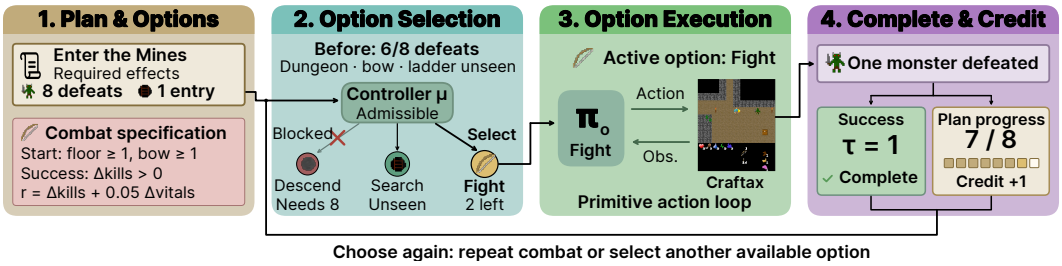


Figure 1: **Plan-guided option selection and execution.** **(1)** The plan supplies required effect counts, and option specifications define initiation, success, and skill rewards. **(2)** With six of eight required defeats credited, the learned controller selects combat among admissible options; ladder search remains available. **(3)** The policy exchanges primitive actions and observations with the environment. **(4)** One defeat completes this execution ($\tau = 1$) and earns one unit of capped plan credit; control returns for reselection. The library is illustrative; plan counts are not direct policy inputs.

entrance and useful combat opportunities (Zabounidis et al., 2026). Both problems follow from the same design — the plan, not the consequence, carries the credit.

We introduce Planning Guided Option Critic (PGOC), which learns the skills and the selector jointly from the specifications alone. The specifications' prerequisites mask inadmissible options, plan-progress targets reward the controller for satisfying remaining requirements, and each completed skill bootstraps from the controller's continuation value rather than from the next skill in a fixed plan. Because the skills and the selector train together, the selector's choices shape the credit every skill receives, and the skills' progress reshapes what the selector should choose. The learned selection pays off when the optimal plan depends on the encountered state: on Craftax, where a fixed plan already suffices, replacing the learned controller with that plan raises mean Mines completion, while on randomized TreasureDash the learned controller wins.

Our contributions are:

- Planning Guided Option Critic (PGOC), which learns a masked policy over options and the skill policies jointly from the supplied specifications, deriving both the masks and the controller's rewards from them
- A **controller-value bootstrap** that credits each completed skill with the continuation value of the state it leaves behind
- **Prerequisite and continued-need option masks** that constrain selection when admissible options exist, together with capped plan-progress rewards that give the controller credit for satisfying remaining requirements
- A five-seed study across Craftax and randomized TreasureDash showing that learned ordering pays off under state-dependent plans but not under a fixed plan, while the option critic's credit assignment holds up under both

## 2 Related Work

**Hierarchical control and skill discovery.** Options and MAXQ formalize temporal abstraction and hierarchical value decomposition (Sutton et al., 1999; Dietterich, 1999). Option-Critic learns the policy over options jointly with the option policies from the task reward alone (Bacon et al., 2017); its reported benefits were primarily in transfer, and it was not shown to clearly improve over a flat policy on the original task. FeUdal Networks and HIRO instead fix the high-level action space in advance, using goals in embedding or state space (Vezhnevets et al., 2017; Nachum et al., 2018). Skill-discovery methods learn the low-level repertoires themselves, without supplied task structure (Eysenbach et al., 2018; Sharma et al., 2019; Park et al., 2023; Bagaria et al., 2021). These approaches split on how the high-level choice is made: a fixed or planner-determined order on one side, and a learned observation-conditioned selector on the other. We show this split is a task-dependent trade-off rather than a universal hierarchy of methods (Section 4.4). SAC-X and SOL are closest to our setting: both coordinate policies trained on separately specified rewards (Riedmiller et al., 2018; Henaff et al., 2025). SOL assumes the rewards are given and learns an observation-conditioned

controller over them; PGOC additionally assumes symbolic prerequisites, which it uses to restrict which options may be selected at all.

**Symbolic guidance.** Planning operators and policy sketches supply task structure to hierarchical learners (Fikes & Nilsson, 1971; Andreas et al., 2016; Yang et al., 2018; Lyu et al., 2019; Illanes et al., 2020; Lee et al., 2022). Reward machines are the closest mechanism to our requirement trackers: they expose task-reward structure as automaton states to the value function, but the automaton's transitions are fixed (Toro Icarte et al., 2022). PGOC's credited-progress records play an analogous bookkeeping role while leaving the execution order to be learned; where reward machines fix the order in the automaton, PGOC treats the ordering itself as the variable under study. Learned option-interest functions similarly gate availability from learned values rather than supplied predicates (Khetarpal et al., 2020), and policy-gradient action masking enforces constraints at the primitive-action level (Huang & Ontañón, 2020); PGOC's masks act at the option level and are evaluated on the symbolic state rather than learned. The credit assignment we study follows the hierarchical decomposition of HBS (Matthews et al., 2026; de Vries et al., 2022), with the difference that our controller is trained jointly with the skills it sequences.

**Language-guided skills.** Language models have been used to write reward code (Wu et al., 2023; Xie et al., 2023; Ma et al., 2023; Klissarov et al., 2023), propose subgoals (Liu et al., 2024), and supply skill specifications (Cheng et al., 2026; Shek & Tokekar, 2025), with interaction used to refine the proposals (Ma et al., 2023; Wang et al., 2023; Nottingham et al., 2024; Wu et al., 2026; Li et al., 2026). These methods address where skill specifications come from. PGOC takes the specifications as given, and addresses the complementary question of how to execute them once supplied.

**Skill composition.** SayCan scores skills by language relevance and learned affordances but does not learn the selection policy (Ahn et al., 2022). MaestroMotif generates code for skill initiation, termination, and composition, with selection performed by a frozen LLM (Klissarov et al., 2024); PGOC's selection policy is learned jointly with the skills from environment interaction. SCALAR refines symbolic skill requirements from RL trajectories and is our direct point of comparison (Zabounidis et al., 2026). It differs from PGOC in three ways: i) it executes skills in a fixed planned order, whereas PGOC's controller is observation-conditioned; ii) it propagates returns across skill boundaries with a clamped next-skill bootstrap, whereas PGOC bootstraps from the controller's continuation value at completion (Section 3.3); and iii) it does not restrict option availability from the symbolic state, whereas PGOC masks inadmissible options. The comparison between the two systems therefore concerns execution order and credit assignment over identical supplied specifications.

## 3 Planning Guided Option Critic

PGOC composes symbolically specified skills with a learned option critic rather than a fixed plan. Each skill is described by preconditions, effects, and a reward function, and its policy is learned through reinforcement learning; the task itself is expressed as requirements on the effects skills must produce (Zabounidis et al., 2026). Because prerequisites induce only a partial order, any single planned sequence fixes skills that do not constrain one another into an order that cannot respond to the state encountered during execution.

Initiation predicates define the admissible options, and the controller selects among them based on its observations when this set is nonempty; Section 3.1 formalizes these specifications. Section 3.2 then defines plan-progress rewards, which give the controller credit for satisfying remaining requirements. Finally, Section 3.3 gives the two-level credit assignment: each skill retains its local reward, and success adds the controller reward accumulated during execution to the skill reward, together with the controller's continuation value. Figure 1 illustrates one selection-and-execution cycle of this hierarchy.

### 3.1 Symbolic Skill Specifications

Let $s_t$ denote the environment state and $h_t$ the policy's recurrent representation at time $t$. We adopt the specification format of SCALAR: a symbolic abstraction $z_t = \phi(s_t)$ exposes the task properties used to evaluate skill preconditions and effects (Zabounidis et al., 2026). Symbolic predicates and reward functions are evaluated on this abstraction; the learned policies act from $h_t$. Episode-level

records $u_t$ track the progress already credited toward each requirement (Section 3.2), distinguishing past contributions from conditions that hold in the current state.

Each skill $o \in \mathcal{O}$ has an initiation predicate $I_o(z_t)$, a completion predicate $\tau_o(z_t, z_{t+1})$, a reward function $r_o$, and a learned policy $\pi_o(a_t \mid h_t)$; we refer to skills as options when they act as units of selection. Initiation specifies when the skill can be selected; completion identifies the transition that achieves the skill's immediate objective. One completion need not satisfy the corresponding plan requirement (Section 3.2).

**Option availability.** The admissible set contains options whose preconditions hold and that can still contribute to the plan. An optional predicate $N_o(z_t, u_t)$ marks whether outstanding requirements remain that option $o$ can advance, giving

$$\mathcal{O}_t = \{o \in \mathcal{O} : I_o(z_t) = 1 \wedge N_o(z_t, u_t) = 1\}. \tag{1}$$

Disabling either restriction replaces the corresponding predicate with the constant 1; Section 4.2 ablates these choices. Satisfying a progress target need not permanently disable its option, since later state changes may make it useful again. Initiation is checked when selecting a new option, rather than at every primitive step of a continuing execution. The native sampler requires a nonempty set to enforce eligibility; the boundary case and its fallback are described in Appendix G. Algorithm 1 describes the nonempty-set case.

**Algorithm 1** PGOC training

**Input:** Skills $\{(I_o, \tau_o, r_o)\}_{o \in \mathcal{O}}$, need predicates $\{N_o\}$, targets $\{B_j, w_j\}$
**Input:** $\gamma$, traces $\lambda_\mu, \lambda_o$, scale $\alpha$
Initialize $\mu, V^\mu$ and each $\pi_o, V_o$
**while** training budget remains **do**
  *Collect rollout*
  $c_{0,j} \leftarrow 0$; select $o_0 \sim \mu(\cdot \mid h_0)$ over $\mathcal{O}_0$ (Eq. 1)
  **for** each primitive step $t$ **do**
    Sample $a_t \sim \pi_{o_t}(\cdot \mid h_t)$
    Observe $z_{t+1}, r_t^{o_t}, q_{t,j}$
    Credit progress; form $r_t^\mu$ (Eqs. 2–3)
    Accumulate controller reward; add the balance to skill reward on success; discard at an unsuccessful execution end
    **if** execution ends ($b_t = 1$) **then**
      Set $U_{t+1}$ (Eq. 6)
      Select $o_{t+1} \sim \mu(\cdot \mid h_{t+1})$ over $\mathcal{O}_{t+1}$
    **end if**
  **end for**
  *Update policies and values*
  Compute $\hat{A}_t^\mu$ at all steps (Eqs. 4–5)
  Compute $\hat{A}_t^{\text{skill}}$, cutting at $b_t = 1$ (Eqs. 7–8)
  PPO update $\mu, V^\mu$ on selection steps
  PPO update each $\pi_o, V_o$ on its primitive steps
**end while**

### 3.2 Plan Progress Requirements

We represent each requirement $j$ by a target amount $B_j$ and a measure of the effect a transition achieves. When each successful execution produces one unit, $B_j$ is the required number of completions; measuring effects rather than counting executions also accommodates partial progress and executions that advance several requirements at once. The controller receives credit for this progress regardless of which option produced it. Each option retains its own skill reward.

Let $q_{t,j} \geq 0$ denote the observed contribution toward requirement $j$, and let $c_{t,j}$ track the progress already credited in the episode, with $c_{0,j} = 0$. Each requirement has a nonnegative weight $w_j$, allowing effects measured in different units to receive different credit. Additional credit is capped at the outstanding amount:

$$g_{t,j} = \min(q_{t,j},\ [B_j - c_{t,j}]_+)\,, \qquad c_{t+1,j} = c_{t,j} + g_{t,j}, \tag{2}$$

$$r_t^{\text{progress}} = \alpha \sum_j w_j g_{t,j}, \tag{3}$$

where $[v]_+ = \max(v, 0)$ and $\alpha \geq 0$ scales the reward. The capped progress component is therefore bounded by $\alpha \sum_j w_j B_j$ within an episode. This bound does not include additional controller reward terms. Appendix I lists the skills; Appendix G gives their Craftax instantiation and controller rewards.

### 3.3 Advantage Estimation

PGOC assigns credit at two levels, an episode-level controller estimate and an execution-level skill estimate. Let $d_t$ indicate that the episode ends after transition $t$; both estimators use the discount $\gamma$ with separate rewards, value estimates, and trace parameters.

**Controller advantage.** Let $\mu(o \mid h_t)$ denote the masked option policy and $V^\mu(h_t)$ its value estimate. Write $r_t^\mu$ for the controller reward, including the plan-progress component in Equation 3. With trace

parameter $\lambda_\mu$, the controller uses

$$\delta_t^\mu = r_t^\mu + \gamma(1-d_t)V^\mu(h_{t+1}) - V^\mu(h_t), \tag{4}$$

$$\hat{A}_t^\mu = \delta_t^\mu + \gamma\lambda_\mu(1-d_t)\hat{A}_{t+1}^\mu. \tag{5}$$

This trace continues across option boundaries, so selecting an option can receive credit for progress achieved by later options, under whichever selection policy is in use. Advantages are computed at each primitive timestep; the controller's policy and value losses use only timesteps at which an option is selected. The controller policy and value are heads on the shared recurrent state. Alongside that state they receive the admissibility mask, credited-progress records, and readiness features as inputs; the active option index is not an input, and the admissible set also constrains the controller through the selection-time logit mask. Value predictions at intermediate execution steps are not regressed directly; they enter learning only as bootstrap targets within the trace, which cuts only at episode termination. A frozen-policy diagnostic checks these intermediate predictions: against the realized discounted controller return on fresh episodes, mid-option values vary across states (standard deviation 5.68, versus 6.29 at selection steps) and track realized returns ($r = 0.89$, prediction $R^2 = 0.75$), with a mean underprediction of 0.93 reward units. The predictions used at selection and at successful boundaries are similarly aligned, so the bootstrap target is informative rather than collapsed.

**Skill advantage.** Let $V_o(h_t)$ be the value estimate for the active option $o_t$, and let $b_t = 1$ when a new option execution begins at the next timestep, including reselection of the same option. Success refers to the completion predicate, $\tau_{o_t}(z_t, z_{t+1}) = 1$. For a nonterminal transition, the continuation value is

$$U_{t+1} = \begin{cases} V_{o_t}(h_{t+1}), & b_t = 0, \\ [V^\mu(h_{t+1})]_+, & b_t = 1 \text{ and } o_t \text{ succeeded}, \\ 0, & b_t = 1 \text{ without success}. \end{cases} \tag{6}$$

Let $K_t$ be the undiscounted controller-reward balance before transition $t$, initialized to zero at each execution, and let $\sigma_t$ indicate success. The trained skill reward is $\tilde{r}_t^{o_t} = r_t^{o_t} + \sigma_t(K_t + r_t^\mu)$. The balance accumulates $r_t^\mu$ until success adds it to the skill reward; an unsuccessful execution end discards it. Pending balances persist across rollouts, and terminal success receives the accumulated credit but no bootstrap. Using this augmented reward, we compute

$$\delta_t^{\text{skill}} = \tilde{r}_t^{o_t} + \gamma(1-d_t)U_{t+1} - V_{o_t}(h_t), \tag{7}$$

$$\hat{A}_t^{\text{skill}} = \delta_t^{\text{skill}} + \gamma\lambda_o(1-d_t)(1-b_t)\hat{A}_{t+1}^{\text{skill}}. \tag{8}$$

At an option boundary, the skill trace stops and successful execution bootstraps from the clipped controller value. The positive part ensures that success is never assigned less continuation credit than failure, which receives zero. Episode termination removes bootstrapping from both estimators. Appendix E traces the rewards, credit transfer, and bootstrap values for one option execution.

**Continuation value.** At a successful boundary, Equation 6 must value the state the execution leaves behind, where several options may be admissible and the controller must choose among them. Bootstrapping the completed skill from the sampled next skill would tie its target to that option's reward definition and to the selection outcome. We instead bootstrap from the controller value, which estimates continuation under the current selection policy and the common plan-progress objective. The completed skill is thus assigned continuation credit for the opportunities it creates for further progress, in addition to its local reward.

**Cross-boundary advantage propagation.** The alternative to bootstrapping from a continuation value is to propagate credit along the prescribed plan, so that rewards from subsequent skills influence the policies that precede them. SCALAR's estimator is the concrete instance of this design that our experiments compare against (Zabounidis et al., 2026). Let $o_t$ denote the active skill in a SCALAR rollout, $r_t^{\text{S}}$ its reward, $V_o^{\text{S}}$ its value estimate, and $\lambda_{\text{S}}$ its trace parameter; its episode-level estimator is

$$\delta_t^{\text{S}} = r_t^{\text{S}} + \gamma(1-d_t)V_{o_{t+1}}^{\text{S}}(h_{t+1}) - V_{o_t}^{\text{S}}(h_t), \tag{9}$$

$$\hat{A}_t^{\text{S}} = \delta_t^{\text{S}} + \gamma\lambda_{\text{S}}(1-d_t)\hat{A}_{t+1}^{\text{S}}. \tag{10}$$

Because the trace is not cut when the skill changes, the prescribed next skill's value and subsequent advantage both enter the preceding skill's update. The boundary value targets $Y_t = V_t + \hat{A}_t$ expose

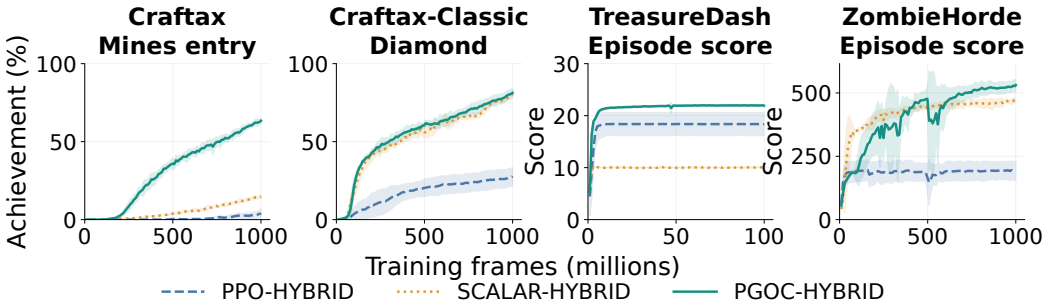


Figure 2: Hybrid training curves over seeds 3–7. Lines show available-seed means and bands sample standard deviation. Craftax tasks report achievement rates; MiniHack tasks report unshaped episode scores. SCALAR uses practice resets. Unfinished seeds contribute only observed windows. Budgets are 1B frames, except TreasureDash (100M).

the difference for a successful, nonterminal transition followed by a new option execution:

$$Y_t^{\mathrm{S}} = r_t^{\mathrm{S}} + \gamma V_{o_{t+1}}^{\mathrm{S}}(h_{t+1}) + \gamma\lambda_{\mathrm{S}}\hat{A}_{t+1}^{\mathrm{S}}, \tag{11}$$

$$Y_t^{\mathrm{skill}} = \tilde{r}_t^{o_t} + \gamma[V^{\mu}(h_{t+1})]_+. \tag{12}$$

The planned-sequence target bootstraps from the prescribed next skill and includes its propagated advantage; PGOC substitutes the controller value and ends the skill trace, so no later option's local reward enters the completed skill's update. Separate clipped PPO objectives train the skill policies on primitive-action samples and the controller on option-selection samples. Appendices G.1 and D.1 detail feedforward boundary clamping and the execution-level return.

## 4 Experiments

We compare PGOC with PPO and SCALAR on Craftax, giving PGOC the same skill library specified in Appendix I. Table 1 also reports SOL and HBS transfer with the two-reward library in Appendix B.6. Ablations test option masks, boundary bootstrapping, plan-progress rewards, and selection.

**Experimental setup.** The Craftax task is to reach the Gnomish Mines. We measure dungeon entry, unlocking the ladder through eight dungeon-floor defeats, and Mines entry. PPO, SCALAR, and PGOC use feedforward, GRU, or Hybrid backbones; Hybrid combines attention and S5 layers. The main Craftax comparison uses five training seeds (3–7) per configuration at a nominal 1B interactions, giving 45 completed policies. Frozen evaluation uses 10,240 fresh complete episodes per seed. We report equal-seed means with superscript 95% CI half-widths. The ablations use the same five-seed protocol; unavailable configurations remain dashed. PGOC normalizes the two hidden option-policy activations with non-affine RMS normalization ($\epsilon = 10^{-6}$), without bounding the logits.

