CRISP: Curriculum-Inducing Primitive Informed Subgoal Prediction for Boosting Hierarchical Reinforcement Learning

Utsav Singh[1], **Vinay P. Namboodiri**[2]

[1]Department of Computer Science and Engineering, IIT Kanpur, India
[2]Department of Computer Science, University of Bath, UK
utsavz@iitk.ac.in, vpn22@bath.ac.uk

Abstract

Hierarchical reinforcement learning (HRL) leverages temporal abstraction to efficiently tackle complex long-horizon tasks. However, HRL often collapses because the low-level primitive's continual updates make earlier sub-goals issued by the high-level policy obsolete, introducing non-stationarity that destabilizes training. We propose CRISP, a curriculum-driven framework that tackles this instability with three key ingredients: (1) primitive-informed parsing (PIP), which adaptively re-labels a handful of expert demonstrations to always generate reachable subgoals by the current low-level primitive; (2) an inverse-reinforcement-learning regularizer that steers the high-level policy toward the expert-induced subgoal distribution and stabilizes learning; and (3) a unified training loop that leverages these components to boost sample efficiency. Across six sparse-reward robotic navigation and manipulation benchmarks, CRISP improves success rates by more than 40% over strong hierarchical and flat baselines and successfully transfers to real-world tasks, demonstrating the promise of curriculum-based HRL for practical scenarios.

1 Introduction

While reinforcement learning (RL) has demonstrated remarkable successes in continuous control domains such as robotic manipulation (Levine et al. 2015; Vecerík et al. 2017), tackling long-horizon continuous control tasks characterized by sparse rewards, inefficient exploration, and difficult credit assignment (Nachum et al. 2019; Kulkarni et al. 2016; Andrychowicz et al. 2017) remains a significant hurdle. Hierarchical reinforcement learning (HRL) offers a promising paradigm to address these challenges by decomposing tasks into subtasks via temporal abstraction, enabling efficient exploration (Dayan and Hinton 1993; Sutton, Precup, and Singh 1999; Parr and Russell 1998; Nachum et al. 2019). Several goal-conditioned HRL frameworks employ a high-level policy that proposes subgoals and a lower-level primitive that executes actions to achieve those subgoals (Nachum et al. 2018; Vezhnevets et al. 2017; Levy, Jr., and Saenko 2017).

HRL Challenges. Despite these benefits, HRL approaches suffer from the issue of non-stationarity. Specifically, as the higher-level policy predicts subgoals, the lower-level primitive behavior evolves during training, causing the higher-level state transition dynamics and reward functions to shift over time. This causes the higher-level policy to continually adjust to a moving target, thus destabilizing learning and slowing convergence. Additionally, the higher-level policy may generate subgoals that are currently unachievable by the lower primitive, further impeding effective learning. Therefore, the higher-level policy should consistently predict subgoals that are achievable given the current lower-level primitive.

To address these challenges, we introduce a novel framework that seamlessly integrates reinforcement learning (RL) and imitation learning (IL) to mitigate HRL's issues of non-stationarity and infeasible subgoal generation. At the core of our approach is *Primitive-Informed Parsing* (PIP), a mechanism that periodically segments expert demonstration trajectories to construct a subgoal transition dataset tailored for the higher-level policy. By continually updating this dataset, PIP adaptively identifies *achievable* subgoals that align with the evolving capabilities of the current lower-level primitive.

Leveraging this adaptive subgoal transition dataset, we employ an inverse reinforcement learning (IRL) objective to regularize the higher-level policy, ensuring it consistently generates subgoals that are achievable by the current lower-level primitive. By grounding subgoal generation in the evolving capabilities of the lower primitive, this approach naturally induces a subgoal curriculum that mitigates non-stationarity in hierarchical reinforcement learning, thereby stabilizing training and improving overall performance.