SCALAR uses intermediate-state curriculum resets during training and disables them for evaluation. PPO uses no primitive-action mask; PGOC uses primitive-action and option-selection masks. PGOC and SCALAR share the same skill objectives and ten-expert sharing map; the controller, continuation target, training resets, and learner settings differ, so Section 4.4 isolates the selection rule with prescribed-order and uniform selector controls that retain the critic. Appendix F lists settings and shared controls. All runs use validated native backends; Appendix K validates Craftax's asynchronous fresh-world generation.

These comparisons isolate components. PGOC versus SCALAR tests the estimators under identical skill specifications, and PGOC versus PPO tests the supplied skill structure. The selector, option-mask, boundary-target, and reward ablations each hold the rest fixed (Sections 4.2–4.3). A discount control isolates the largest confound. At $\gamma = 0.99$, PGOC-Hybrid falls to 34.70% Mines, so the $\gamma = 0.999$ horizon accounts for much of the gap over SCALAR (Appendix K.1).

### 4.1 Long-Horizon Task Completion

PGOC improves Mines completion with the GRU and Hybrid architectures. PGOC-GRU reaches 64.53%, versus 9.35% for SCALAR-RNN and 0.00% for PPO-RNN; PGOC-Hybrid reaches 62.80%,

| Method | Craftax (%) | | | Craftax-Classic (%) | | MiniHack | | |
|---|---|---|---|---|---|---|---|---|
| | Dungeon | Ladder | Mines | Diamond | Eat Plant | Quest-Hard (%) | TreasureDash (score) | ZombieHorde (kills) |
| PPO-FC | $50.99^{\pm 2.88}$ | $1.14^{\pm 0.13}$ | $0.05^{\pm 0.02}$ | $24.17^{\pm 1.29}$ | $11.18^{\pm 1.84}$ | $0.00^{\pm 0.00}$ | $17.54^{\pm 2.79}$ | $3.93^{\pm 2.35}$ |
| PPO-RNN | $89.48^{\pm 2.48}$ | $0.04^{\pm 0.03}$ | $0.00^{\pm 0.00}$ | $29.62^{\pm 4.95}$ | $0.00^{\pm 0.01}$ | $0.00^{\pm 0.00}$ | $18.36^{\pm 2.79}$ | $5.62^{\pm 0.42}$ |
| PPO-Hybrid | $\mathbf{96.62^{\pm 0.30}}$ | $17.44^{\pm 6.61}$ | $4.32^{\pm 3.40}$ | $27.24^{\pm 7.38}$ | $0.00^{\pm 0.00}$ | $0.00^{\pm 0.00}$ | $18.36^{\pm 2.79}$ | $4.96^{\pm 1.22}$ |
| SOL | $0.10^{\pm 0.21}$ | $0.00^{\pm 0.00}$ | $0.00^{\pm 0.00}$ | $0.03^{\pm 0.05}$ | $0.00^{\pm 0.00}$ | $0.00^{\pm 0.00}$ | $14.36^{\pm 0.14}$ | $10.04^{\pm 1.55}$ |
| HBS | $0.11^{\pm 0.25}$ | $0.00^{\pm 0.00}$ | $0.00^{\pm 0.00}$ | $0.49^{\pm 1.22}$ | $0.11^{\pm 0.16}$ | $0.00^{\pm 0.00}$ | $14.37^{\pm 0.15}$ | $\mathbf{7.49^{\pm 6.28}}$ |
| SCALAR-FC | $62.80^{\pm 3.46}$ | $29.47^{\pm 2.62}$ | $5.13^{\pm 1.28}$ | $40.47^{\pm 1.77}$ | $11.20^{\pm 0.76}$ | $\mathbf{31.45^{\pm 7.25}}$ | $10.09^{\pm 0.20}$ | $\mathbf{8.84^{\pm 4.19}}$ |
| SCALAR-RNN | $89.14^{\pm 0.63}$ | $36.13^{\pm 1.10}$ | $9.35^{\pm 0.84}$ | $\mathbf{80.40^{\pm 4.90}}$ | $55.57^{\pm 20.88}$ | $\mathbf{56.02^{\pm 30.51}}$ | $10.08^{\pm 0.19}$ | $\mathbf{11.97^{\pm 1.17}}$ |
| SCALAR-Hybrid | $76.38^{\pm 3.87}$ | $27.46^{\pm 3.54}$ | $7.66^{\pm 0.39}$ | $80.43^{\pm 1.06}$ | $\mathbf{82.65^{\pm 3.23}}$ | – | $10.08^{\pm 0.18}$ | – |
| PGOC-FC | $73.54^{\pm 2.74}$ | $31.72^{\pm 11.49}$ | $8.56^{\pm 6.40}$ | $55.41^{\pm 21.83}$ | $14.94^{\pm 19.63}$ | $0.00^{\pm 0.00}$ | $22.09^{\pm 0.04}$ | $\mathbf{10.43^{\pm 5.52}}$ |
| PGOC-GRU | $\mathbf{96.38^{\pm 0.40}}$ | $\mathbf{81.60^{\pm 1.86}}$ | $\mathbf{64.53^{\pm 2.51}}$ | $\mathbf{85.20^{\pm 2.77}}$ | $0.86^{\pm 0.91}$ | $\mathbf{7.64^{\pm 20.07}}$ | $\mathbf{22.22^{\pm 0.02}}$ | $\mathbf{11.85^{\pm 4.75}}$ |
| PGOC-Hybrid | $\mathbf{96.90^{\pm 0.23}}$ | $\mathbf{80.17^{\pm 1.40}}$ | $\mathbf{62.80^{\pm 2.31}}$ | $\mathbf{80.81^{\pm 3.79}}$ | $16.12^{\pm 40.36}$ | $\mathbf{64.41^{\pm 52.32}}$ | $\mathbf{22.21^{\pm 0.04}}$ | $\mathbf{13.03^{\pm 0.82}}$ |

Table 1: Performance. Superscripts give 95% CI half-widths; bold marks the best row per column and rows whose interval overlaps it (a descriptive overlap guide, not an equivalence test); dashes denote unavailable results. Craftax, Craftax-Classic, and Quest-Hard report success percentages; TreasureDash reports mean episode score and ZombieHorde mean kills (Henaff et al., 2025).

versus 7.66% for SCALAR-Hybrid and 4.32% for PPO-Hybrid. PGOC's ladder and Mines advantage holds in every matched recurrent and Hybrid seed. The feedforward variant is weaker and noisier, with PGOC-FC reaching 8.56% versus SCALAR-FC's 5.13% and exceeding it in four of five seeds.

PPO-Hybrid and PGOC-Hybrid have similar dungeon-entry success (96.62% versus 96.90%), but PGOC completes the later progression far more often. Among episodes that unlock the ladder, PGOC-Hybrid enters the Mines in 76.8–80.3% across seeds, versus 25.4–35.1% for SCALAR-Hybrid.

Figure 2 shows Hybrid learning on Craftax Mines, Classic Diamond, randomized TreasureDash, and ZombieHorde. The full Craftax milestone curves are in Appendix B.8.

### 4.2 Prerequisites, Continued Need, and Selection

Initiation $I_o$ blocks options whose preconditions do not hold, and continued need $N_o$ removes options whose requirements are already met. The two masks are not equally important. Removing the prerequisite mask collapses PGOC-Hybrid to 0.00% Mines success and ladder completion, because the controller can then initiate skills it cannot execute. Removing the continued-need mask only trims Mines success to 55.19% from 62.80%; the plan survives, since re-selecting a finished skill wastes steps without breaking the chain. We ablate the two masks in a $2 \times 2$ design on PGOC-Hybrid, holding the skill library, learner, reward rules, primitive-action mask, and budget fixed (Table 2). The prerequisite mask is load-bearing; the continued-need mask is a refinement. The same table reports the selector controls, which Section 4.4 interprets.

### 4.3 Option-Boundary Bootstrapping

We compare the controller-value boundary target in Equation 6 with a next-option target that substitutes $[V_{o_{t+1}}(h_{t+1})]_+$ while retaining the success gate, positive-part clipping, and trace cut, and with omitting continuation credit entirely. Only the target differs; the skill library, option masks, plan-progress rewards, learner settings, and training budget are held fixed (Table 5). On PGOC-Hybrid, the controller target produces higher mean Mines success (62.80%) than both the next-option target (23.46%) and no continuation (42.05%). The next-option target also degrades dungeon entry and ladder completion sharply. The raw-reward controller-reward comparison is in Appendix B.1.

The reward comparison reverses on Craftax-Classic Eat Plant. There the plan-progress reward underperforms the raw task reward (16.12% versus 77.35% for Hybrid), and the controller-value target underperforms the next-option target (16.12% versus 50.62%). Eat Plant forces a long wait, since the crop ripens only after about 600 game steps. Frozen-policy diagnoses show the failing policies plant the crop in over 99% of episodes but die before it ripens; the learned controller's weakest configuration spends over 70% of its primitive steps executing the final option while its plant is still unripe. The plan-progress reward is exhausted once preparation completes, so it gives no signal during the survival phase, whereas the raw task reward continues to shape behavior until the

| | | Craftax (%) | | | Craftax-Classic (%) | | MiniHack | |
|---|---|---|---|---|---|---|---|---|
| Ablated | Variant | Dungeon | Ladder | Mines | Diamond | Eat Plant | Quest-Hard (%) | TreasureDash (score) |
| *Guidance masks* | | | | | | | | |
| | Full PGOC (control) | $96.90^{\pm 0.23}$ | $80.17^{\pm 1.40}$ | $62.80^{\pm 2.31}$ | $\mathbf{80.81^{\pm 3.79}}$ | $16.12^{\pm 40.36}$ | $\mathbf{64.41^{\pm 52.32}}$ | $22.21^{\pm 0.04}$ |
| | No completed mask | $95.83^{\pm 0.26}$ | $73.20^{\pm 1.61}$ | $55.19^{\pm 1.18}$ | $73.33^{\pm 2.23}$ | $0.86^{\pm 1.00}$ | $3.07^{\pm 8.24}$ | $\mathbf{22.59^{\pm 0.18}}$ |
| | No prerequisite mask | $64.99^{\pm 13.88}$ | $0.00^{\pm 0.00}$ | $0.00^{\pm 0.00}$ | $62.53^{\pm 4.95}$ | $2.98^{\pm 6.87}$ | $0.00^{\pm 0.00}$ | $22.27^{\pm 0.12}$ |
| | Neither option mask | $68.27^{\pm 7.90}$ | $1.04^{\pm 2.88}$ | $0.04^{\pm 0.10}$ | $63.45^{\pm 3.56}$ | $1.17^{\pm 1.57}$ | $0.00^{\pm 0.00}$ | $\mathbf{22.59^{\pm 0.18}}$ |
| *Selection rule (critic retained)* | | | | | | | | |
| | Learned controller (control) | $96.90^{\pm 0.23}$ | $80.17^{\pm 1.40}$ | $62.80^{\pm 2.31}$ | $\mathbf{80.81^{\pm 3.79}}$ | $16.12^{\pm 40.36}$ | $\mathbf{64.41^{\pm 52.32}}$ | $22.21^{\pm 0.04}$ |
| | Prescribed order | $\mathbf{97.39^{\pm 0.17}}$ | $\mathbf{85.21^{\pm 1.04}}$ | $\mathbf{69.38^{\pm 1.48}}$ | $\mathbf{83.59^{\pm 4.00}}$ | $0.92^{\pm 1.58}$ | $\mathbf{41.21^{\pm 3.81}}$ | $10.88^{\pm 0.19}$ |
| | Uniform over admissible | $\mathbf{97.12^{\pm 0.32}}$ | $\mathbf{84.60^{\pm 1.46}}$ | $\mathbf{67.91^{\pm 3.65}}$ | $\mathbf{81.83^{\pm 1.25}}$ | $0.16^{\pm 0.21}$ | $\mathbf{71.75^{\pm 47.44}}$ | $13.32^{\pm 0.14}$ |
| *Boundary target (critic retained)* | | | | | | | | |
| | Controller value (control) | $96.90^{\pm 0.23}$ | $80.17^{\pm 1.40}$ | $62.80^{\pm 2.31}$ | $\mathbf{80.81^{\pm 3.79}}$ | $16.12^{\pm 40.36}$ | $\mathbf{64.41^{\pm 52.32}}$ | $22.21^{\pm 0.04}$ |
| | Next option's value | $66.04^{\pm 3.60}$ | $38.75^{\pm 10.40}$ | $23.46^{\pm 10.77}$ | $\mathbf{78.25^{\pm 3.25}}$ | $\mathbf{50.62^{\pm 37.23}}$ | $\mathbf{13.90^{\pm 58.55}}$ | $22.28^{\pm 0.12}$ |
| | No continuation value | $92.63^{\pm 1.05}$ | $59.73^{\pm 1.41}$ | $42.05^{\pm 2.64}$ | $\mathbf{76.77^{\pm 3.43}}$ | $1.46^{\pm 1.61}$ | $\mathbf{17.33^{\pm 33.63}}$ | $\mathbf{22.15^{\pm 0.29}}$ |
| *Controller reward* | | | | | | | | |
| | Plan progress (control) | $96.90^{\pm 0.23}$ | $80.17^{\pm 1.40}$ | $62.80^{\pm 2.31}$ | $\mathbf{80.81^{\pm 3.79}}$ | $16.12^{\pm 40.36}$ | $\mathbf{64.41^{\pm 52.32}}$ | $22.21^{\pm 0.04}$ |
| | Raw task reward | $96.24^{\pm 0.27}$ | $69.20^{\pm 2.04}$ | $54.26^{\pm 1.45}$ | $\mathbf{77.66^{\pm 7.94}}$ | $\mathbf{77.35^{\pm 4.95}}$ | $\mathbf{29.33^{\pm 63.78}}$ | $22.27^{\pm 0.12}$ |

Table 2: Component ablations on PGOC-Hybrid. Blocks vary the option masks, the selection rule, the continuation target $U_{t+1}$, and the controller reward; control rows are the PGOC-Hybrid cells of Table 1, and seed counts, intervals, and bolding follow that table. The prescribed Craftax cells use the matched reviewer control; its Classic and MiniHack cells retain the separate prescribed runs. Backbones: Appendix B.1.

payoff. This task exposes a limit of the plan-progress construction, which suits goals with incremental, immediate progress and offers no guidance through a long, reward-free delay.

### 4.4 Learned versus Prescribed Option Selection

The main comparison evaluates complete systems, so it does not separate the option critic's credit assignment from the learned selection rule built on it. To isolate the selector, we hold the skill library, option masks, plan-progress rewards, boundary target, learner settings, and training budget fixed, and replace only the learned selection rule with two reference selectors that retain the critic (Table 2): a prescribed-order selector following the canonical SCALAR skill sequence, and a uniform selector sampling the admissible set at each decision.

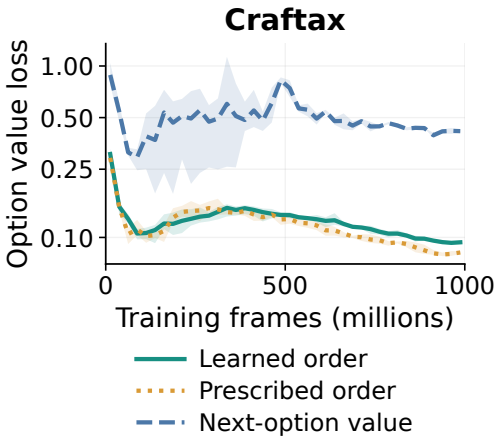


Figure 3: Option-value loss weighted by training-sample frequency; unvisited options have zero weight. Lines show available-seed means per 25M frames; shading shows their range. Prescribed order has 13.1% lower final-20% loss and 12.7% lower within-option raw advantage variance than learned order (five seeds; Table 2's † setup).

Because both controls keep the option critic, their performance isolates the selection rule within PGOC. The prescribed-order control reaches 69.38% mean Mines success for Hybrid, about 61 points above the SCALAR-Hybrid row in Table 1, which follows the same skill sequence with SCALAR's own estimator. That gap combines PGOC's credit assignment, masks, rewards, and learner settings; it is not attributed to credit assignment alone.

Whether the selection rule must additionally be learned is a separate question, and its answer depends on the task. Craftax's Mines plan is fixed by the skill prerequisites before any interaction with the world, so a fixed valid selector is not disadvantaged there. Replacing the learned selection rule with that fixed plan removes the selection policy as a source of changing targets. Under this control, learned and prescribed selection reach 62.80% and 69.38% Mines success for Hybrid, and

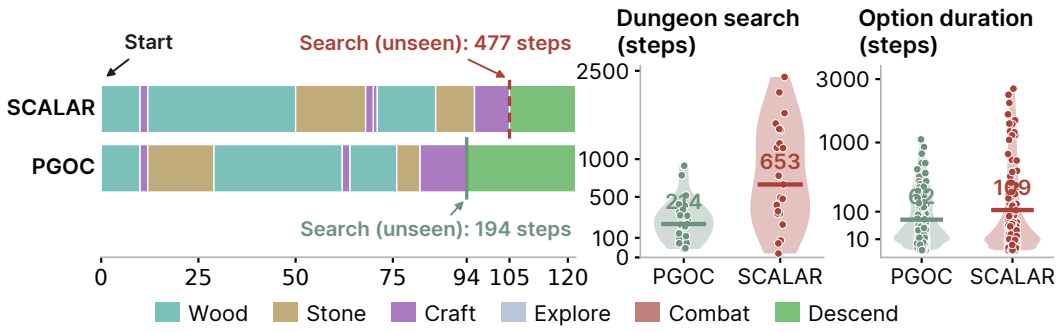


Figure 4: Final seed-3 PGOC-Hybrid and SCALAR-Hybrid policies, traced from frozen weights in 24 matched worlds. Left: one matched world — the pair closest to both methods' median search durations; from the identical start, PGOC gathers stone before wood and SCALAR the reverse, and each searches 194 versus 477 steps for the dungeon entrance. Centre: search duration from the start of descent to entry; PGOC's is shorter in 16 of the 21 worlds where both enter (3 SCALAR episodes never enter). Right: option-run duration distribution; duration axes are power-compressed.

the prescribed selector exceeds learned selection in all five matched seeds. Prescribing the order also removes one source of nonstationarity from the skills' learning problem. Under a learned selector, the continuation policy changes while the skills and critics are still training, so the controller value each skill bootstraps from changes with it; prescribing the selector removes that particular source and restricts the learned policy class. The training diagnostics are consistent with this: prescribed order lowers option-value loss and within-option advantage variance in all five seeds (Figure 3; Appendix B.3), and next-option bootstrapping gives a larger loss still. The skills, representation, and visitation distribution all continue to change, so these diagnostics are consistent with easier value estimation but do not establish it as the mechanism.

Randomized TreasureDash rewards the opposite behavior, since its floor geometry is drawn per episode and no fixed pre-interaction plan can be optimal. There the learned controller reaches a mean score of 22.21, against 10.88 for the prescribed order and 13.31 for uniform selection. Committing to the fixed order — collect the gold thresholds, then descend — forfeits the geometry, and both fixed selectors collapse to the level of SCALAR (10.08) despite retaining PGOC's masks, plan-progress rewards, and boundary bootstrap. The learned score is not a ceiling artifact either. The maximum attainable score is 29 across the 21 layouts, and the learned mean sits about 3.1 points below its per-layout optimum, so headroom remains. The credit assignment accounts for the fixed-plan task, the learned ordering accounts for the state-dependent one, and the selection rule is a design choice conditioned on task structure rather than a universally learned component.

Figure 4 traces the final seed-3 policies in 24 matched worlds. PGOC searches for the dungeon entrance faster than SCALAR in 16 of the 21 worlds where both find it (median 214 versus 653 steps), and its option runs end sooner (mean 62 versus 109 steps).

## 5 Conclusion

PGOC learns an option-critic controller and the skills jointly from their symbolic specifications, deriving the option masks and plan-progress rewards from them and bootstrapping each completed skill from the controller's continuation value. On Craftax, PGOC with recurrent backbones reaches the Gnomish Mines in 62.8–64.5% of frozen evaluation episodes, versus 7.7–9.3% for SCALAR. A fixed selector reaches 69.38% on Craftax, where a fixed plan suffices; on randomized TreasureDash, whose optimal plan is state-dependent, the learned controller wins.

**Limitations.** PGOC relies on supplied skill specifications and does not correct erroneous preconditions or reward rules during learning. Its plan-progress reward offers no guidance through a long, reward-free delay (as on Eat Plant), and the implementation can produce unresolved empty option frontiers. See Appendix A.