CRISP efficiently integrates RL and IL by jointly optimizing the RL objective (which promotes efficient autonomous exploration) with an IRL regularization that mitigates non-stationarity and stabilizes hierarchical training. This unified framework enables the agent to explore effectively while leveraging a curriculum of achievable subgoals, leading to more stable learning and enhanced sample efficiency.

We evaluate CRISP on six challenging simulated domains: maze navigation, pick-and-place, bin packing, hollow, rope manipulation, and the franka-kitchen suite, showing that our approach achieves over 40% higher success rates than strong hierarchical and flat baselines, while consistently delivering superior sample efficiency and stable hierarchical learning. Further, we perform experiments on real-world pick-and-place, bin, and rope-manipulation tasks, where the same advantages persist. Figure 1 illustrates the PIP procedure.
Our main contributions are as follows:

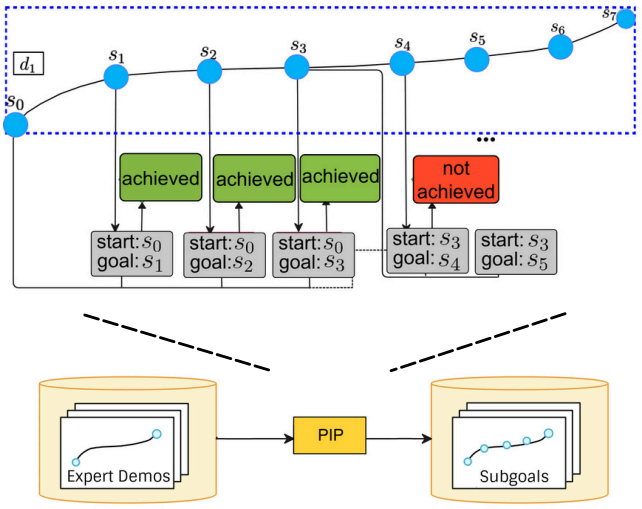


Figure 1. PIP Overview. PIP segments expert demonstrations by consecutively passing demonstration states as subgoals (s_i: for $i = 1$ to 6). If the lower primitive is unable to achieve state i in c timesteps (here $i =$ 4th timestep), s_{i-1} (here s_3) is selected as subgoal for initial state s_0, since s_{i-1} (here s_3) was the last reachable subgoal. The process is repeated with s_{i-1} (here s_3) as next initial state.

- **Adaptive subgoal dataset generation via PIP.** CRISP employs PIP to periodically generate a dataset of achievable subgoals for the lower primitive.
- **Curriculum via IRL regularization.** CRISP employs an IRL based objective to guide the high-level policy to predict a curriculum of achievable subgoals, thereby mitigating non-stationarity in HRL.
- **Extensive simulated benchmarks.** Across six sparse-reward navigation and manipulation tasks, CRISP achieves more than 40% higher success rates and faster convergence than prior methods.
- **Real-robot validation.** The policies trained with CRISP transfer directly to physical pick-and-place, bin, and rope-manipulation scenarios without additional fine-tuning, outperforming competing approaches.

2 Related Work

Hierarchical Reinforcement Learning. Learning effective policy hierarchies is a long-standing goal in RL (Barto and Mahadevan 2003; Sutton, Precup, and Singh 1999; Parr and Russell 1998; Dietterich 1999). Goal-conditioned control constrains exploration to specific targets (Kaelbling 1993; Foster and Dayan 2002) and has been lifted to hierarchical settings through two-level architectures (Wulfmeier et al. 2019, 2020; Ding et al. 2019). To curb non-stationarity arising from an evolving low-level policy, HIRO (Nachum et al. 2018) and HAC (Levy, Jr., and Saenko 2017) relabel past subgoals in the replay buffer. CRISP instead imposes an IRL regularizer that steers the high-level policy toward *reachable* subgoals predicted by primitive-informed parsing (PIP), yielding greater stability. Reset-controller work focuses on learning policies that return the agent to safe states inside simulation (Salimans and Chen 2018; Florensa et al. 2017; Peng et al. 2018); CRISP uses environment resets only to *parse* expert demonstrations and is agnostic to resets during training. Skill-prior methods pre-train on related tasks before fine-tuning (Pertsch, Lee, and Lim 2020; Singh et al. 2020), but often struggle under distribution shift or sub-optimal demonstrations. Handcrafted primitive libraries (Dalal, Pathak, and Salakhutdinov 2021; Nasiriany, Liu, and Zhu 2021) avoid learning low-level skills at the cost of extensive domain engineering; CRISP learns both hierarchy levels jointly, eliminating this manual burden.