# 6 USE OF GENERATIVE AI, ETHICS, AND REPRODUCIBILITY

### AI USE STATEMENT

Generative AI tools were used to assist with drafting and editing text and with developing and checking code. All AI-assisted content was reviewed by the authors; LLM-generated research claims were checked against the cited literature, and AI-assisted code was tested for correctness against the reference implementations described in Appendix K. The authors take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

### ETHICS STATEMENT

This work studies reinforcement learning in simulated game environments and does not involve human subjects, personal data, or the release of datasets. We are not aware of conflicts of interest, sponsorship, bias, privacy, or security concerns particular to this work. Potential broader impacts of improved long-horizon skill composition are discussed in the Impact Statement.

### REPRODUCIBILITY STATEMENT

Appendix F describes the training and evaluation protocol, including budgets, seed ranges, frozen-evaluation quotas, and confidence-interval conventions, and points to the configurations for every method, environment, and backbone. The method-owned native backends and the reference implementations used for numerical validation are described in Appendix K, and per-seed outcomes behind every aggregate are listed in Appendix B.7. Source code is included in the supplementary material.

## REFERENCES


Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. *arXiv preprint arXiv:2204.01691*, 2022. URL https://arxiv.org/abs/2204.01691v2.

Jacob Andreas, Dan Klein, and Sergey Levine. Modular Multitask Reinforcement Learning with Policy Sketches. *arXiv preprint arXiv:1611.01796*, 2016. URL https://arxiv.org/abs/1611.01796.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The Option-Critic Architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017. doi: 10.1609/aaai.v31i1.10916. URL https://ojs.aaai.org/index.php/AAAI/article/view/10916.

Akhil Bagaria, Jason K Senthil, and George Konidaris. Skill Discovery for Exploration and Planning using Deep Skill Graphs. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 521–531, 2021. URL https://proceedings.mlr.press/v139/bagaria21a.html.

Shuo Cheng, Zhaoyi Li, Kelin Yu, and Danfei Xu. Continual Robot Learning via Language-Guided Skill Acquisition. *Transactions on Machine Learning Research*, 2026. URL https://openreview.net/forum?id=oYRNxxGN9u.

Joery A. de Vries, Thomas M. Moerland, and Aske Plaat. On Credit Assignment in Hierarchical Reinforcement Learning. *arXiv preprint arXiv:2203.03292*, 2022. URL https://arxiv.org/abs/2203.03292.

Thomas G. Dietterich. Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. *arXiv preprint arXiv:cs/9905014*, 1999. URL https://arxiv.org/abs/cs/9905014.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is All You Need: Learning Skills without a Reward Function. *arXiv preprint arXiv:1802.06070*, 2018. URL https://arxiv.org/abs/1802.06070.

Richard E. Fikes and Nils J. Nilsson. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2:189–208, 1971. doi: 10.1016/0004-3702(71)90010-5. URL https://doi.org/10.1016/0004-3702(71)90010-5.

Mikael Henaff, Scott Fujimoto, Michael Matthews, and Michael Rabbat. Scalable Option Learning in High-Throughput Environments. *arXiv preprint arXiv:2509.00338*, 2025. URL https://arxiv.org/abs/2509.00338v3.

Shengyi Huang and Santiago Ontañón. A Closer Look at Invalid Action Masking in Policy Gradient Algorithms. *arXiv preprint arXiv:2006.14171*, 2020. URL https://arxiv.org/abs/2006.14171.

León Illanes, Xi Yan, Rodrigo Toro Icarte, and Sheila A. McIlraith. Symbolic Plans as High-Level Instructions for Reinforcement Learning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pp. 540–550, 2020. doi: 10.1609/icaps.v30i1.6750. URL https://ojs.aaai.org/index.php/ICAPS/article/view/6750.

Khimya Khetarpal, Martin Klissarov, Maxime Chevalier-Boisvert, Pierre-Luc Bacon, and Doina Precup. Options of Interest: Temporal Abstraction with Interest Functions. *arXiv preprint arXiv:2001.00271*, 2020. URL https://arxiv.org/abs/2001.00271.

Martin Klissarov, Pierluca D'Oro, Shagun Sodhani, Roberta Raileanu, Pierre-Luc Bacon, Pascal Vincent, Amy Zhang, and Mikael Henaff. Motif: Intrinsic Motivation from Artificial Intelligence Feedback. *arXiv preprint arXiv:2310.00166*, 2023. URL https://arxiv.org/abs/2310.00166.

Martin Klissarov, Mikael Henaff, Roberta Raileanu, Shagun Sodhani, Pascal Vincent, Amy Zhang, Pierre-Luc Bacon, Doina Precup, Marlos C. Machado, and Pierluca D'Oro. MaestroMotif: Skill Design from Artificial Intelligence Feedback. *arXiv preprint arXiv:2412.08542*, 2024. URL https://arxiv.org/abs/2412.08542v1.

Junkyu Lee, Michael Katz, Don Joven Agravante, Miao Liu, Geraud Nangue Tasse, Tim Klinger, and Shirin Sohrabi. Hierarchical Reinforcement Learning with AI Planning Models. *arXiv preprint arXiv:2203.00669*, 2022. URL https://arxiv.org/abs/2203.00669.

Bowen Li, Mayank Mishra, Y. Isabel Liu, Stone Tao, Nishanth Kumar, Alexander G. Gray, Ruwan Wickramarachchi, Jonathan Francis, Sebastian Scherer, and Tom Silver. Recover, Discover, Plan: Learning Skills and Concepts from Robot Failures. *arXiv preprint arXiv:2606.18328*, 2026. URL https://arxiv.org/abs/2606.18328.

Zeyuan Liu, Ziyu Huan, Xiyao Wang, Jiafei Lyu, Jian Tao, Xiu Li, Furong Huang, and Huazhe Xu. World Models with Hints of Large Language Models for Goal Achieving. *arXiv preprint arXiv:2406.07381*, 2024. URL https://arxiv.org/abs/2406.07381.

Daoming Lyu, Fangkai Yang, Bo Liu, and Steven Gustafson. SDRL: Interpretable and Data-efficient Deep Reinforcement Learning Leveraging Symbolic Planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 2970–2977, 2019. doi: 10.1609/aaai.v33i01.33012970. URL https://ojs.aaai.org/index.php/AAAI/article/view/4153.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-Level Reward Design via Coding Large Language Models. *arXiv preprint arXiv:2310.12931*, 2023. URL https://arxiv.org/abs/2310.12931.

Michael Matthews, Michael Beukman, Benjamin Ellis, Mikayel Samvelyan, Matthew Jackson, Samuel Coward, and Jakob Foerster. Craftax: A Lightning-Fast Benchmark for Open-Ended Reinforcement Learning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 35104–35137, 2024. URL https://proceedings.mlr.press/v235/matthews24a.html.

Michael Tryfan Matthews, Anssi Kanervisto, Jakob Foerster, Pierluca D'Oro, Scott Fujimoto, and Mikael Henaff. Hierarchical Behaviour Spaces. *arXiv preprint arXiv:2604.24558*, 2026. URL https://arxiv.org/abs/2604.24558v1.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-Efficient Hierarchical Reinforcement Learning. *arXiv preprint arXiv:1805.08296*, 2018. URL https://arxiv.org/abs/1805.08296.

Kolby Nottingham, Bodhisattwa Prasad Majumder, Bhavana Dalvi Mishra, Sameer Singh, Peter Clark, and Roy Fox. Skill Set Optimization: Reinforcing Language Model Behavior via Transferable Skills. *arXiv preprint arXiv:2402.03244*, 2024. URL https://arxiv.org/abs/2402.03244.

Seohong Park, Oleh Rybkin, and Sergey Levine. METRA: Scalable Unsupervised RL with Metric-Aware Abstraction. *arXiv preprint arXiv:2310.08887*, 2023. URL https://arxiv.org/abs/2310.08887.

Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Van de Wiele, Volodymyr Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by Playing - Solving Sparse Reward Tasks from Scratch. *arXiv preprint arXiv:1802.10567*, 2018. URL https://arxiv.org/abs/1802.10567.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-Aware Unsupervised Discovery of Skills. *arXiv preprint arXiv:1907.01657*, 2019. URL https://arxiv.org/abs/1907.01657.

Chak Lam Shek and Pratap Tokekar. Option Discovery Using LLM-guided Semantic Hierarchical Reinforcement Learning. *arXiv preprint arXiv:2503.19007*, 2025. URL https://arxiv.org/abs/2503.19007.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artificial Intelligence*, 112:181–211, 1999. doi: 10.1016/S0004-3702(99)00052-1. URL https://www.sciencedirect.com/science/article/pii/S0004370299000521.

Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, and Sheila A. McIlraith. Reward Machines: Exploiting Reward Function Structure in Reinforcement Learning. *Journal of Artificial Intelligence Research*, 73:173–208, 2022. doi: 10.1613/jair.1.12440. URL https://arxiv.org/abs/2010.03950.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. FeUdal Networks for Hierarchical Reinforcement Learning. *arXiv preprint arXiv:1703.01161*, 2017. URL https://arxiv.org/abs/1703.01161.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An Open-Ended Embodied Agent with Large Language Models. *arXiv preprint arXiv:2305.16291*, 2023. URL https://arxiv.org/abs/2305.16291.

Xiyang Wu, Zongxia Li, Guangyao Shi, Alexander Duffy, Tyler Marques, Matthew Lyle Olson, Tianyi Zhou, and Dinesh Manocha. Co-Evolving LLM Decision and Skill Bank Agents for Long-Horizon Tasks. *arXiv preprint arXiv:2604.20987*, 2026. URL https://arxiv.org/abs/2604.20987.

Yue Wu, Yewen Fan, Paul Pu Liang, Amos Azaria, Yuanzhi Li, and Tom M. Mitchell. Read and Reap the Rewards: Learning to Play Atari with the Help of Instruction Manuals. *arXiv preprint arXiv:2302.04449*, 2023. URL https://arxiv.org/abs/2302.04449.

Tianbao Xie, Siheng Zhao, Chen Henry Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. Text2Reward: Reward Shaping with Language Models for Reinforcement Learning. *arXiv preprint arXiv:2309.11489*, 2023. URL https://arxiv.org/abs/2309.11489.

Fangkai Yang, Daoming Lyu, Bo Liu, and Steven Gustafson. PEORL: Integrating Symbolic Planning and Hierarchical Reinforcement Learning for Robust Decision-Making. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pp. 4860–4866, 2018. doi: 10.24963/ijcai.2018/675. URL https://www.ijcai.org/Proceedings/2018/675.

Renos Zabounidis, Yue Wu, Simon Stepputtis, Woojun Kim, Yuanzhi Li, Tom Mitchell, and Katia Sycara. SCALAR: Learning and Composing Skills through LLM Guided Symbolic Planning and Deep RL Grounding. *arXiv preprint arXiv:2603.09036*, 2026. URL https://arxiv.org/abs/2603.09036v1.

## IMPACT STATEMENT

This paper studies how agents compose supplied skills to accomplish long-horizon tasks, a foundational question in reinforcement learning. Progress on this question could contribute to capabilities with positive applications, such as assistants that carry out extended multi-step tasks reliably, as well as applications with negative effects. Our work does not target any specific deployment, and the empirical evaluation is confined to simulated game environments. We do not see impacts particular to this work that warrant further discussion here.

# A LIMITATIONS AND ONGOING WORK

**Evidence scope.** The Craftax PPO, SCALAR, and PGOC comparisons use five completed training seeds per configuration. Other main-comparison and ablation cells have incomplete five-seed coverage, as indicated in Tables 1 and 2. The limited seed and task coverage restrict conclusions about robustness and transfer. Completed frozen-policy evaluations and training diagnostics answer different questions; the latter describe the learning process and are not additional independent evaluation replicates.

**Attribution.** The main comparison fixes the skill specifications and sharing map, but evaluates complete systems with different learner settings (Appendix F); Section 4.4 isolates the selection rule. The no-continuation control (Appendix B.1) tests the boundary bootstrap against omitting continuation credit, and the plan-progress reward is compared with the raw task reward in the same ablation. Even under these controls, the systems still differ in controller, continuation target and practice resets.

**Applicability.** PGOC relies on supplied skill specifications and does not correct erroneous preconditions or reward rules during learning. Results depend on the architecture and supplied skill library. The implementation can produce empty option frontiers, where its reference-preserving sampler does not enforce eligibility (Appendix G). The objective-level results in Appendix D assume nonempty admissible sets and do not provide guarantees for this fallback.

**Further work.** Priorities are broader evaluation on Craftax-Classic and MiniHack, diagnosing the remaining ablation comparisons, and testing learned selection where a fixed plan cannot be optimal: the randomized TreasureDash and Quest-Hard cells of Table 1, where the prescribed and uniform Craftax controls are no longer adequate references. Appendix B.6 distinguishes the completed five-seed MiniHack baselines from the single-seed transfer pilots. The latter use different protocols from the main comparison and do not establish cross-environment or SOL/HBS baseline superiority.

# B Additional Experiments

This appendix collects the experiments beyond the main comparison of Table 1: the per-backbone component ablations, the controller configuration-selection study that chose the released configuration, the boundary-target and controller-reward ablations, the SOL/HBS transfer results, the per-seed frozen-evaluation records behind every aggregate, and per-achievement learning curves on Craftax.

## B.1 Component Ablations for All Backbones

The ablation controls below use the released PGOC configuration; the boundary-target and controller-reward comparisons are run on PGOC-Hybrid.

| | | Craftax (%) | | | Craftax-Classic (%) | | MiniHack | | |
|---|---|---|---|---|---|---|---|---|---|
| Ablated | Variant | Dungeon | Ladder | Mines | Diamond | Eat Plant | Quest-Hard (%) | TreasureDash (score) | ZombieHorde (kills) |
| *Guidance masks* | | | | | | | | | |
| | Full PGOC (control) | $\mathbf{73.54^{\pm 2.74}}$ | $\mathbf{31.72^{\pm 11.49}}$ | $\mathbf{8.56^{\pm 6.40}}$ | $\mathbf{55.41^{\pm 21.83}}$ | $14.94^{\pm 19.63}$ | $\mathbf{0.00^{\pm 0.00}}$ | $22.09^{\pm 0.04}$ | $\mathbf{10.43^{\pm 5.52}}$ |
| | No completed mask | $\mathbf{72.28^{\pm 3.86}_{n=4}}$ | $\mathbf{30.28^{\pm 1.08}_{n=4}}$ | $\mathbf{7.33^{\pm 0.87}_{n=4}}$ | $62.24^{\pm 1.88}$ | $6.58^{\pm 12.98}$ | $\mathbf{0.00^{\pm 0.00}_{n=3}}$ | $\mathbf{22.43^{\pm 0.19}}$ | – |
| | No prerequisite mask | $51.02^{\pm 5.15}_{n=4}$ | $0.00^{\pm 0.01}_{n=4}$ | $0.00^{\pm 0.00}_{n=4}$ | $0.00^{\pm 0.00}$ | $0.10^{\pm 0.08}$ | $\mathbf{0.00_{n=1}}$ | $\mathbf{22.13^{\pm 0.20}}$ | – |
| | Neither option mask | $36.13^{\pm 19.79}$ | $0.00^{\pm 0.00}$ | $0.00^{\pm 0.00}$ | $\mathbf{56.39^{\pm 39.15}}$ | $0.06^{\pm 0.04}$ | $\mathbf{0.00_{n=1}}$ | $\mathbf{22.44^{\pm 0.12}}$ | – |
| *Selection rule (critic retained)* | | | | | | | | | |
| | Learned controller (control) | $\mathbf{73.54^{\pm 2.74}}$ | $\mathbf{31.72^{\pm 11.49}}$ | $\mathbf{8.56^{\pm 6.40}}$ | $\mathbf{55.41^{\pm 21.83}}$ | $14.94^{\pm 19.63}$ | $\mathbf{0.00^{\pm 0.00}}$ | $22.09^{\pm 0.04}$ | $\mathbf{10.43^{\pm 5.52}}$ |
| | Prescribed order | $\mathbf{74.30^{\pm 1.22}}$ | $\mathbf{36.64^{\pm 5.35}}$ | $\mathbf{10.46^{\pm 3.18}}$ | $\mathbf{73.30^{\pm 2.34}}$ | $\mathbf{62.48^{\pm 42.45}}$ | $\mathbf{0.00^{\pm 0.00}_{n=2}}$ | $10.87^{\pm 0.18}$ | – |
| | Uniform over admissible | $\mathbf{74.10^{\pm 2.10}_{n=4}}$ | $\mathbf{33.97^{\pm 13.57}_{n=4}}$ | $\mathbf{10.36^{\pm 8.07}_{n=4}}$ | $58.08^{\pm 3.43}$ | $0.01^{\pm 0.01}$ | $\mathbf{0.00^{\pm 0.00}_{n=3}}$ | $13.31^{\pm 0.13}$ | – |
| *Boundary target (critic retained)* | | | | | | | | | |
| | Controller value (control) | $\mathbf{73.54^{\pm 2.74}}$ | $\mathbf{31.72^{\pm 11.49}}$ | $\mathbf{8.56^{\pm 6.40}}$ | $\mathbf{55.41^{\pm 21.83}}$ | $14.94^{\pm 19.63}$ | $\mathbf{0.00^{\pm 0.00}}$ | $22.09^{\pm 0.04}$ | $\mathbf{10.43^{\pm 5.52}}$ |
| | Next option's value | $4.93^{\pm 8.81}_{n=4}$ | $0.47^{\pm 1.45}_{n=4}$ | $0.04^{\pm 0.14}_{n=4}$ | $0.00^{\pm 0.00}$ | $34.61^{\pm 27.44}$ | $\mathbf{0.00^{\pm 0.00}_{n=3}}$ | $22.16^{\pm 0.14}$ | – |
| | No continuation value | $\mathbf{71.53^{\pm 6.16}_{n=4}}$ | $\mathbf{27.26^{\pm 13.27}_{n=4}}$ | $\mathbf{5.94^{\pm 4.56}_{n=4}}$ | $51.14^{\pm 15.92}$ | $\mathbf{50.15^{\pm 38.23}}$ | $\mathbf{0.00^{\pm 0.00}_{n=3}}$ | $\mathbf{21.95^{\pm 0.38}}$ | – |
| *Controller reward* | | | | | | | | | |
| | Plan progress (control) | $\mathbf{73.54^{\pm 2.74}}$ | $\mathbf{31.72^{\pm 11.49}}$ | $\mathbf{8.56^{\pm 6.40}}$ | $\mathbf{55.41^{\pm 21.83}}$ | $14.94^{\pm 19.63}$ | $\mathbf{0.00^{\pm 0.00}}$ | $22.09^{\pm 0.04}$ | $\mathbf{10.43^{\pm 5.52}}$ |
| | Raw task reward | $67.00^{\pm 1.83}$ | $20.49^{\pm 3.30}$ | $4.63^{\pm 1.01}$ | $\mathbf{54.95^{\pm 18.18}}$ | $\mathbf{75.92^{\pm 7.34}_{n=4}}$ | $\mathbf{0.83^{\pm 10.55}_{n=2}}$ | $\mathbf{22.13^{\pm 0.20}}$ | – |

Table 3: Component ablations for PGOC-FC across all environments under the released method, with superscript 95% CI half-widths. Cells average completed training seeds from 3–7; a subscript gives $n$ when fewer than five are available, and a single seed has no interval. Dashes denote no audited result. Columns and metrics follow Table 1; control rows are its PGOC-FC cells.

| | | Craftax (%) | | | Craftax-Classic (%) | | MiniHack | | |
|---|---|---|---|---|---|---|---|---|---|
| Ablated | Variant | Dungeon | Ladder | Mines | Diamond | Eat Plant | Quest-Hard (%) | TreasureDash (score) | ZombieHorde (kills) |
| *Guidance masks* | | | | | | | | | |
| | Full PGOC (control) | $\mathbf{96.38^{\pm 0.40}}$ | $\mathbf{81.60^{\pm 1.86}}$ | $\mathbf{64.53^{\pm 2.51}}$ | $85.20^{\pm 2.77}$ | $0.86^{\pm 0.91}$ | $\mathbf{7.64^{\pm 20.07}}$ | $22.22^{\pm 0.02}$ | $\mathbf{11.85^{\pm 4.75}}$ |
| | No completed mask | $\mathbf{94.14^{\pm 6.74}_{n=4}}$ | $\mathbf{72.86^{\pm 14.73}_{n=4}}$ | $\mathbf{56.09^{\pm 9.42}_{n=4}}$ | $77.06^{\pm 7.99}_{n=4}$ | $0.70^{\pm 0.75}_{n=4}$ | $\mathbf{0.05^{\pm 0.62}_{n=2}}$ | $\mathbf{22.50^{\pm 0.17}_{n=4}}$ | – |
| | No prerequisite mask | $61.19^{\pm 26.61}_{n=4}$ | $0.00^{\pm 0.00}_{n=4}$ | $0.00^{\pm 0.00}_{n=4}$ | $73.61^{\pm 2.25}_{n=4}$ | $13.31^{\pm 21.07}_{n=4}$ | $\mathbf{0.00_{n=1}}$ | $\mathbf{22.28^{\pm 0.16}_{n=4}}$ | – |
| | Neither option mask | $57.84^{\pm 20.14}_{n=4}$ | $0.00^{\pm 0.02}_{n=4}$ | $0.00^{\pm 0.00}_{n=4}$ | $74.21^{\pm 3.06}_{n=4}$ | $1.27^{\pm 0.46}_{n=4}$ | $\mathbf{0.00_{n=1}}$ | $\mathbf{22.50^{\pm 0.17}_{n=4}}$ | – |
| *Selection rule (critic retained)* | | | | | | | | | |
| | Learned controller (control) | $\mathbf{96.38^{\pm 0.40}}$ | $\mathbf{81.60^{\pm 1.86}}$ | $\mathbf{64.53^{\pm 2.51}}$ | $85.20^{\pm 2.77}$ | $0.86^{\pm 0.91}$ | $\mathbf{7.64^{\pm 20.07}}$ | $22.22^{\pm 0.02}$ | $\mathbf{11.85^{\pm 4.75}}$ |
| | Prescribed order | $\mathbf{92.51^{\pm 7.71}_{n=4}}$ | $\mathbf{74.34^{\pm 17.69}_{n=4}}$ | $\mathbf{60.90^{\pm 13.63}_{n=4}}$ | $\mathbf{92.68^{\pm 2.33}_{n=4}}$ | $1.39^{\pm 1.01}_{n=4}$ | $\mathbf{17.77^{\pm 225.83}_{n=2}}$ | $10.88^{\pm 0.19}$ | – |
| | Uniform over admissible | $\mathbf{96.53^{\pm 0.28}_{n=4}}$ | $\mathbf{82.94^{\pm 1.23}_{n=4}}$ | $\mathbf{67.00^{\pm 1.96}_{n=4}}$ | $\mathbf{89.25^{\pm 1.64}_{n=4}}$ | $\mathbf{28.33^{\pm 68.39}_{n=4}}$ | $\mathbf{14.84^{\pm 188.61}_{n=2}}$ | $13.30^{\pm 0.20}_{n=4}$ | – |
| *Boundary target (critic retained)* | | | | | | | | | |
| | Controller value (control) | $\mathbf{96.38^{\pm 0.40}}$ | $\mathbf{81.60^{\pm 1.86}}$ | $\mathbf{64.53^{\pm 2.51}}$ | $85.20^{\pm 2.77}$ | $0.86^{\pm 0.91}$ | $\mathbf{7.64^{\pm 20.07}}$ | $22.22^{\pm 0.02}$ | $\mathbf{11.85^{\pm 4.75}}$ |
| | Next option's value | $39.96^{\pm 16.95}_{n=4}$ | $9.87^{\pm 13.36}_{n=4}$ | $1.46^{\pm 2.25}_{n=4}$ | $0.00^{\pm 0.00}_{n=4}$ | $\mathbf{76.31^{\pm 21.66}_{n=4}}$ | $\mathbf{24.37^{\pm 309.59}_{n=2}}$ | $\mathbf{22.31^{\pm 0.25}_{n=4}}$ | – |
| | No continuation value | $94.46^{\pm 0.61}_{n=4}$ | $60.72^{\pm 5.48}_{n=4}$ | $44.58^{\pm 4.90}_{n=4}$ | $80.92^{\pm 1.72}_{n=4}$ | $4.46^{\pm 9.48}_{n=4}$ | $\mathbf{0.00^{\pm 0.00}_{n=2}}$ | $\mathbf{22.05^{\pm 0.41}_{n=4}}$ | – |
| *Controller reward* | | | | | | | | | |
| | Plan progress (control) | $\mathbf{96.38^{\pm 0.40}}$ | $\mathbf{81.60^{\pm 1.86}}$ | $\mathbf{64.53^{\pm 2.51}}$ | $85.20^{\pm 2.77}$ | $0.86^{\pm 0.91}$ | $\mathbf{7.64^{\pm 20.07}}$ | $22.22^{\pm 0.02}$ | $\mathbf{11.85^{\pm 4.75}}$ |
| | Raw task reward | $95.81^{\pm 0.44}_{n=4}$ | $74.43^{\pm 2.18}_{n=4}$ | $55.54^{\pm 2.40}_{n=4}$ | $\mathbf{91.24^{\pm 2.49}_{n=4}}$ | $\mathbf{74.74^{\pm 2.34}_{n=4}}$ | $\mathbf{0.63^{\pm 1.86}_{n=2}}$ | $\mathbf{22.28^{\pm 0.16}_{n=4}}$ | – |

Table 4: Component ablations for PGOC-GRU across all environments under the released method, with superscript 95% CI half-widths. Cells average completed training seeds from 3–7; a subscript gives $n$ when fewer than five are available, and a single seed has no interval. Dashes denote no audited result. Columns and metrics follow Table 1; control rows are its PGOC-GRU cells.

| | | Craftax (%) | | | Craftax-Classic (%) | | MiniHack | | |
|---|---|---|---|---|---|---|---|---|---|
| Ablated | Variant | Dungeon | Ladder | Mines | Diamond | Eat Plant | Quest-Hard (%) | TreasureDash (score) | ZombieHorde (kills) |
| *Guidance masks* | | | | | | | | | |
| | Full PGOC (control) | $96.90^{\pm 0.23}$ | $80.17^{\pm 1.40}$ | $62.80^{\pm 2.31}$ | $\mathbf{80.81^{\pm 3.79}}$ | $16.12^{\pm 40.36}$ | $\mathbf{64.41^{\pm 52.32}}$ | $22.21^{\pm 0.04}$ | $\mathbf{13.03^{\pm 0.82}}$ |
| | No completed mask | $95.83^{\pm 0.26}$ | $73.20^{\pm 1.61}$ | $55.19^{\pm 1.18}$ | $73.33^{\pm 2.23}$ | $0.86^{\pm 1.00}$ | $3.07^{\pm 8.24}$ | $\mathbf{22.59^{\pm 0.18}}$ | – |
| | No prerequisite mask | $64.99^{\pm 13.88}$ | $0.00^{\pm 0.00}$ | $0.00^{\pm 0.00}$ | $62.53^{\pm 4.95}$ | $2.98^{\pm 6.87}$ | $0.00^{\pm 0.00}$ | $22.27^{\pm 0.12}$ | – |
| | Neither option mask | $68.27^{\pm 7.90}$ | $1.04^{\pm 2.88}$ | $0.04^{\pm 0.10}$ | $63.45^{\pm 3.56}$ | $1.17^{\pm 1.57}$ | $0.00^{\pm 0.00}$ | $\mathbf{22.59^{\pm 0.18}}$ | – |
| *Selection rule (critic retained)* | | | | | | | | | |
| | Learned controller (control) | $96.90^{\pm 0.23}$ | $80.17^{\pm 1.40}$ | $62.80^{\pm 2.31}$ | $\mathbf{80.81^{\pm 3.79}}$ | $16.12^{\pm 40.36}$ | $\mathbf{64.41^{\pm 52.32}}$ | $22.21^{\pm 0.04}$ | $\mathbf{13.03^{\pm 0.82}}$ |
| | Prescribed order | $\mathbf{97.39^{\pm 0.17}}$ | $\mathbf{85.21^{\pm 1.04}}$ | $\mathbf{69.38^{\pm 1.48}}$ | $\mathbf{83.59^{\pm 4.00}}$ | $0.92^{\pm 1.58}$ | $\mathbf{41.21^{\pm 3.81}}$ | $10.88^{\pm 0.19}$ | – |
| | Uniform over admissible | $\mathbf{97.12^{\pm 0.32}}$ | $\mathbf{84.60^{\pm 1.46}}$ | $\mathbf{67.91^{\pm 3.65}}$ | $\mathbf{81.83^{\pm 1.25}}$ | $0.16^{\pm 0.21}$ | $\mathbf{71.75^{\pm 47.44}}$ | $13.32^{\pm 0.14}$ | – |
| *Boundary target (critic retained)* | | | | | | | | | |
| | Controller value (control) | $96.90^{\pm 0.23}$ | $80.17^{\pm 1.40}$ | $62.80^{\pm 2.31}$ | $\mathbf{80.81^{\pm 3.79}}$ | $16.12^{\pm 40.36}$ | $\mathbf{64.41^{\pm 52.32}}$ | $22.21^{\pm 0.04}$ | $\mathbf{13.03^{\pm 0.82}}$ |
| | Next option's value | $66.04^{\pm 3.60}$ | $38.75^{\pm 10.40}$ | $23.46^{\pm 10.77}$ | $\mathbf{78.25^{\pm 3.25}}$ | $\mathbf{50.62^{\pm 37.23}}$ | $\mathbf{13.90^{\pm 58.55}}$ | $22.28^{\pm 0.12}$ | – |
| | No continuation value | $92.63^{\pm 1.05}$ | $59.73^{\pm 1.41}$ | $42.05^{\pm 2.64}$ | $\mathbf{76.77^{\pm 3.43}}$ | $1.46^{\pm 1.61}$ | $\mathbf{17.33^{\pm 33.63}}$ | $\mathbf{22.15^{\pm 0.29}}$ | – |
| *Controller reward* | | | | | | | | | |
| | Plan progress (control) | $96.90^{\pm 0.23}$ | $80.17^{\pm 1.40}$ | $62.80^{\pm 2.31}$ | $\mathbf{80.81^{\pm 3.79}}$ | $16.12^{\pm 40.36}$ | $\mathbf{64.41^{\pm 52.32}}$ | $22.21^{\pm 0.04}$ | $\mathbf{13.03^{\pm 0.82}}$ |
| | Raw task reward | $96.24^{\pm 0.27}$ | $69.20^{\pm 2.04}$ | $54.26^{\pm 1.45}$ | $\mathbf{77.66^{\pm 7.94}}$ | $\mathbf{77.35^{\pm 4.95}}$ | $\mathbf{29.33^{\pm 63.78}}$ | $22.27^{\pm 0.12}$ | – |

Table 5: Component ablations for PGOC-Hybrid across all environments under the released method, with superscript 95% CI half-widths. Columns and metrics follow Table 1; control rows are its PGOC-Hybrid cells. Only the prescribed Craftax cells use the matched reviewer control; prescribed cells for other tasks and backbones come from separate run groups.

### B.2 Variance on Sparse-Reward Tasks

The released Hybrid control reaches $16.12 \pm 40.36$ on Eat Plant and the raw-task-reward ablation $77.35 \pm 4.95$. The control's wide interval reflects genuine instability: across seeds the agent plants the crop in over 99% of episodes but usually dies before the 600-tick ripening delay completes, so a single surviving seed shifts the mean by tens of points. The plan-progress reward offers no guidance through that delay, and the raw task reward, which credits the survival the task actually requires, is structurally better suited to it. We report the point estimate of the reversal but treat its magnitude as noisy; the task is a limitation of the plan-progress reward more than evidence of a precise effect size.

### B.3 Per-Option Value Losses

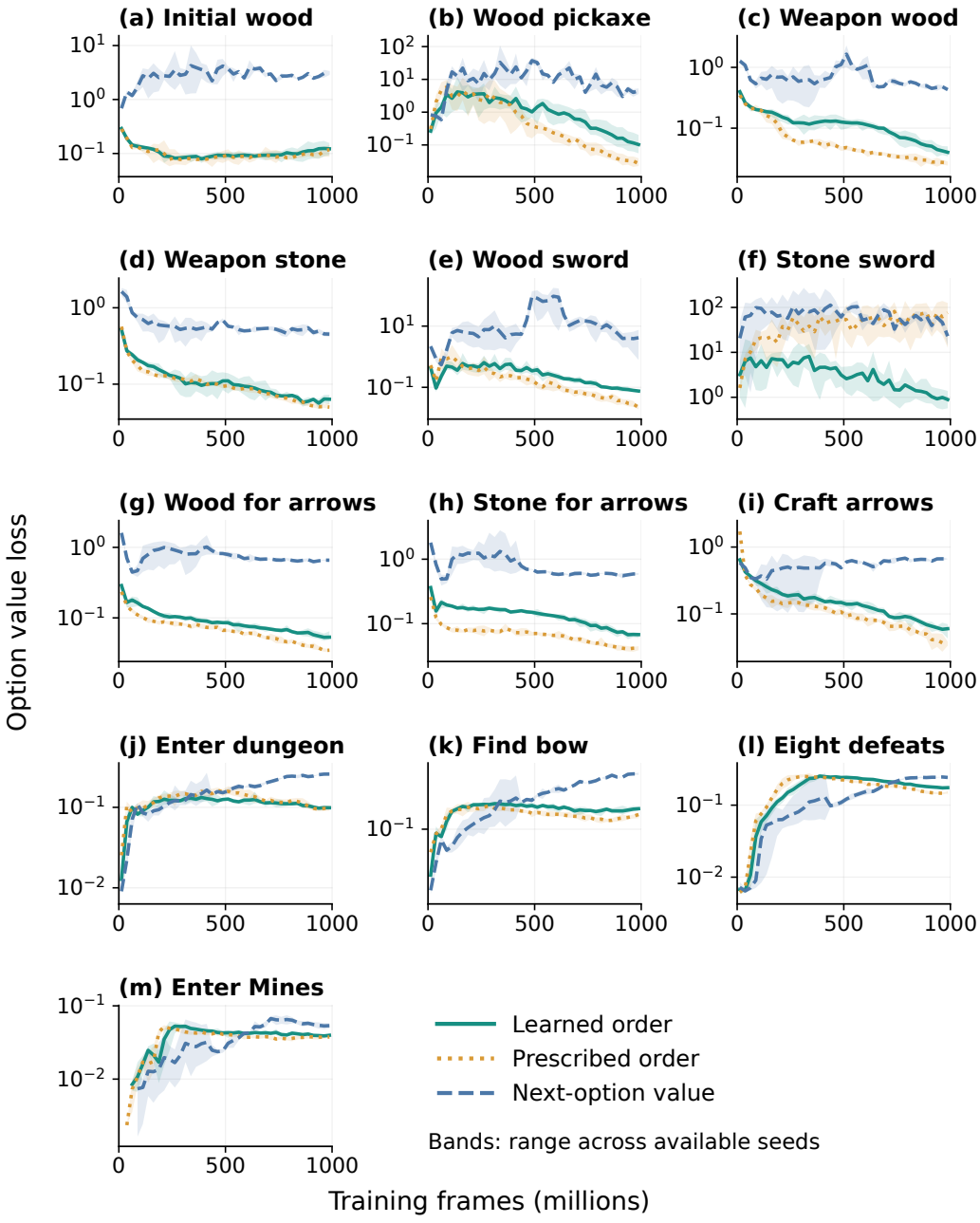


Figure 5: Per-option value losses for Craftax PGOC-Hybrid, using the same training runs as Figure 3. Each panel shows one option; lines average the seeds with samples of that option in each bin, and shading shows their range. Logarithmic axes have panel-specific limits. Prescribed order uses Table 2's † configuration.

### B.4 ADVANTAGE-VARIANCE MEASUREMENT

The advantage-variance statistic of Section 4.4 averages within-option, within-update variances of the raw GAE advantages, before actor normalization. For seed $s$, update $u$, and option $o$, the training log records the sample count $n_{suo}$, the raw advantage mean and standard deviation, and the value loss $\ell_{suo}$. Over the final 20% of training (a window $W$ of 1,526 rollout updates per run), the statistic is

$$D_s = \frac{\sum_{u \in W} \sum_o n_{suo}\, \sigma^2_{suo}}{\sum_{u \in W} \sum_o n_{suo}}, \qquad \bar{D} = \tfrac{1}{5} \sum_{s=3}^{7} D_s, \tag{13}$$

with the value-loss statistic computed under the same weighting. The reduction is in variance, not standard deviation; the corresponding full-training variance reduction is 4.6%, also in the same direction in all five seeds, so the headline comparison concerns late training rather than a uniform throughout-training effect. The prescribed-minus-learned variance difference is $-0.0249$ in raw advantage-squared units, with a paired Student-$t$ 95% confidence interval $[-0.0328, -0.0170]$ across the five seeds.

Counts measure training transitions under each option, repeated across the PPO passes; they are not option-start counts or additional independent observations. Each seed receives equal weight, and the thirteen task options routed to ten shared experts are not independent replicates. Sampled rows below the learner's minimum-option-count threshold have their gradients gated off but remain in the logged moments. Advantage dispersion and value loss are closely related, since value targets are constructed from advantages and the old value predictions; the two statistics are therefore not independent evidence, and neither measures the conditional variance of the estimator itself.

### B.5 Controller Configuration Selection

The released PGOC configuration was selected from a development study on the full-Craftax task using PGOC-Hybrid. Table 6 reports the frozen-evaluation outcome of the candidate controller configurations that reached a completed frozen evaluation. These runs precede the released method and use its full-Craftax development line; they inform the choice of controller entropy and the use of readiness features, and are distinct from the component ablations of Section B.1, which vary the released method one component at a time. The development-line prescribed selector here differs from the matched prescribed control of Section 4.4 (69.38% Mines), which trains against the released method rather than the development line.

| Configuration | Seeds | Dungeon (%) | Ladder (%) | Mines (%) |
|---|---|---|---|---|
| Prescribed selector (development line) | 5 | $97.28^{\pm 0.30}$ | $85.11^{\pm 0.83}$ | $68.68^{\pm 1.85}$ |
| Combined learned (base) | 3 | $96.23^{\pm 0.36}$ | $79.78^{\pm 1.89}$ | $61.53^{\pm 5.93}$ |
| + controller entropy 0.01 | 1 | 96.94 | 82.09 | 64.80 |
| + readiness features | 1 | 96.36 | 81.52 | 65.22 |
| + controller entropy 0.002 | 1 | 96.42 | 79.27 | 62.25 |
| + per-step controller $\lambda = 0.975$ | 1 | 96.39 | 79.79 | 62.61 |

Table 6: Controller configuration selection on the full-Craftax development line (PGOC-Hybrid), from frozen evaluations of 10,240 fresh episodes per seed; single-seed rows are point estimates. A family of SMDP-$\lambda$ controller variants was explored and abandoned before any completed frozen evaluation and is not reported numerically.

The SMDP-$\lambda$ variants (per-step controller trace parameters $0.95$, $0.8$, $0.6$, with and without value clipping and a value-trace variant) were explored during development and abandoned before any completed a frozen evaluation; we report them without numbers rather than as failed results. The readiness features and raised controller entropy are retained in the released configuration and are listed in Appendix F.4.

**Development budget and seed overlap.** The development study spans 35 recorded run attempts across 23 configuration/backbone groups in the two on-disk Craftax controller families (a verified lower bound, not an exhaustive inventory of all historical tuning). The deciding rows in Table 6 are single-seed frozen evaluations on the reported metric, so the released configuration was selected on the test metric. Development used the same training-seed range (3–7) as the reported main comparison, and readiness-feature selection used seed 3 in particular; the reported 3–7 seed set is therefore not held out from model selection. Fresh evaluation episodes are independent of training, but they are not independent of this configuration selection. We disclose this rather than claim a held-out estimate; a locked-recipe confirmation on previously unused training seeds would be required for one.

### B.6 SOL AND HBS TRANSFER RESULTS

This section reports the earlier SOL and HBS transfer pilots. The Craftax and Craftax-Classic pilots use one training seed; Table 1 gives the current multi-seed results.

#### B.6.1 CRAFTAX AND CRAFTAX-CLASSIC

We transfer SOL and HBS (Henaff et al., 2025; Matthews et al., 2026) using two worker rewards:

$$r_t^{(1)} = \#\{\text{new achievements at } t\}, \qquad r_t^{(2)} = \Delta\text{health}_t + 0.33(\Delta\text{food}_t + \Delta\text{water}_t + \Delta\text{energy}_t).$$

Both environments use unweighted new-achievement counts for the first reward. The second uses signed changes in raw game units before reset, without clipping or normalization by bar capacity. The controller receives $r_t^{(1)}$. SOL selects either worker reward; HBS selects coefficients in $\{0, 0.5, 1\}$ for each reward and uses their unnormalized linear combination, with the all-zero choice mapped to $(0.5, 0.5)$. No RND reward is used. The return reported below is the original environment return, retained as an evaluation diagnostic; it is not the worker survival reward.