Learning from Demonstrations. Prior works (Nair et al. 2017; Rajeswaran et al. 2017; Hester et al. 2017) that leverage expert demonstrations to solve complex tasks have demonstrated impressive results. Expert demonstrations have been used to bootstrap option learning (Krishnan et al. 2017a; Fox et al. 2017; Shankar and Gupta 2020; Kipf et al. 2019). Other approaches use imitation learning to bootstrap hierarchical approaches in complex task domains (Shiarlis et al. 2018; Krishnan et al. 2017b, 2019; Kipf et al. 2019). Prior work (Gupta et al. 2019) uses fixed window based approach for parsing expert demonstrations to generate subgoal transition dataset for training higher level policy using imitation learning. However, such methods may produce sub-optimal subgoals for the lower primitive. In contrast, our relabeling approach (PIP) segments expert demonstration trajectories into *meaningful* subtasks by predicting reachable subgoals and balancing the task-split between hierarchical levels.

Curriculum Learning. Our approach is inspired from curriculum learning (Bengio et al. 2009), where task difficulty gradually increases in complexity, propelling the policy to achieve incrementally harder subgoals. A genetic curriculum based approach (Song and Schneider 2022) identifies unsolved scenarios to automatically generate an associated curriculum via adversarial training. ACCEL (Parker-Holder et al. 2022) proposes a regret based curriculum approach that keeps a record of previous scenarios, and selects the ones with highest regret. Prior work generates subgoals while considering the lower primitive performance, by factoring in the task success rate (Fournier et al. 2018; Florensa et al. 2018; Racaniere et al. 2019), value function (Ren et al. 2019; Sharma et al. 2021), achieved state density (Pitis et al. 2020), and value uncertainty (Kim, Lee, and Choi 2023). CRISP differs by deriving curricula directly from expert trajectories via IRL-regularized subgoal prediction, providing an implicit difficulty schedule tied to the ability of lower primitive.

3 Background

We consider *Universal Markov Decision Process* (UMDP) (Schaul et al. 2015) setting, where Markov Decision processes (MDP) are augmented with the goal space G. UMDPs are represented as a 6-tuple (S, A, P, R, γ, G), where S is the state space, A is the action space, $P(s^{'}|s,a) = \mathbb{P}(s_{t+1} = s^{'}|s_t = s, a_t = a)$ is the transition function that describes the probability of reaching state $s^{'}$, when the agent takes action a in the current state s. The reward function R gen-

Algorithm 1: PIP: Primitive Informed Parsing

Initialize $D_g = \{\}$
(final goal) $s_f = g$
for each trajectory $e = (s_0^e, s_1^e, .., s_{T-1}^e)$ in D **do**
  (initial state) $s_{in} \leftarrow s_0^e$
  Initialize list of subgoals $D_g^e = \{\}$
  for i = 1 **to** $T-1$ **do**
    Reset to initial state s_{in}
    Pass s_i^e as the current goal to π_L
    if s_i^e is not achieved by π_L in c time-steps **then**
      Add (s_{in}, s_{i-1}^e, s_f) to D_g^e
      $s_{in} \leftarrow s_{i-1}^e$
  $D_g \leftarrow D_g \cup D_g^e$