All four runs use seed 3 and a nominal 1B game actions. Controller decisions do not count as game actions. Table 16 gives the architecture, optimizer, V-trace, and hierarchy settings, including the active-head likelihood and entropy calculation. Evaluation freezes the final policy and uses 2,048 complete fresh episodes (seed 1701, eight episodes per environment slot).

| | | Mean return | | Successes / 2,048 episodes | | | | |
|---|---|---|---|---|---|---|---|---|
| Environment | Method | 100M | 1B | Dungeon | Ladder | Mines | Diamond | Eat Plant |
| Craftax | SOL | 4.17 | 14.43 | 1 | 0 | 0 | 0 | 9 |
| Craftax | HBS | 4.12 | 16.56 | 1 | 0 | 0 | 0 | 0 |
| Classic | SOL | 3.84 | 15.12 | – | – | – | 0 | 1 |
| Classic | HBS | 3.83 | 15.57 | – | – | – | 0 | 0 |

Table 7: SOL/HBS transfer on Craftax and Craftax-Classic. Returns come from frozen evaluations at 100M and 1B game actions; success counts use the final policy. Dashes mark milestones absent from Craftax-Classic. These single-seed pilots are separate from Table 1.

Environment return increases during training, but these policies rarely complete the target tasks: neither full-Craftax method unlocks the ladder or enters the Mines, and neither Classic method collects a diamond. Thus, increasing return alone does not establish adequate long-horizon progression.

#### B.6.2 MINIHACK

TreasureDash uses five independent training seeds (3–7), a nominal 100M consumed policy transitions, and 1,024 frozen episodes per seed. On the original fixed-layout level, SOL reaches $24.44^{\pm 4.15}$ mean episode score and HBS $22.30^{\pm 3.39}$, with superscript 95% confidence-interval half-widths across seeds. Table 1 reports the randomized floor-geometry variant of TreasureDash. ZombieHorde uses a nominal 1B consumed policy transitions under the same five-seed evaluation protocol; SOL reaches $10.04^{\pm 1.55}$ and HBS $7.49^{\pm 6.28}$ zombies killed per episode (Table 1).

Both tasks use the released SOL configuration: 1,920 parallel environments, 256-step rollouts, a 512-unit LSTM, 64-dimensional glyph embeddings, a $12 \times 12$ crop, and 32,768-transition minibatches with one optimization epoch and no shuffling. Adam uses learning rate $10^{-4}$, $\epsilon = 10^{-7}$, and $(\beta_1, \beta_2) = (0.9, 0.999)$. The discount is 0.99; V-trace importance weights are clipped at 1. Policy clipping is 0.1 with the released reciprocal lower ratio bound, value clipping is 1, entropy and value-loss coefficients are 0.003 and 0.5, and gradient norm is clipped at 4. The native learner overlaps one population rollout with learning.

TreasureDash workers receive $20r_{\text{stairs}}$ or $r_{\text{gold}}$; the controller receives $0.001(20r_{\text{stairs}} + r_{\text{gold}})$. ZombieHorde workers receive task score or $20\Delta\text{health}$; the controller receives $0.001$ times task score. Worker rewards are scaled by 0.01 and clipped at 100. SOL selects one reward; HBS uses the released unnormalized two-coefficient mixing rule in $\{0, 0.5, 1\}$, with an all-zero choice replaced by

$(0.5, 0.5)$. No task-specific HBS configuration was released, so HBS transfers this rule to the SOL task settings. Both use option durations $1, 2, 4, \ldots, 128$, the released all-head likelihood, and no RND or curriculum resets. Consumed transitions include controller-only decisions; actual game actions are logged separately. Frozen metrics are unscaled episode score for TreasureDash and zombies killed per episode for ZombieHorde.

### B.7 PER-SEED DISTRIBUTION

Figure 6 shows Gnomish Mines frozen success rates for PPO, SCALAR, and PGOC from Table 1, and the PGOC-Hybrid ablations from Table 2. It uses the same completed evaluations and equal-seed means as those tables. Prescribed and uniform selection have higher means than the learned Hybrid controller, whereas the next-option and no-continuation targets have lower means. The per-seed values and their audited sources are retained with the figure.

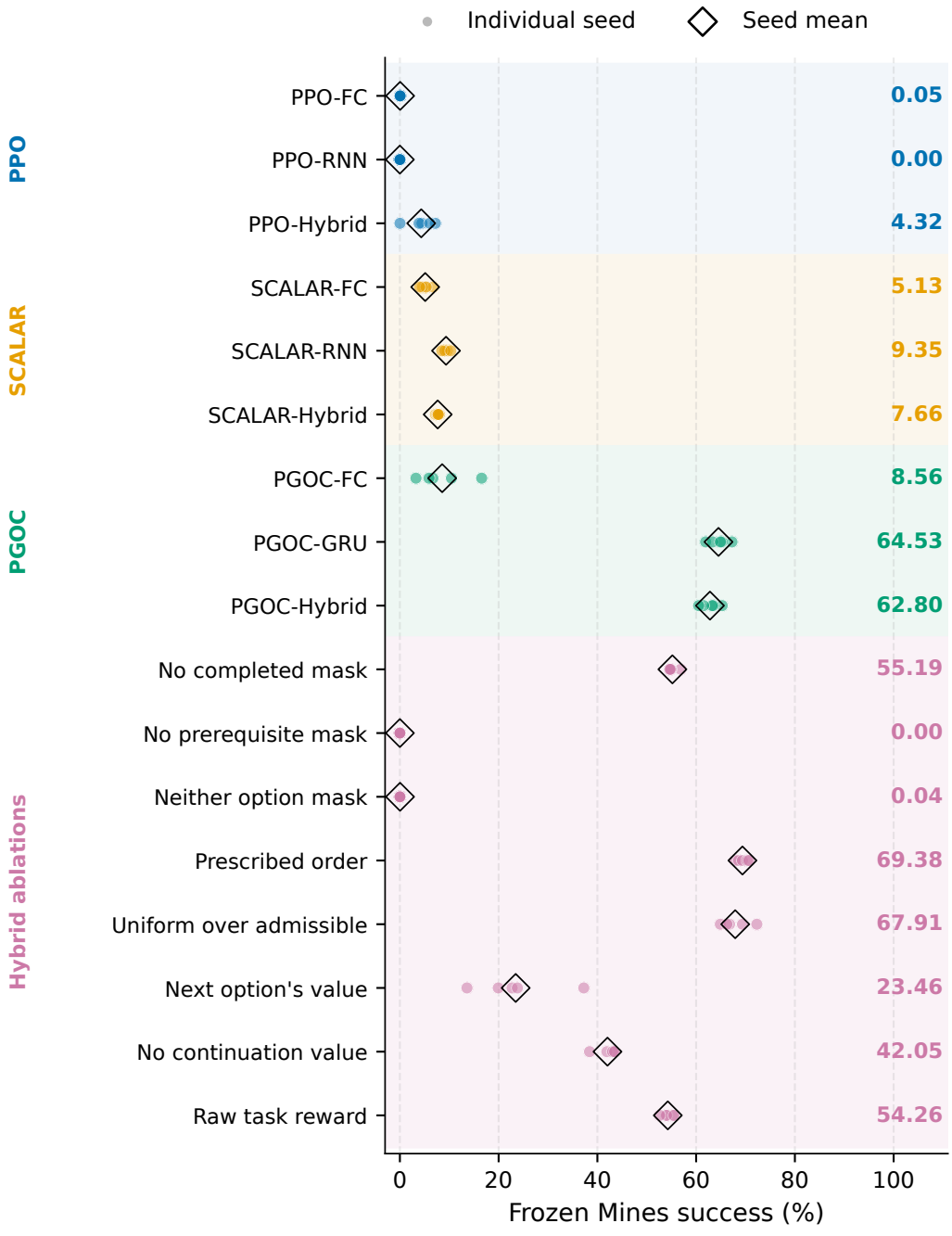


Figure 6: Per-seed Gnomish Mines success for PPO, SCALAR, and PGOC (Table 1) and the Hybrid ablations (Table 2). Points show seeds 3–7; diamonds and numbers show their means. Unavailable configurations are omitted. Prescribed order uses the matched Craftax-Hybrid reviewer control, as in Table 2.

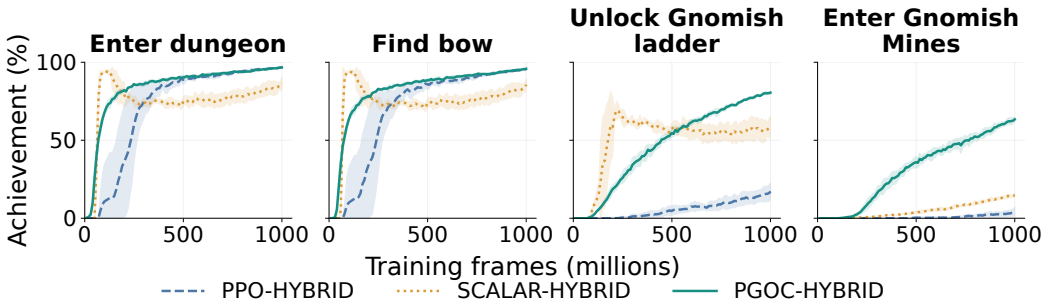


Figure 7: Craftax Hybrid milestone learning for PPO, SCALAR, and PGOC. Lines show equal-seed means and bands show sample standard deviations over seeds 3–7, at a nominal 1B frames. SCALAR uses practice resets during training; Table 1 reports frozen evaluation from fresh worlds.

### B.8 Per-Achievement Learning on Craftax

Figure 7 shows Hybrid learning across dungeon entry, bow acquisition, ladder unlock, and Mines entry. These milestones locate the late-progression gap while the main Figure 2 compares four tasks.

Figure 8 shows PGOC's per-achievement success rate over the full training budget for the three backbones, restricted to the achievements reached during training. The curves make the learning order visible: resource and basic-tool achievements saturate early, while the dungeon and Mines milestones rise only after the prerequisite equipment is in place. Figure 9 compares the frozen per-achievement rates of PPO, SCALAR, and PGOC on the Hybrid backbone. PPO and SCALAR spread success across nearly every achievement, whereas PGOC concentrates on the Mines progression and skips orthogonal ones (for example, iron and diamond tools that the plan does not require). Beyond the four milestones above, comparable baseline achievement curves were not recorded, so the broader cross-method comparison is at the frozen endpoint.

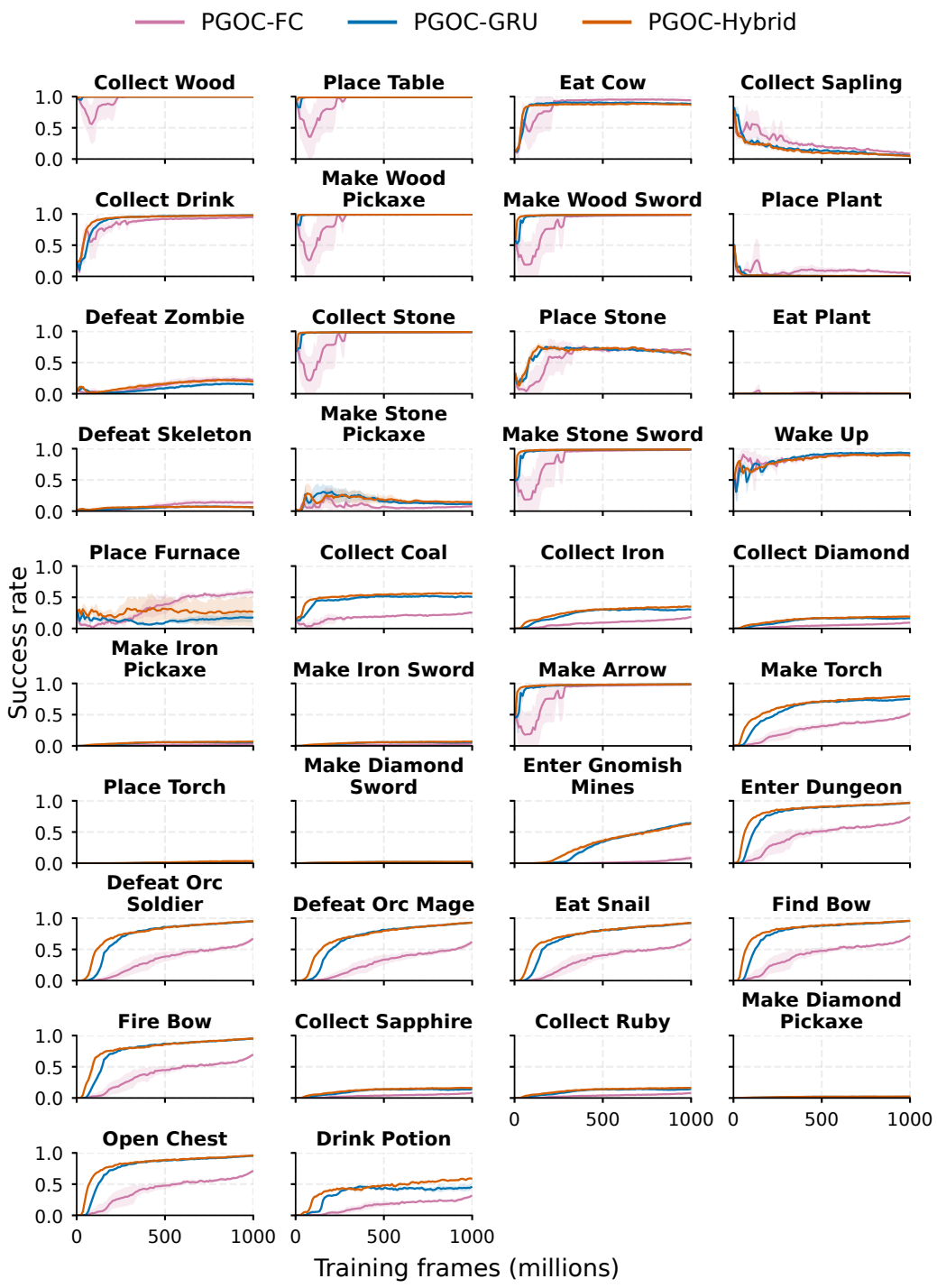


Figure 8: PGOC per-achievement training curves on Craftax, for the achievements reached during training. Lines are equal-seed means across seeds 3–7 and bands are standard deviations, for the FC, GRU, and Hybrid backbones of the released configuration. Success rate is the fraction of episodes completing the achievement within each training window.

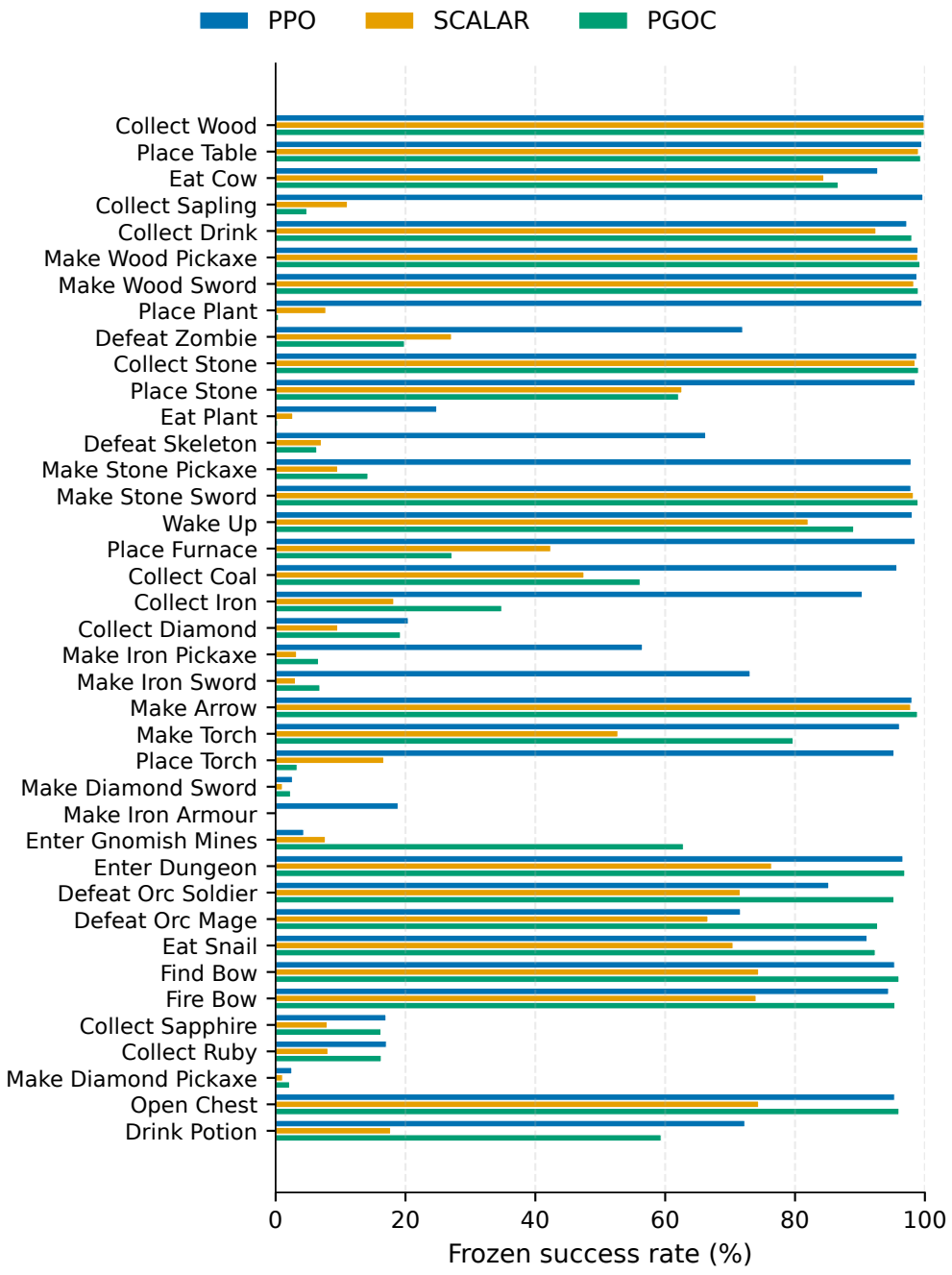


Figure 9: Frozen per-achievement success rates for PPO, SCALAR, and PGOC on the Hybrid backbone, mean across seeds 3–7, on the achievements reached by any method in frozen evaluation. PPO and SCALAR share the Table 1 cohort; PGOC uses the released configuration.

# C NOTATION

Table 8 collects the symbols used in Section 3 and Appendix D.

Table 8: Notation. The right column gives the equation or section in which each symbol is defined.

| Symbol | Meaning | Defined in |
|---|---|---|
| *State and observations* | | |
| $s_t$ | Environment state at primitive step $t$ | Sec. 3.1 |
| $h_t$ | Recurrent representation; input to all learned policies | Sec. 3.1 |
| $z_t = \phi(s_t)$ | Symbolic abstraction used to evaluate predicates and rewards | Sec. 3.1 |
| $u_t$ | Episode-level record of progress already credited | Sec. 3.1 |
| *Skills and availability* | | |
| $\mathcal{O}$ | Skill (option) library | Sec. 3.1 |
| $I_o(z_t)$ | Initiation predicate of option $o$ | Sec. 3.1 |
| $\tau_o(z_t, z_{t+1})$ | Completion predicate of option $o$ | Sec. 3.1 |
| $r_o$ | Local reward function of option $o$ | Sec. 3.1 |
| $\pi_o(a_t \mid h_t)$ | Learned primitive-action policy of option $o$ | Sec. 3.1 |
| $N_o(z_t, u_t)$ | Continued-need predicate of option $o$ | Eq. 1 |
| $\mathcal{O}_t$ | Admissible option set at step $t$ | Eq. 1 |
| *Plan-progress rewards* | | |
| $B_j$ | Target amount of requirement $j$ | Sec. 3.2 |
| $w_j$ | Nonnegative weight of requirement $j$ | Eq. 3 |
| $q_{t,j}$ | Observed contribution toward requirement $j$ at $t$ | Eq. 2 |
| $c_{t,j}$ | Progress already credited toward $j$ in the episode | Eq. 2 |
| $g_{t,j}$ | Newly credited progress toward $j$ at $t$ | Eq. 2 |
| $r_t^{\text{progress}}$ | Controller plan-progress reward | Eq. 3 |
| $\alpha$ | Scale of the plan-progress reward | Eq. 3 |
| *Controller and skill estimators* | | |
| $\mu(o \mid h_t)$ | Masked controller policy over options | Sec. 3.3 |
| $V^\mu(h_t), V_o(h_t)$ | Controller and option value estimates | Sec. 3.3 |
| $r_t^\mu$ | Controller reward, including plan progress | Eq. 4 |
| $\gamma$ | Discount shared by both estimators | Sec. 3.3 |
| $\lambda_\mu, \lambda_o$ | Controller and skill trace parameters | Eqs. 5, 8 |
| $d_t$ | Episode-termination indicator | Sec. 3.3 |
| $b_t$ | Option-boundary indicator (new execution at $t{+}1$) | Sec. 3.3 |
| $\sigma_t$ | Success indicator, $\tau_{o_t}(z_t, z_{t+1}) = 1$ | Sec. 3.3 |
| $K_t$ | Undiscounted controller-reward balance before $t$ | Sec. 3.3 |
| $\tilde{r}_t^{o_t}$ | Trained skill reward, $r_t^{o_t} + \sigma_t(K_t + r_t^\mu)$ | Sec. 3.3 |
| $U_{t+1}$ | Continuation value at an option boundary | Eq. 6 |
| $\delta_t^\mu, \hat{A}_t^\mu$ | Controller TD error and advantage | Eqs. 4–5 |
| $\delta_t^{\text{skill}}, \hat{A}_t^{\text{skill}}$ | Skill TD error and advantage | Eqs. 7–8 |
| *SCALAR comparison* | | |
| $r_t^{\text{S}}, V_o^{\text{S}}, \lambda_{\text{S}}$ | SCALAR reward, value estimate, trace parameter | Eqs. 9–10 |
| $Y_t^{\text{S}}, Y_t^{\text{skill}}$ | Boundary value targets at a successful boundary | Eqs. 11–12 |

# D FORMAL PROPERTIES OF THE OPTION OBJECTIVES

This appendix states the formal properties of the execution-level objectives defined in Section 3.3, and separates properties of an objective from guarantees about its optimization. The results assume $0 < \gamma < 1$, bounded rewards, nonempty admissible sets, and option durations $D \geq 1$ with almost-sure termination — the semi-Markov Bellman setting for options (Sutton et al., 1999). The nonempty-set assumption is a condition of this analysis, not a property guaranteed by the Craftax implementation's empty-frontier fallback (Appendix G). None of these statements order the quality of policies trained with these objectives under finite-trace GAE and PPO; they concern the objectives themselves.

### D.1 Execution-Level Return

The skill reward specifies what an option should accomplish; the controller continuation instead values the state left for subsequent execution. For a complete option execution of duration $D$ with success indicator $S$, write the discounted reward sums

$$L_o = \sum_{j=0}^{D-1} \gamma^j r^o_{t+j}, \qquad C = \sum_{j=0}^{D-1} \gamma^j r^\mu_{t+j}, \tag{14}$$

for the skill and controller rewards, and $C_{\text{pay}} = \gamma^{D-1} \sum_{j=0}^{D-1} r^\mu_{t+j}$ for the accumulated controller credit delivered on the final reward transition. For $\lambda_o = 1$ the execution return is $L_o + SC_{\text{pay}} + \gamma^D S[V^\mu(h_{t+D})]_+$, with zero terminal continuation. All PGOC configurations in the experiments use this delayed credit transfer.

### D.2 Return omitted by the success gate

To isolate the success gate from reward timing, compare the implemented objective with a reference that credits controller rewards at their original timesteps, $Y^{\text{acc}}_o = L_o + SC + \gamma^D S[V^\mu(h_{t+D})]_+$. Let $\mathcal{V}_0$ be the exact controller value under a fixed reference library, $Q_\pi = \mathbb{E}_\pi[C + \gamma^D \mathcal{V}_0(X_D)]$ the one-execution value of a candidate option policy, and $W_{\text{cont}}(\pi) = \mathbb{E}_\pi[\gamma^D S \mathcal{V}_0(X_D)]$ and $W_{\text{acc}}(\pi) = \mathbb{E}_\pi[SC + \gamma^D S \mathcal{V}_0(X_D)]$ its continuation-only and accumulated-credit components.

**Proposition 1** (Return omitted by success-conditioned credit)**.** *If $C \geq 0$ and $\mathcal{V}_0 \geq 0$, then $Q_\pi - W_{\text{cont}}(\pi) = \mathbb{E}_\pi[C + \gamma^D(1-S)\mathcal{V}_0(X_D)]$ and $Q_\pi - W_{\text{acc}}(\pi) = \mathbb{E}_\pi[(1-S)(C + \gamma^D \mathcal{V}_0(X_D))]$. Both are nonnegative, and adding accumulated credit reduces the omitted return by exactly $\mathbb{E}_\pi[SC]$.*

*Proof sketch.* Subtract each component from $Q_\pi$; their difference is $W_{\text{acc}} - W_{\text{cont}} = \mathbb{E}_\pi[SC] \geq 0$. □

For nonnegative controller rewards, $C_{\text{pay}} \leq C$ on each trajectory, so the delayed transfer omits both failed-execution credit and part of the discounted reward on successful executions; the exact reduction from the continuation-only reference is $\mathbb{E}[SC_{\text{pay}}]$.

### D.3 Bounded loss of local return

Adding controller credit changes which behaviors a skill prefers. Consider a weighted objective $J(\pi) = \ell_\pi + \beta W_\pi$ with $\beta \geq 0$ and bounded contextual credit $0 \leq W_\pi \leq B$, where $\ell_\pi$ is the option's local return. The progress cap in Equation 3 supplies the range $B$ for the capped progress component.

**Proposition 2** (Bounded loss of local return)**.** *Let $\ell^* = \sup_\pi \ell_\pi$. If $J(\pi_\epsilon) \geq \sup_\pi J(\pi) - \epsilon_{\text{opt}}$ for $\epsilon_{\text{opt}} \geq 0$, then $\ell^* - \ell_{\pi_\epsilon} \leq \beta B + \epsilon_{\text{opt}}$.*

*Proof sketch.* Nonnegative credit gives $\sup_\pi J(\pi) \geq \ell^*$, while $J(\pi_\epsilon) \leq \ell_{\pi_\epsilon} + \beta B$; combine with approximate optimality. □

A positive success-gated bonus therefore does not universally preserve maximal completion probability: under exact optimization and a completion-probability local objective, a policy that always succeeds with zero contextual payoff can tie one that succeeds less often but receives credit on success. The bound concerns the quantity actually optimized; a range bound on exact returns is not automatically a range bound on an unconstrained critic prediction.

### D.4 Improving one option improves the composition

Replace the reference policy $\pi_0$ of one option $o$ by $\pi_1$ at every invocation, holding the controller and other options fixed. Let $\mathcal{V}_0$ and $\mathcal{V}_1$ be the composed values before and after, and $Q_i = \mathbb{E}_{\pi_i}[C + \gamma^D \mathcal{V}_0(X_D)]$ evaluate one execution under the old continuation value. Let $K_1$ be the new library's discounted decision-time transition kernel, whose row sums satisfy $\kappa_1 \leq \gamma < 1$.

**Proposition 3** (Replacing one option)**.** $(\mathcal{T}_1\mathcal{V}_0)(x) - \mathcal{V}_0(x) = \mu(o \mid x)[Q_1(x) - Q_0(x)]$. *If this residual is at least $-d$ everywhere with $d \geq 0$, then $\mathcal{V}_1(x) \geq \mathcal{V}_0(x) - d/(1-\kappa_1)$; in particular, $Q_1 \geq Q_0$ at every decision state where the option can be selected implies $\mathcal{V}_1 \geq \mathcal{V}_0$.*

*Proof sketch.* Only option $o$ changes the Bellman expectation, giving the residual identity; the bound follows by iterating $\mathcal{V}_1 - \mathcal{V}_0 = \sum_{n\geq 0} K_1^n r$ with $\|K_1\|_\infty \leq \kappa_1 < 1$. □

The condition is pointwise: a favorable average under the old initiation distribution does not establish it, and changing the option can change future visitation. Updates to shared parameters that alter the controller or other options are outside this one-option result.

# E WORKED EXAMPLE OF TWO-LEVEL CREDIT FLOW

Table 9 traces one episode fragment through the quantities defined in Section 3.3, in the illustrative setting of Figure 1: the combat option has a one-defeat completion predicate, the combat requirement has target $B = 8$ and unit weight, $\alpha = 1$, $\gamma = 0.999$, and the episode begins with credited progress $c_0 = 6$.

The controller selects combat at $t$=0; the execution runs for three primitive steps and ends with a defeat at $t$=3. The selection-step controller reward includes the unit option-selection bonus, so $r_0^\mu = 1$; the two exchange steps produce no credit. The defeat at $t$=3 completes the execution ($\sigma_3 = 1$) and contributes $q_3 = 1$, capped by the outstanding amount: $g_3 = \min(1, [8-6]_+) = 1$, giving $r_3^\mu = 1$ and $c_4 = 7$. Combat remains needed ($7 < 8$), so it stays admissible. The accumulated balance $K_3 = r_0^\mu + r_1^\mu + r_2^\mu = 1$ is added to the skill reward at the boundary, which also receives the current controller reward: $\tilde{r}_3^o = r_3^o + \sigma_3(K_3 + r_3^\mu) = 3$. Because the execution succeeded, the continuation value is the clipped controller value $U_4 = [V^\mu(h_4)]_+$ rather than zero.

A rollout boundary intervenes before the next execution completes: the fragment is split after $t$=5, and combat's second execution resumes in the next rollout with its pending balance intact, $K_6 = 1$ (Section 3.3). That execution ends at $t$=7 without a defeat: the accumulated balance is discarded, the continuation value is $U_8 = 0$, and no plan progress is credited.

Table 9: Example trace of the two-level credit flow, constructed for illustration rather than recorded from a rollout. Columns are defined in Section 3.3 (Eqs. 6–8 and 4–5).

| $t$ | Event | $r_t^o$ | $q_t$ | $r_t^\mu$ | $K_t$ | $b_t$ | $\tilde{r}_t^o$ | $U_{t+1}$ | $\delta_t^\mu$ |
|---|---|---|---|---|---|---|---|---|---|
| 0 | *select combat* | — | 0 | 1 | 0 | 0 | — | — | $1 + \gamma V^\mu(h_1) - V^\mu(h_0)$ |
| 1 | combat: exchange | 0 | 0 | 0 | 1 | 0 | 0 | $V_o(h_2)$ | $\gamma V^\mu(h_2) - V^\mu(h_1)$ |
| 2 | combat: exchange | 0 | 0 | 0 | 1 | 0 | 0 | $V_o(h_3)$ | ⋮ |
| 3 | combat: defeat, $\sigma_3$=1 | 1 | 1 | 1 | 1 | 1 | 3 | $[V^\mu(h_4)]_+$ | $1 + \gamma V^\mu(h_4) - V^\mu(h_3)$ |
| 4 | *select combat* | — | 0 | 1 | 0 | 0 | — | — | $1 + \gamma V^\mu(h_5) - V^\mu(h_4)$ |
| 5 | combat: exchange | 0 | 0 | 0 | 1 | 0 | 0 | $V_o(h_6)$ | ⋮ |
| | *rollout boundary — pending balance persists ($K_6 = 1$)* | | | | | | | | |
| 6 | combat: exchange | 0 | 0 | 0 | 1 | 0 | 0 | $V_o(h_7)$ | ⋮ |
| 7 | combat: ends, no success | 0 | 0 | 0 | 1 | 1 | 0 | 0 | $\gamma V^\mu(h_8) - V^\mu(h_7)$ |

The trace makes the two estimators' asymmetry concrete. At $t$=3 the skill trace cuts ($\hat{A}_3^{\text{skill}} = \delta_3^{\text{skill}}$, since the $(1 - b_3)$ factor in Equation 8 is zero), so no later option's local reward enters the completed skill's update; with $V^\mu(h_4) = 2.0$, the boundary TD error is $\delta_3^{\text{skill}} = 3 + 0.999 \times 2.0 - V_o(h_3)$. The controller trace, by contrast, continues through both boundaries: later TD errors propagate backward through $\gamma\lambda_\mu$ factors into the advantages of the earlier selection steps $\hat{A}_0^\mu$ and $\hat{A}_4^\mu$, so each selection can receive credit for progress achieved by the option it chose and by later options. The controller's policy and value losses use only the selection steps ($t$=0 and $t$=4 here); intermediate execution steps contribute value predictions only as bootstrap targets within the controller trace.

# F TRAINING AND EVALUATION PROTOCOL

**Code availability.** The code is included in the supplementary material.

**Reproducibility.** The supplementary archive contains the method-owned native C/CUDA backends and the reference implementations used for numerical validation (Appendix K), the configurations for every method–environment–backbone combination in Tables 11–16, and the frozen-evaluation harness. The Craftax and Craftax-Classic environments are those of Matthews et al. (2024); the MiniHack tasks use the released configurations referenced in Appendix H. All numerical results derive from frozen policies evaluated on fresh episodes with training and evaluation seed ranges given in Table 10; Appendix B.7 reports the per-seed distribution behind the central Craftax Mines aggregates, and the interval conventions in this section. Every reported cell is traceable to its training run and frozen evaluation.

**Budgets, seeds, and evaluation.** The main Craftax comparison and its ablations train with 1,024 parallel environments under a nominal 1B-interaction budget, rounded down to whole rollouts; the learning-rate schedule spans the same 1B interactions. The main comparison and original-library option-mask, boundary-target, selector, and reward ablations use five equally weighted training seeds (3, 4, 5, 6, 7). The full-mask controller-target PGOC policies are shared controls across the applicable comparisons, rather than additional independent observations. The current RMS-controller cohort covers the nine main configurations and the fifteen additional ablation configurations, reported in Appendix B.7 and Appendix B.1. Only policies that completed the full training budget and frozen evaluation enter numerical results; missing seeds are shown explicitly in Appendix B.7. All learned-controller results use RMS normalization without logit bounding. Prescribed and uniform controls are reused because they bypass the option-policy logits and their gradients; their learner and task settings are unchanged. Evaluation freezes policy and optimizer parameters and runs 10,240 fresh, complete episodes (evaluation seed 1701; ten episodes per environment slot) with no artificial step cutoff; SCALAR's curriculum resets are disabled for evaluation. Ray schedules independent single-GPU native training and evaluation jobs. All runs use the continuously replenished asynchronous fresh-world stream; Appendix K describes its validation and throughput checks. Fresh-world generation is distinct from SCALAR's intermediate-state practice resets.

**Confidence intervals.** Numerical results use $\bar{x}^{\pm h}$, where $h = t_{0.975,n-1}s/\sqrt{n}$ is the half-width of a two-sided 95% Student-$t$ confidence interval for the equally weighted seed mean; $s$ is the sample standard deviation and $n = 5$ for completed configurations. single-seed cells have no across-seed interval. The separate single-seed Craftax and Classic pilots in Appendix B.6 have no across-seed confidence intervals. Intervals use unrounded per-seed measurements, not pooled episodes, and are not clipped to the metric's bounds. The approximation assumes approximately normal seed means; zero observed seed variance yields zero half-width. Training-curve bands remain descriptive sample standard deviations, as labeled. Figure 6 reports Mines seed distributions for PPO, SCALAR, PGOC, and the Hybrid ablations, making their across-seed spread directly visible.

### F.1 Budgets and Hyperparameter Conventions

Table 10: Training budgets and frozen-evaluation quotas per training seed. We report budgets in environment interactions (one policy step in one environment); where a method's counter differs (game actions or consumed policy transitions), the distinction is stated below. Main comparisons use seeds 3–7. Evaluation episodes start from fresh worlds; SCALAR's practice resets are disabled.

| Environment | Training budget | Training seeds | Evaluation episodes |
|---|---|---|---|
| *Main comparisons* | | | |
| Craftax | 1B interactions | 3–7 | 10,240 |
| Craftax-Classic | 1B interactions | 3–7 | 10,240 |
| MiniHack Quest-Hard | 1B interactions | 3–7 | 1,024 |
| MiniHack TreasureDash | 100M interactions | 3–7 | 1,024 |
| MiniHack ZombieHorde | 1B interactions | 3–7 | 1,024 |
| *Exploratory pilots* | | | |
| SOL/HBS: Craftax and Classic | 1B game actions | 3 | 2,048 |
| PGOC-GRU+RND: TreasureDash | 100,007,936 actions | 42 | 1,000 |

Main-comparison budgets are nominal and rounded to complete training batches. MiniHack counts policy actions, which can differ from game ticks; SCALAR's practice-reset replay does not count toward this budget. The TreasureDash PGOC pilots report their exact completed action count. The Craftax, Classic, and Quest-Hard SOL/HBS pilots count actual game actions and exclude controller-only decisions. TreasureDash and ZombieHorde SOL/HBS use the released code's consumed policy-transition counter, including controller-only decisions, rounded up to a complete 32,768-transition batch.

Tables 11–16 give the settings for each method, environment, and backbone. A value spanning the architecture columns applies to every displayed backbone; "–" indicates an inapplicable setting. Each table includes its optimizer settings. Learning-rate decay spans the nominal training budget unless stated otherwise. Recurrent minibatches preserve complete environment sequences. The transfer settings describe the results in Appendix B.6; unreported method–environment pairs in Table 1 do not imply additional completed experiments.

### F.2 PPO HYPERPARAMETERS

Table 11: PPO on Craftax, Craftax-Classic, and MiniHack Quest-Hard. The three environments share the optimization settings below. Environment-specific encoders and Hybrid backbones are distinguished explicitly.

| Hyperparameter | FC | GRU | Hybrid |
|---|---|---|---|
| *Rollouts and optimization* | | | |
| Parallel environments | | 1,024 | |
| Rollout length | 64 | 64 | 128 |
| Transitions per rollout | 65,536 | 65,536 | 131,072 |
| Transitions per minibatch | 8,192 | 8,192 | 16,384 |
| Minibatches per epoch | | 8 | |
| Optimization epochs | | 4 | |
| Initial learning rate | | $2 \times 10^{-4}$ | |
| Learning-rate schedule | | Linear decay to zero; no warmup | |
| Optimizer | | Adam, $(\beta_1, \beta_2) = (0.9, 0.999)$ | |
| Adam $\epsilon$ | | $10^{-5}$ | |
| Maximum gradient norm | | 1.0 | |
| *Returns and losses* | | | |
| Advantage estimator | | GAE | |
| Discount $\gamma$ | 0.99 | 0.99 | 0.999 |
| GAE $\lambda$ | | 0.8 | |
| Policy / value clipping | | 0.2 / 0.2 | |
| Entropy coefficient | 0.01 | 0.01 | 0.002 |
| Value-loss coefficient | | 0.5 | |
| Reward clipping | | None | |
| Primitive-action mask / RND | | None / off | |
| *Policy architecture* | | | |
| Hidden width | 512 | 512 | 256 |
| Temporal core: Craftax / Classic | None | GRU | `TSSSSS` |
| Temporal core: Quest-Hard | None | GRU | `STST` |
| GRU layers | – | 1 | – |
| Hybrid attention heads / memory | – | – | 8 / 128 |
| Quest glyph crop | | $9 \times 9$ | |
| Quest glyph embedding dimension | | 32 | |

Craftax and Craftax-Classic use their symbolic observations. Quest-Hard uses a glyph crop, status and message features, and the previous action. Here and below, `T` denotes an attention block and `S` an S5 block; patterns list blocks in execution order. Recurrent minibatches preserve environment sequences. For PPO on Classic, one policy supplies both the Diamond and Eat Plant measurements.