Algorithm 2: CRISP

Require: D (expert demonstrations)
  p (population hyperparameter)
  Initialize higher level subgoal transition dataset $D_g = \{\}$
  for epoch $i = 1 \ldots N$ **do**
    if $i\%p == 0$ **then**
      Clear D_g
      Populate D_g by relabeling D using PIP
    for j = 1 **to** $T-1$ **do**
      Collect experience using π_H and π_L
    Update lower policy via RL and IRL (Eq 4)
    Sample transitions from D_g
    Update higher policy via RL and IRL (Eq 3)

erates rewards r at every timestep. γ is the discount factor, and G is the goal space. In the UMDP setting, a fixed goal g is selected for an episode, and $\pi(a|s,g)$ denotes the goal-conditioned policy. The discounted future state distribution is represented as $d^\pi(s) = (1-\gamma)\sum_{t=0}^{T}\gamma^t P(s_t = s|\pi)$, and the c-step future state distribution for policy π is represented as $d_c^\pi(s) = (1-\gamma^c)\sum_{t=0}^{T}\gamma^{tc} P(s_{tc} = s|\pi)$. The overall objective is to learn policy $\pi(a|s,g)$ which maximizes the expected future discounted reward objective $J = (1-\gamma)^{-1}\mathbb{E}_{s\sim d^\pi, a\sim\pi(a|s,g), g\sim G}[r(s_t, a_t, g)]$

3.1 Problem Formulation

Let s be the current state and g be the final goal for the current episode. In our goal-conditioned hierarchical RL setup, the overall policy π is divided into multi-level policies. The higher level policy $\pi^H(s_g|s,g)$ predicts subgoals (Dayan and Hinton 1993) s_g for the lower level primitive $\pi^L(a|s,s_g)$, which in turn executes primitive actions a directly on the environment. The lower primitive π^L tries to achieve subgoal s_g within c timesteps, by maximizing intrinsic rewards r_{in} provided by the higher level policy. The higher level policy π^H gets extrinsic reward r_{ex} from the environment, and predicts the next subgoal s_g for the lower primitive. The process is continued until either the final goal g is achieved, or the episode terminates. We consider sparse reward setting where the lower primitive is sparsely rewarded intrinsic reward 0 if the agent reaches within δ^L distance of the predicted subgoal s_g and -1 otherwise: $r_{in} = -1(\|s_t - s_g\|_2 > \delta^L)$, and the higher level policy is sparsely rewarded extrinsic reward 0 if the achieved goal is within δ^H distance of the final goal g, and -1 otherwise: $r_{ex} = -1(\|s_t - g\|_2 > \delta^H)$. The expert demonstrations are represented as $D = \{e^i\}_{i=1}^N$, where $e^i = (s_0^e, s_1^e, \ldots, s_{T-1}^e)$.

3.2 Limitations of Existing HRL Approaches

HRL promises the advantages of temporal abstraction and improved exploration (Nachum et al. 2019). However it suffers from non-stationarity due to unstable lower primitive behavior. This hinders applying HRL advances to complex tasks, especially in sparse reward scenarios. The primary motivation of this work is to devise a hierarchical curriculum learning based approach to mitigate non-stationarity in HRL.

4 Methodology

This section explains (i) primitive-informed parsing (PIP) that adaptively builds the subgoal dataset according to the current lower primitive, (ii) IRL regularization that conditions the higher level policy to predict achievable sugoals, and (iii) joint SAC optimisation that unifies RL and IRL updates.

4.1 Primitive Informed Parsing (PIP)

PIP leverages the *current* low-level policy π_L to re-segment the expert, state-only dataset D into a buffer of *reachable* sub-goal transitions D_g (overview in Fig. 1).