Table 12: PPO on MiniHack TreasureDash and ZombieHorde. Both environments share the optimization and rollout settings. The task rewards, encoder, and optimizer settings follow the SOL protocol (Henaff et al., 2025); collection uses the Craftax PPO batch layout, with standard PPO and GAE.

| Hyperparameter | FC | GRU | Hybrid |
|---|---|---|---|
| *Rollouts and optimization* | | | |
| Parallel environments | | 1,024 | |
| Rollout length | 64 | 64 | 128 |
| Transitions per rollout | 65,536 | 65,536 | 131,072 |
| Transitions per minibatch | 8,192 | 8,192 | 16,384 |
| Minibatches per epoch / epochs | | 8 / 1 | |
| Recurrent training length | – | 64 | 128 |
| Learning rate | | $10^{-4}$, constant | |
| Optimizer | | Adam, $(\beta_1, \beta_2) = (0.9, 0.999)$ | |
| Adam $\epsilon$ | | $10^{-7}$ | |
| Maximum gradient norm | | 4.0 | |
| *Returns and losses* | | | |
| Advantage estimator | | GAE, $\lambda = 1$ | |
| Discount $\gamma$ | | 0.99 | |
| Policy clipping | | 0.1; ratio interval $[0.9, 1.1]$ | |
| Value clipping | | 1.0 | |
| Entropy / value-loss coefficient | | 0.003 / 0.5 | |
| Task-reward scale | | 0.01 | |
| Reward clipping | | None | |
| Primitive-action mask / RND | | None / off | |
| *Policy architecture and task interface* | | | |
| Policy width: TreasureDash | 2,340 | 512 | 256 |
| Policy width: ZombieHorde | 2,341 | 512 | 256 |
| Temporal core | None | GRU | `TSSSSS` |
| GRU layers | – | 1 | – |
| Glyph crop | | $12 \times 12$ | |
| Glyph embedding dimension | | 128 | |
| Primitive actions: Treasure / Zombie | | 9 / 10 | |
| Episode limit: Treasure / Zombie | | 40 / 1,500 actions | |
| No-progress timeout | | 150 actions | |

The value $\lambda = 1$ is the selected GAE setting, not a GAE hyperparameter reported by SOL, whose learner uses V-trace. TreasureDash retains the task reward $20r_{\text{stairs}} + r_{\text{gold}}$ before scaling; ZombieHorde uses the task score reward. Additional option-reward shaping is not added to these PPO baselines. Reported TreasureDash scores and ZombieHorde kill counts are unscaled evaluation metrics. The five-seed cohorts use the collection settings above; earlier 32-environment runs are excluded from their reported aggregates.

### F.3 SCALAR HYPERPARAMETERS

Table 13: SCALAR on Craftax and Craftax-Classic. The architectures share the optimizer and curriculum settings except for the environment-specific reset probability and skill library. Classic trains a separate policy for each of Diamond and Eat Plant.

| Hyperparameter | FC | GRU | Hybrid |
|---|---|---|---|
| *Rollouts and optimization* | | | |
| Parallel environments | | 1,024 | |
| Rollout length | 64 | 128 | 128 |
| Transitions per rollout | 65,536 | 131,072 | 131,072 |
| Transitions per minibatch | 8,192 | 16,384 | 16,384 |
| Minibatches per epoch / epochs | | 8 / 4 | |
| Initial learning rate | | $2 \times 10^{-4}$ | |
| Learning-rate schedule | | Linear decay to zero; no warmup | |
| Optimizer / Adam $\epsilon$ | | Adam / $10^{-5}$ | |
| Maximum gradient norm | | 1.0 | |
| Discount $\gamma$ / GAE $\lambda$ | | 0.99 / 0.8 | |
| Policy clipping | | 0.2 | |
| Entropy / value-loss coefficient | | 0.01 / 0.5 | |
| *Architecture and skill learning* | | | |
| Expert width | 512 | 512 | 256 |
| Temporal core | None | GRU | TSSSSS |
| Actor LayerNorm | | Off | |
| Boundary bootstrap clamping | On | Off | Off |
| Per-skill balancing cap / threshold | | 4 / 64 | |
| Curriculum success threshold | | 0.3 | |
| Success EMA update weight | | 0.1 | |
| RND | | Off | |
| *Environment-specific settings* | | | |
| Practice-reset probability: Craftax | | 0.99 | |
| Practice-reset probability: Classic | | 0.9 | |
| Tasks / experts: Craftax | | 13 / 10 | |
| Tasks / experts: Classic Diamond | | 11 / 8 | |
| Tasks / experts: Classic Eat Plant | | 8 / 7 | |
| Primitive actions: Craftax / Classic | | 43 / 17 | |
| Practice-reset probability at evaluation | | 0 | |

SCALAR uses episode-termination-aware GAE and its prescribed skill composition. FC applies the boundary clamping described in Appendix G.1; GRU and Hybrid retain the recurrent continuation rule. Hybrid uses five S5 blocks, one attention block with memory 128, and GLU heads. The Craftax skill-sharing map appears in Appendix I.

Table 14: SCALAR with RND on MiniHack Quest-Hard, TreasureDash, and ZombieHorde. The learner and RND settings are shared across the three environments; training budgets and evaluation quotas are in Table 10.

| Hyperparameter | FC | GRU | Hybrid |
|---|---|---|---|
| *Rollouts and optimization* | | | |
| Parallel environments / rollout length | | 1,024 / 64 | |
| Transitions per rollout / minibatch | | 65,536 / 8,192 | |
| Minibatches per epoch / epochs | | 8 / 4 | |
| Initial learning rate | | $2 \times 10^{-4}$; linear decay to zero | |
| Optimizer / Adam $\epsilon$ | | Adam / $10^{-5}$ | |
| Maximum gradient norm | | 1.0 | |
| Discount $\gamma$ / GAE $\lambda$ | | 0.99 / 0.8 | |
| Policy clipping | | 0.2 | |
| Entropy / value-loss coefficient | | 0.01 / 0.5 | |
| *Architecture and curriculum* | | | |
| Hidden width | 512 | 512 | 256 |
| Temporal core | None | GRU | `SSTT` |
| Per-skill balancing cap / threshold | | 4 / 64 | |
| Practice-reset probability | | 0.9 | |
| Progressive curriculum threshold | | 0.6 | |
| Success EMA update weight | | 0.1 | |
| Practice resets at evaluation | | Off | |
| *Random network distillation* | | | |
| Predictor hidden layers / width | | 3 / 256 | |
| RND output dimension | | 512 | |
| Predictor learning rate | | $3 \times 10^{-4}$ | |
| Predictor epochs per update | | 4 | |
| RND warmup | | 20 learner updates | |
| Intrinsic reward coefficient | | 1.0 | |
| Predictor-loss / intrinsic-advantage coefficient | | 0.01 / 0.01 | |
| Glyph input scale for RND | | 0.0081994645 | |
| Intrinsic reward normalization / episodic RND | | Off / off | |
| *Environment-specific skill composition* | | | |
| Tasks / experts: Quest-Hard | | 7 / 7 | |
| Tasks / experts: TreasureDash | | 10 / 2 | |
| Tasks / experts: ZombieHorde | | 62 / 4 | |
| Primitive actions: Quest / Treasure / Zombie | | 20 / 9 / 10 | |

Quest-Hard uses the source-derived 20-action interface with a 1,500-game-tick episode limit. TreasureDash and ZombieHorde retain the respective 40- and 1,500-action limits. Practice-reset replay is excluded from the new-action training budget. RND warmup is separate from the policy learning-rate schedule, which has no warmup.

### F.4 PGOC HYPERPARAMETERS

Table 15: PGOC on Craftax, Craftax-Classic, and MiniHack. The learner, architecture, and controller settings are shared across environments; the per-environment skill library, action space, and (Table 10) budget and evaluation quota differ. Each ablation changes only its stated mask, continuation target, selector, or reward component.

| Hyperparameter | FC | GRU | Hybrid |
|---|---|---|---|
| *Rollouts and optimization* | | | |
| Parallel environments / rollout length | | 1,024 / 128 | |
| Transitions per rollout / minibatch | | 131,072 / 16,384 | |
| Minibatches per epoch / epochs | | 8 / 4 | |
| Initial learning rate | | $2 \times 10^{-4}$; linear decay to zero | |
| Optimizer | | Adam, $(\beta_1, \beta_2) = (0.9, 0.999)$ | |
| Adam $\epsilon$ / maximum gradient norm | | $10^{-5}$ / 1.0 | |
| Discount $\gamma$ | | 0.999 | |
| Skill GAE $\lambda$ | | 0.8 | |
| Policy / value clipping | | 0.2 / 0.2 | |
| Skill entropy / value-loss coefficient | | 0.002 / 0.5 | |
| Reward clipping | | None | |
| *Architecture and controller* | | | |
| Hidden width | | 256 | |
| Temporal core | None | GRU | `TSSSSS` |
| Hybrid attention heads / memory | – | – | 8 / 128 |
| Actor LayerNorm / $\epsilon$ | | On / $10^{-5}$ | |
| Controller policy / value-loss coefficient | | 1.0 / 0.5 | |
| Controller entropy coefficient | | 0.01 | |
| Controller hidden normalization | | Non-affine RMS, $\epsilon = 10^{-6}$ | |
| Controller logit bound | | Off | |
| Controller GAE $\lambda_\mu$ | | 0.95 | |
| Minimum option sample fraction | | 0.005 | |
| Boundary continuation | | Controller value | |
| Controller reward gating | | Successful option completion | |
| Option-selection bonus / history window | | 1.0 / 50 learner updates | |
| *Environment and initialization* | | | |
| Skill library | | Original SCALAR library | |
| Prerequisite / completed-option masks | | On / on | |
| Practice resets / RND | | Off / off | |
| Pretrained initialization / warmup | | None / none | |
| Tasks / experts: Craftax | | 13 / 10 | |
| Tasks / experts: Classic Diamond | | 11 / 8 | |
| Tasks / experts: Classic Eat Plant | | 8 / 7 | |
| Tasks / experts: Quest-Hard | | 7 / 7 | |
| Tasks / experts: TreasureDash | | 10 / 2 | |
| Tasks / experts: ZombieHorde | | 62 / 4 | |
| Primitive actions: Craftax / Classic | | 43 / 17 | |
| Primitive actions: Quest / Treasure / Zombie | | 20 / 9 / 10 | |

The skill library and expert-sharing map are in Appendix I. Controller rewards and the option-selection exploration bonus are defined in Appendix G. Fixed-order and uniform-selector ablations retain the same skill learner; they replace the selector rather than training a separate option-selection policy. The prescribed-order control uses controller entropy coefficient $0.03$ (the learned rows above use $0.01$); because the prescribed selector bypasses the controller's option-policy logits, this coefficient does not drive its selection.

**Parameter counts.** The ten shared experts and the controller are heads on a single network, so the hierarchical model adds the controller policy and value heads and the per-expert output heads

to the shared trunk. These heads account for a small fraction of the total parameters, which are dominated by the shared encoder and temporal core common to PPO, SCALAR, and PGOC at the same backbone width (Table 15); the PGOC differences over the baselines are therefore unlikely to arise from an increased parameter count.

### F.5 SOL AND HBS TRANSFER HYPERPARAMETERS

Table 16: SOL and HBS transfer pilots on Craftax and Craftax-Classic. These settings apply to the single-seed, two-reward experiments in Appendix B.6, separately from the main five-seed comparisons.

| Hyperparameter | SOL | HBS |
|---|---|---|
| *Rollouts and optimization* | | |
| Parallel environments | 1,024 | |
| Rollout / recurrent training length | 256 / 256 | |
| Trajectories per minibatch | 128 | |
| Sequence positions per minibatch | 32,768 | |
| Optimization epochs | 1 | |
| Actor groups / rollout policy lag | 2 / 1 rollout | |
| Learning rate | $10^{-4}$, constant | |
| Optimizer | Adam, $(\beta_1, \beta_2) = (0.9, 0.999)$ | |
| Adam $\epsilon$ / maximum gradient norm | $10^{-7}$ / 4.0 | |
| *Returns and losses* | | |
| Return estimator | V-trace | |
| Discount $\gamma$ / V-trace $(\bar{\rho}, \bar{c})$ | 0.999 / $(1, 1)$ | |
| Policy-ratio clipping interval | $[1/1.1,\ 1.1]$ | |
| Value clipping | 1.0 | |
| Base entropy / value-loss coefficient | 0.003 / 0.5 | |
| Controller entropy multiplier | 10 | 1 |
| Likelihood and entropy | Active action heads only | |
| Reward scale / RND | 1.0 / off | |
| *Architecture and hierarchy* | | |
| Encoder / temporal core | ReLU MLP, 512 / LSTM, 512 | |
| Worker reward channels | New achievements; signed survival changes | |
| Controller reward | New-achievement count | |
| Worker reward selection | One of two channels | Weighted combination |
| Coefficient values | – | $\{0, 0.5, 1\}$ per channel |
| All-zero coefficient fallback | – | $(0.5, 0.5)$ |
| Coefficient normalization | – | None |
| Available option durations | $\{1, 2, 4, 8, 16, 32, 64, 128\}$ | |
| Primitive actions: Craftax / Classic | 43 / 17 | |

The two worker rewards are defined in Appendix B.6. Controller decisions occupy sequence positions but do not count as actual game actions toward the 1B training budget. Thus sequence positions in this table are distinct from environment transitions in the PPO and SCALAR tables.

# G CRAFTAX INSTANTIATION

**Instantiation in Craftax.** The resource-budget component of the Craftax implementation is an instance of this formulation. For resource $j$, the target $B_j$ is a specified quantity and the observed contribution is $q_{t,j} = [n_j(z_{t+1}) - n_j(z_t)]_+$, where $n_j$ is its inventory count. This measures acquired quantities rather than the number of option completions. Spending a resource does not decrease its credited progress or renew its collection reward; current inventory requirements are evaluated separately through the symbolic predicates. These updates apply to transitions within an episode, and the progress records reset at episode boundaries.

**Controller rewards.** The skill library uses the nine capped progress channels specified in Appendix I, with total credit at most 32. The plan callback zeros its emitted controller reward on transitions that it marks as terminal.

**Option exploration bonus.** All native PGOC variants add a unit selection bonus to the emitted controller reward. At each episode start, an option is marked for exploration with probability one minus its recent success fraction; with no recorded outcomes this probability is one. The fraction uses the pooled option successes and unsuccessful episode ends from the previous 50 learner updates. The first selection of each marked option in the episode receives the bonus, and later selections do not. This adds at most one unit per option per episode: 13 for the 13-option library. The bonus is separate from the capped plan-progress component and is included in the controller-reward balance used for success-gated skill credit.

The dungeon example in Figure 1 illustrates the distinction between completion, progress, and availability. It uses a one-defeat completion predicate to illustrate the general formulation; the experimental combat option instead completes at the eight-defeat threshold. In this base Craftax example (Matthews et al., 2024), $\Delta\mathrm{kills}$ counts floor-one defeats and $\Delta\mathrm{vitals}$ sums changes in health, food, drink, and energy; the rendered state and transition are illustrative, not a measured rollout. Starting from six defeated monsters, one further defeat satisfies the combat completion predicate and advances credited progress to seven of eight. Combat remains needed, so the controller may select it again; descent is still masked. In this base option specification, reaching eight defeats exhausts the combat progress reward and makes its continued-need predicate false, while satisfying the defeat prerequisite for descent. Additional defeats can still satisfy the combat completion predicate without earning further progress credit. The tests answer different questions: whether an execution succeeded, how much of its effect remains rewardable, and which options should be available at the next decision.

**Empty option frontiers and diagnostic interpretation.** The implementation can reach a state in which no option's initiation and continued-need predicates both hold — for example, when inventory thresholds for the remaining options are unmet — leaving no eligible option. Eligibility is evaluated from these predicates alone and does not consult the credited-progress records (Appendix I). For this boundary case, the reference sampler subtracts $10^9$ from every option logit and samples the resulting categorical distribution. Thus the intended restriction to eligible choices requires a nonempty set; the fallback does not enforce eligibility for an empty set. In the frozen Craftax-Hybrid controls this case is reached at 6,641 of 171,410 option-selection events (3.87%), a final-policy frequency rather than a training-time rate. An option can also continue after its initiation conditions cease to hold, because initiation is checked at selection rather than at every primitive step. Per-option frame allocations therefore describe which policy executed and do not establish that its prerequisites held throughout execution. Checkpoint frontier counts are state snapshots, not estimates of training-time occupancy or causal evidence for a performance difference.

## G.1 SCALAR BOUNDARY CLAMPING

Equations 9 and 10 describe the episode-level GAE used by the recurrent SCALAR implementations. The feedforward implementation additionally clamps the next value and the propagated advantage to be nonnegative at successful, nonterminal skill boundaries. Its boundary target therefore replaces $V^{\mathrm{S}}_{o_{t+1}}(h_{t+1})$ and $\hat{A}^{\mathrm{S}}_{t+1}$ in Equation 11 with their positive parts. It still propagates the subsequent skill's advantage; this differs from PGOC's zero trace carry and controller-value bootstrap at a successful boundary.

# H MiniHack Environment Details

The three MiniHack tasks share a common observation interface and differ in their action spaces, horizons, and reported metrics. All three are MiniHack tasks running under the NetHack Learning Environment; the TreasureDash and ZombieHorde layouts and task rules follow the released configurations of Henaff et al. (2025). Policy observations consist of a glyph crop centered on the agent, bottom-line status features, the recent message, and the previous action. The crop size and embedding dimension are given per method in Tables 11–14; SOL and HBS retain the released $12 \times 12$ crop and 64-dimensional glyph embeddings (Appendix F.5).

**Quest-Hard.** The task follows the source-derived interface used by SCALAR's Quest experiments (Zabounidis et al., 2026): 20 primitive actions and a 1,500-game-tick episode limit. The reported metric is the percentage of frozen evaluation episodes that complete the quest. Policies act from a $9 \times 9$ glyph crop with 32-dimensional glyph embeddings. SCALAR uses 7 tasks with 7 experts; its intermediate-state practice resets do not count toward the 1B-action training budget and are disabled during frozen evaluation.

**TreasureDash.** The task retains the released action and reward rules of Henaff et al. (2025): 9 primitive actions and a 40-action horizon, and the reported metric is the mean unscaled episode score. An arrival at the staircase on the final (action-40) transition earns the staircase reward but is assigned a timeout end status by the stock engine rather than task-success termination. For the five-seed SOL/HBS results, workers receive $20\, r_{\mathrm{stairs}}$ or $r_{\mathrm{gold}}$ and the controller receives $0.001$ times their sum. PPO and SCALAR use the task reward without additional option shaping. A 150-action no-progress timeout applies within the episode. The main comparison uses the randomized floor-geometry variant: the start column (seven values) and coin spacing (three values) are drawn per episode, giving 21 distinct layouts, with the staircase at a fixed column. Training and evaluation sample from the same layout distribution; the staircase option receives a $0.1$ distance-shaping coefficient toward it.

**ZombieHorde.** The task uses 10 primitive actions and a 1,500-action horizon with the same 150-action no-progress timeout. The reported metric is the mean number of zombies killed per frozen episode, following Henaff et al. (2025). SOL/HBS workers receive the task score or $20\, \Delta\mathrm{health}$ and the controller receives $0.001$ times the task score. SCALAR uses 62 tasks shared across 4 experts. An independent count of kill messages differs slightly from the score-qualified counter, which omits kill-bearing death actions; all reported figures use the score-qualified definition.

**Budgets.** Training budgets count consumed policy actions rather than game ticks. For SOL and HBS this counter includes controller-only decisions, which occupy sequence positions without stepping the game; actual game actions are logged separately. Budgets are rounded to complete training batches (Table 10).

# I OPTION LIBRARIES

PGOC uses the SCALAR skill specifications and expert-sharing map while retaining its learned option controller. This section gives the option library for each Craftax environment: the main Craftax progression below, and the two Craftax-Classic progressions in Section I.2.

## I.1 CRAFTAX

The main Craftax library has thirteen options and ten shared actor–critic experts. Table 17 lists them in order of their task indices, with the option-to-expert map $(0, 1, 0, 2, 3, 4, 0, 2, 5, 6, 7, 8, 9)$: repeated gathering options share experts but remain separate controller choices.

Table 17: The thirteen-option library in task-index order, with the initiation and continued-need predicates evaluated on the symbolic state. `surface` holds on dungeon level zero; `table-wood` holds with a crafting table or at least three wood; `table-stone` requires `table-wood` and at least one stone; `pick` requires a wood pickaxe; `weapons` is `pick` with fewer than two swords; `arrows` requires two swords and fewer than sixteen arrows. Progress weights and caps are controller-side credit limits per episode (see text); eligibility does not depend on them.

| # | Completion | Initiation | Continued need | Expert | Weight | Cap |
|---|---|---|---|---|---|---|
| 0 | 3 wood | `surface`, no pickaxe | wood $< 3$ | 0 | 0.25 | 15 |
| 1 | wood pickaxe | `surface`, `table-wood` | no pickaxe | 1 | 1 | — |
| 2 | 10 wood | `surface`, `weapons` | wood $< 10$ | 0 | 0.25 | (15) |
| 3 | 8 stone | `surface`, `weapons` | stone $< 8$ | 2 | 0.25 | 9 |
| 4 | wood sword | `surface`, `pick`, `table-wood` | no sword | 3 | 1 | — |
| 5 | stone sword | `surface`, `pick`, sword, `table-stone` | one sword | 4 | 1 | — |
| 6 | 10 wood | `surface`, `arrows` | wood $< 10$ | 0 | 0.25 | (15) |
| 7 | 8 stone | `surface`, `arrows`, `pick` | stone $< 8$ | 2 | 0.25 | (9) |
| 8 | 16 arrows | `surface`, 2 swords, `table-stone` | arrows $< 16$ | 5 | 0.25 | 16 |
| 9 | dungeon entry | `surface`, 2 swords, 16 arrows | not entered | 6 | 5 | — |
| 10 | bow | dungeon floor | no bow | 7 | 1 | — |
| 11 | 8 floor defeats | dungeon floor, bow | defeats $< 8$ | 8 | 1 | — |
| 12 | Mines entry | dungeon floor, 8 defeats | not entered | 9 | 5 | — |

Two properties of these predicates matter for the learned controller. Initiation permits the canonical SCALAR skill sequence without imposing a sequential cursor: wood and stone gathering remain selectable whenever their inventory thresholds hold, so crafting that consumes inventory re-enables replenishment, and gathering for ammunition (indices 6–8) can be interleaved with the weapon prerequisites that precede it. Eligibility is evaluated from the symbolic state alone and does not consult the credited-progress records, so exhausting a capped progress channel does not remove the corresponding option from the admissible set; continued need is what removes it, driven by the uncapped inventory thresholds in the fourth column.

**Why the resource caps exceed recipe costs.** The progress caps on wood, stone, and arrows bound controller credit, not availability. The wood cap of 15 covers the canonical sequence's total acquisition — three units toward the pickaxe, ten toward the weapons, and ten toward the arrows — with additional budget for replenishment after crafting consumption; it is not a minimum recipe cost. The stone cap of 9 covers the two eight-stone gathering options against consumption by the stone sword and arrows, and the arrow cap of 16 matches the single ammunition requirement exactly. Spending resources does not renew these budgets.

**Local objectives.** We retain the inventory and achievement completion predicates, sparse resource or equipment deltas, and local reward shaping of SCALAR. The vital term is $0.1$ times the sum of health, food, drink, and energy deltas, with each delta truncated toward zero before summation; the lava and death terms are retained. Where specified by the skill, distance shaping gives $0.01$ times the reduction in Euclidean distance to a remembered target, only when the sparse delta is zero; its 400-integer relative-position memory is reward state and is not supplied as an additional policy observation.

**Controller credit.** Positive acquisition increments for wood, stone, and arrows carry weight $0.25$ under the caps above; one pickaxe tier, two sword tiers, one bow, and the eight dungeon-floor defeats carry unit weight; dungeon and Mines entry carry weight five. The total controller progress credit is bounded by 32 per episode; the canonical sequence reaches $0.25 \times (15+9+16) = 10$ from resources, 12 from the four equipment milestones and eight defeats, and 10 from the two entries, so the bound is met exactly when all channels are exhausted. This bound concerns the progress component; the separate option exploration bonus described in Appendix G adds at most 13 further units per episode. The skill objective retains PGOC's success-gated controller credit and controller-value boundary target; local terminal penalties retain the SCALAR rules.

**Protocol.** We evaluate FC, GRU, and Hybrid variants using seeds 3–7, the same nominal 1B-interaction budget, and 10,240 fresh frozen-evaluation episodes per trained policy. The variants retain PGOC learner settings, including discount $0.999$, skill entropy coefficient $0.002$, and expert width 256. There are 43 permitted primitive actions; four padding logits are always masked. Training uses fresh starts without SCALAR curriculum resets, and the asynchronous fresh-world generator is enabled; the SCALAR controls retain their $0.99$ training reset probability, and all evaluations disable curriculum resets. The comparison holds the skill definitions and sharing map fixed, but it still differs in controller, continuation target, training resets, and selected learner settings, so it does not isolate learned option selection on its own.

### I.2 Craftax-Classic

Craftax-Classic uses separate, smaller libraries for its two goal progressions rather than the thirteen-option main library. Both are generated from the supplied Classic skill operators by the planner, which folds ephemeral prerequisites (placing a crafting table or furnace) into the persistent skills that require them, so the libraries below contain only the persistent options.

**Diamond.** The Diamond library has eleven options and eight experts, with option-to-expert map $(0, 1, 2, 0, 3, 4, 0, 2, 5, 6, 7)$. Table 18 lists them in plan order. The progression is a single chain with no unordered pair: each option's prerequisite is the equipment produced two steps earlier, so the planner's sequence is the only valid order, and the learned controller's role is to value completions and recover from consumption and interruption rather than to reorder skills. Gathering wood repeats at indices 0, 3, and 6 and gathering stone at indices 2 and 7, each pair sharing one expert.

Table 18: The Craftax-Classic Diamond library in plan order. Prerequisites are evaluated against inventory and placed stations; the ephemeral station-placement operators are folded into the skills that require them (see text). Quantities are the planner's completion targets.

| # | Completion requirement | Prerequisite | Expert | Qty |
|---|---|---|---|---|
| 0 | 3 wood | — | 0 | 3 |
| 1 | wood pickaxe | 3 wood | 1 | 1 |
| 2 | 2 stone | wood pickaxe | 2 | 2 |
| 3 | 3 wood | — | 0 | 3 |
| 4 | stone pickaxe | 3 wood, 1 stone | 3 | 1 |
| 5 | 1 coal | stone pickaxe | 4 | 1 |
| 6 | 3 wood | — | 0 | 3 |
| 7 | 2 stone | wood pickaxe | 2 | 2 |
| 8 | 1 iron | stone pickaxe | 5 | 1 |
| 9 | iron pickaxe | 3 wood, 2 stone, 1 coal, 1 iron | 6 | 1 |
| 10 | 1 diamond | iron pickaxe | 7 | 1 |

**Eat Plant.** The Eat Plant library has eight options and seven experts; both wood-gathering options reuse one expert. Table 19 lists them in plan order. Unlike the Diamond chain, this progression contains a genuine scheduling choice: the planted crop takes 600 game steps to ripen, and placing it first lets preparation proceed while it grows, so the fixed planner sequence (plant, then prepare, then eat) is also efficient. The preparation requirements on the final option — four stone and a wooden sword — are preparation choices rather than planting mechanics. Place Plant's completion requires both sapling expenditure and a newly tracked plant coordinate, so walking toward an existing plant

does not count; the tracked site persists while the agent leaves to prepare, and Eat Plant completes on the first eat achievement at a ripe site.

Table 19: The Craftax-Classic Eat Plant library in plan order. The final option's stone and sword requirements are preparation choices, not planting mechanics (see text).

| # | Completion requirement | Prerequisite | Expert | Qty |
|---|---|---|---|---|
| 0 | 1 sapling | — | 4 | 1 |
| 1 | place plant | 1 sapling | 5 | 1 |
| 2 | 3 wood | — | 0 | 3 |
| 3 | wood pickaxe | 3 wood | 1 | 1 |
| 4 | 4 stone | wood pickaxe | 3 | 4 |
| 5 | 3 wood | — | 0 | 3 |
| 6 | wood sword | 3 wood | 2 | 1 |
| 7 | eat plant | ripe tracked plant, 4 stone, wood sword | 6 | 1 |

Both Classic libraries use 17 permitted primitive actions and the vital term of $0.1$ times the health delta described in Section I.1.

### I.3 MiniHack

The MiniHack experiments use SCALAR's native task decomposition of each environment rather than the Craftax libraries above.

**TreasureDash.** The library has ten tasks over two experts: nine gold-collection tasks with inventory thresholds 1 through 9 routed to a `collect_gold` expert, followed by a staircase task routed to a `reach_downstairs` expert. Gold tasks reward the change in inventory gold; the staircase task rewards the staircase signal. The task reward of Section H ($20\, r_{\text{stairs}} + r_{\text{gold}}$) is the sum these tasks decompose.

**ZombieHorde.** The library has 62 tasks over four experts, built from a repeating cycle per zombie $n \in \{1, \ldots, 16\}$: approach (`approach_zombie`, completing on adjacency) and defeat (`defeat_zombie`, completing at $n$ cumulative kills), followed for $n < 16$ by retreat (`retreat_to_starting_temple`) and recover (`recover_to_full_health`, completing at full health inside the temple); the tail is omitted after the sixteenth kill, giving $16 \times 2 + 15 \times 2 = 62$ tasks. Combat tasks reward the task-score delta; the retreat and recovery tasks reward the health delta, so the decomposition interleaves fight and survival objectives under the same expert-sharing discipline as the Craftax libraries.

**Quest-Hard.** Quest-Hard uses the source-derived interface with seven tasks and seven experts over the 20-action space (Table 14), following the equipment, crossing, and combat progression of the underlying task; its option semantics follow the source's milestone curriculum rather than the explicit threshold sequences above.

# J SOL AND HBS ADAPTATION

SOL and HBS are hierarchical baselines that coordinate separately specified worker rewards without symbolic prerequisites. Both were released for MiniHack; neither has a published Craftax or Craftax-Classic configuration, and HBS has no task-specific configuration for the MiniHack tasks we use. This section records the per-environment adaptation; the resulting numbers are in Table 1 and Appendix B.6.

**Native port of the released learner.** All environments share one ported learner. The released SOL and HBS `Learner.train` (Henaff et al., 2025; Matthews et al., 2026) serves as the numerical oracle, checked on fresh, partly stale, and entirely stale trajectories and on full 32,768-transition batches before any training; no numerical tolerance was loosened. This is a validated reimplementation with declared task transfers, not a claim that each task reproduces a published experiment. Table 20 summarizes the adaptation per environment.

Table 20: Per-environment SOL/HBS adaptation. "Task config" is the source of the environment and task settings; "reward" is how the worker/controller rewards are assigned. All use the same validated native learner and the released MiniHack geometry where one exists.

| Environment | Task config | Worker / controller reward |
|---|---|---|
| Craftax | transfer (none released) | two-reward / achievement |
| Craftax-Classic | transfer (none released) | two-reward / achievement |
| TreasureDash | released SOL | $20\,r_{\text{stairs}}, r_{\text{gold}}$ / $0.001\times$sum |
| ZombieHorde | released SOL | score,$20\,\Delta$health / $0.001\times$score |
| Quest-Hard | Quest20 transfer | milestone / milestone |

**Craftax and Craftax-Classic.** No SOL/HBS task configuration exists, so we supply two worker rewards — unweighted new-achievement count, and a signed survival term $\Delta\text{health} + 0.33(\Delta\text{food} + \Delta\text{water} + \Delta\text{energy})$ — and give the controller the achievement reward. SOL selects one worker reward; HBS applies its unnormalized two-coefficient mixing rule in $\{0, 0.5, 1\}$ with the all-zero choice mapped to $(0.5, 0.5)$. The two-reward definition and full settings are in Appendices B.6 and F.5. These pilots improve their environment return during training but rarely complete the target milestones, so increasing return alone does not establish long-horizon progression. Table 1 reports the multi-seed comparisons.

**TreasureDash and ZombieHorde.** These use the released SOL task configuration unchanged: 1,920 parallel environments, 256-step rollouts, a 512-unit LSTM, 64-dimensional glyph embeddings, a $12 \times 12$ crop, and 32,768-transition minibatches with one optimization epoch and no shuffling, with the released V-trace and clipping settings. Workers receive $20\,r_{\text{stairs}}$ or $r_{\text{gold}}$ (TreasureDash) and task score or $20\,\Delta\text{health}$ (ZombieHorde); the controller receives $0.001$ times the corresponding sum. HBS has no released task-specific configuration, so it transfers its released unnormalized mixing rule to the released SOL task settings. No RND or curriculum resets are used. Table 1 uses randomized TreasureDash geometry for every method; the original fixed-layout pilots are reported in Appendix B.6.

**Quest-Hard.** The Quest transfer uses the Quest20 interface and seven milestone rewards; Table 1 reports its completed seeds.

**Comparison to PGOC.** SOL and HBS coordinate worker rewards but use no symbolic prerequisites, option masks, or controller-value boundary bootstrap. Their comparison to PGOC tests whether PGOC's gains require its specific use of the supplied specifications. The baselines are functional learners: each pilot improves its own environment return during training, even where the target milestones are not reached (Appendix B.6). The gap therefore reflects the method, not a failed port.

# K Native Execution and Asynchronous World Generation

The rerun executes independent single-GPU jobs on NVIDIA L40S workers using Ray for scheduling. Each job reserves one GPU and sixteen CPU cores: eight environment threads and eight background world generators. The model, rollout inference, returns and loss, backward pass, and Adam update use the method's native PufferLib C/CUDA backend. Reference implementations are used for numerical validation. Before long training, each configuration passes checks of forward evaluation, returns and loss, gradients, optimizer state transitions, and a complete update, as well as checkpoint continuation and frozen evaluation.

**Fresh-world stream.** Fresh resets consume an ordered stream of newly generated Craftax worlds from a queue with capacity 256. Eight background generators refill this queue while the learner and environment stepping continue. The queue is not a finite set of worlds recycled across episodes. The production runs record the stream version, capacity, and worker count; checkpoint continuation preserves the stream's seed and consumed position. For each admitted configuration, paired synchronous and asynchronous generation using the same ordered stream agree exactly on the checked final learner and continuation states. SCALAR's intermediate-state curriculum resets remain separate from fresh-world generation and are disabled in frozen evaluation.

**Compute.** All runs use NVIDIA L40S GPUs with one GPU and sixteen CPU cores per job. At the admitted asynchronous configurations, the Craftax methods sustain roughly 72k (SCALAR/PGOC-Hybrid) to 161k (PGOC-FC) primitive frames per second (Table 23), so one 1B-frame Craftax training run takes on the order of two to four hours on a single GPU. The 110-policy Craftax cohort (45 main and 65 ablation pairs) therefore consumes on the order of a few hundred GPU-hours in total, including frozen evaluation. MiniHack workloads are slower per transition: the Quest-Hard runs sustain roughly 9–13k actions per second, so the 1B-action MiniHack runs account for the majority of the total compute.

**Prelaunch throughput checks.** Table 23 reports both PGOC libraries and the PPO/SCALAR controls. Each configuration is timed on a Ray worker at its production rollout and minibatch geometry, with 1,024 environments and the selected learner settings. The paired checks execute 24 updates, exclude the first four as warmup, and divide the remaining primitive environment frames by the elapsed update time. Timed updates include rollout collection, learning, and shared logger calls. The synchronous condition generates the same ordered worlds on demand; the asynchronous condition uses the eight background generators. The checks also verify consumption beyond the initial queue capacity and exact checkpoint replay. These are bounded prelaunch measurements, not full-training averages; throughput can shift during learning as episode lengths and reset frequency change. All long runs use the admitted asynchronous configuration.

**Frozen evaluation.** Each final policy is evaluated on 10,240 fresh, complete episodes, with ten episodes assigned to each of 1,024 environment slots and no artificial step cutoff. The evaluation collector is checked against the training policy's trajectories, and policy and optimizer states remain unchanged during evaluation. The asynchronous fresh-world stream remains enabled. Training seeds are independent; reported outcome means and sample standard deviations give the five training seeds equal weight.

### K.1 Confound and Selector Controls

These controls complete the matched comparisons referenced in Sections 4 and 4.4. Each new policy trains on the full budget and evaluates on frozen episodes (10,240 for Craftax, 1,024 for TreasureDash). Means and sample standard deviations are over the training seeds 3–7; the Seeds column gives the number of completed seeds where one remains in training.

Table 21: Craftax-Hybrid confound controls (frozen Mines success, %). The released learned PGOC control and the released SCALAR-Hybrid control ($\gamma = 0.99$, reset 0.99) are the references. The $\gamma$ and reset rows isolate the discount and practice-reset confounds; the SCALAR-style estimator rows retain the PGOC scaffold and change both the estimator and the delayed controller-credit payment.

| Condition | Seeds | Mines (%) |
|---|---|---|
| Released learned PGOC (reference) | 5 | $62.80 \pm 1.86$ |
| PGOC, matched prescribed selector | 5 | $69.38 \pm 1.19$ |
| PGOC scaffold, SCALAR-style estimator, prescribed | 5 | $12.77 \pm 2.70$ |
| PGOC scaffold, SCALAR-style estimator, learned | 5 | $6.12 \pm 9.59$ |
| PGOC, no success gate | 5 | $56.38 \pm 1.31$ |
| PGOC, no positive boundary-value clipping | 5 | $62.16 \pm 1.94$ |
| PGOC, no exploration bonus | 5 | $60.42 \pm 1.91$ |
| PGOC, $\gamma = 0.99$ | 5 | $34.70 \pm 1.50$ |
| Released SCALAR-Hybrid ($\gamma = 0.99$, reset 0.99) | 5 | $7.66 \pm 0.39$ |
| Native SCALAR, $\gamma = 0.999$, reset 0.99 | 5 | $3.60 \pm 0.55$ |
| Native SCALAR, $\gamma = 0.99$, reset 0 | 5 | $0.99 \pm 0.60$ |
| Native SCALAR, $\gamma = 0.999$, reset 0 | 4 | $1.28 \pm 1.20$ |
| Native SCALAR + PGOC masks/prescribed ($\gamma = 0.99$, reset 0.99) | 5 | $6.82 \pm 1.03$ |

Table 22: TreasureDash-Hybrid selector controls (frozen mean episode return). The frozen-skill rows train only the selector on one predeclared audited released seed-3 skill bank.

| Condition | Seeds | Mean return |
|---|---|---|
| Released learned PGOC (reference) | 5 | $22.21 \pm 0.03$ |
| Learned selector on common frozen skills | 5 | $22.29 \pm 0.18$ |
| Prescribed selector on common frozen skills | 5 | $10.88 \pm 0.16$ |
| Uniform selector on common frozen skills | 5 | $13.31 \pm 0.11$ |

### K.2 Frozen-Skill Controls

The frozen-skill rows hold one predeclared audited seed-3 skill bank fixed and train only the selector. Two things follow. First, the learned selector on frozen skills reaches 22.29, statistically indistinguishable from the released jointly trained 22.21 — so the performance is not an artifact of co-adapting the skills to a particular selector during training; the selector alone reproduces it. Second, the frozen prescribed (10.88) and uniform (13.31) controls still collapse, confirming that the failure of the fixed selectors is a property of the selection rule rather than of how the skills were trained. The learned ordering, not the skill bank, is what recovers the geometry.

### K.3 Estimator and Confound Rows

The SCALAR-style estimator rows change the estimator and the delayed controller-credit payment together, so they are not a bootstrap-only intervention and not a native SCALAR run. The $\gamma$/reset factorial sharpens the discount reading in the main text: the released SCALAR-Hybrid ($\gamma = 0.99$, reset 0.99) reaches 7.66%, and every native SCALAR cell stays at or below 3.60% regardless of discount or resets, while PGOC at $\gamma = 0.99$ still reaches 34.70%. The discount explains part of PGOC's advantage over the released control but does not close the gap between the two estimators under matched settings.

| Configuration | Sync (kSPS) | Async (kSPS) | Ratio |
|---|---|---|---|
| PPO-FC | 86.2 | 142.9 | 1.66× |
| PPO-RNN | 84.4 | 138.1 | 1.64× |
| PPO-Hybrid | 66.1 | 95.4 | 1.44× |
| SCALAR-FC | 85.8 | 96.5 | 1.12× |
| SCALAR-RNN | 80.4 | 93.8 | 1.17× |
| SCALAR-Hybrid | 65.8 | 72.5 | 1.10× |
| PGOC-FC | 91.7 | 161.3 | 1.76× |
| PGOC-GRU | 83.5 | 135.0 | 1.62× |
| PGOC-Hybrid | 54.2 | 72.9 | 1.34× |

Table 23: Prelaunch Ray L40S throughput checks at production geometry. kSPS denotes thousands of primitive environment frames per second. Each condition measures twenty updates after four warmup updates; these are bounded validation timings, not full-training averages. The paired conditions use the same reset stream and agree on checked final states.