We explain how PIP adaptively parses expert demonstration trajectories from D to generate subgoal transition dataset D_g. We start with current lower primitive π_L and an expert state demonstration trajectory $e = (s_0^e, s_1^e, \ldots, s_{T-1}^e)$. The environment is reset to initial state s_0^e. Starting at $i = 1$ to $T-1$, we incrementally provide states s_i^e as subgoals to lower primitive π_L. π_L tries to achieve s_i^e within c timesteps from the initial state. If π_L achieves the subgoal s_i^e within c timesteps, we provide s_{i+1}^e as the next subgoal. Conversely, if π_L fails to achieve the subgoal s_i^e from the initial state, we add s_{i-1}^e to the list of subgoals. Intuitively, since s_{i-1}^e was the last subgoal achieved by lower-level primitive, it is a good candidate for the next subgoal from initial state. Once we add s_{i-1}^e to the list of subgoals, we continue the process after setting s_{i-1}^e as the new initial state, until we reach the end of e. This subgoal transition sequence D_g^e is subsequently added to D_g. The PIP pseudocode is given in Algorithm 1.

Notably, PIP assumes ability to reset the environment to any state in D. Although this seems impracticable in real world robotic scenarios, this becomes feasible in our setup since we first learn good policies in simulation, and then deploy them in real world robotic scenarios. This follows the underlying assumption that with enough training in simulation, the policy becomes general enough to perform well in real world tasks. We perform extensive experiments to support this claim in Experiments section and discuss various ways to relax this assumption in Discussion section.

4.2 Inverse-RL Regularisation

The sub-goal buffer D_g produced by PIP serves as expert data for an IRL regulariser that nudges the *high-level* policy to pro-

pose *reachable* goals for the lower primitive. We follow the GAIL framework (Ho and Ermon 2016) with the stabilising least-squares GAN loss (LSGAN) (Mao et al. 2016).

Let $(s^e, s^e_g, s^e_{\text{next}}) \sim D_g$ denote an subgoal transition where s^e is the current state, s^e_g the (supervised) sub-goal, s^e_{next} the next state, and g^e the final task goal. The high-level policy $\pi^H_{\theta_H}(\cdot|s^e, g^e)$ predicts a sub-goal s_g, while a discriminator $D^H_{\varepsilon_H}$ with parameters ε_H tries to distinguish expert sub-goals from policy-generated ones. J^H_D represents upper level IRL objective, which depends on parameters $(\theta_H, \varepsilon_H)$. We bootstrap the learning of higher level policy by optimizing:

$$\max_{\theta_H}\min_{\varepsilon_H} J^H_D(\theta_H,\varepsilon_H) = \max_{\theta_H}\min_{\varepsilon_H}\frac{1}{2}\mathbb{E}_{(s^e,s^e_g,\cdot)\sim D_g}[\mathbb{D}^H_{\varepsilon_H}(s^e_g)-1]^2 + \frac{1}{2}\mathbb{E}_{(s^e,\cdot,\cdot)\sim D_g, s_g\sim\pi^H_{\theta_H}(\cdot|s^e,g^e)}[\mathbb{D}^H_{\varepsilon_H}(\pi^H_{\theta_H}(\cdot|s^e,g^e))-0]^2. \tag{1}$$

Minimising (1) with respect to ε_H and maximising it with respect to θ_H pushes $\pi^H_{\theta_H}$ to generate sub-goals that are indistinguishable from those in D_g, thus inducing an automatic curriculum whose difficulty evolves according to the low-level policy.

Similarly for lower level primitive, let $(s^f, a^f, s^f_{next}) \sim D^L_g$ be lower level expert transition where s^f is current state, s^f_{next} is next state, g^f is final goal, a is the primitive action predicted by lower policy $\pi^L_{\theta_L}(\cdot|s^f, s^e_g)$ with parameters θ_L, and $\mathbb{D}^L_{\varepsilon_L}$ be the lower level discriminator with parameters ε_L. Let J^L_D represent lower level IRL objective, which depends on parameters $(\theta_L, \varepsilon_L)$. The lower level IRL objective is thus:

$$\max_{\theta_L}\min_{\varepsilon_L} J^L_D(\theta_L,\varepsilon_L) = \max_{\theta_L}\min_{\varepsilon_L}\frac{1}{2}\mathbb{E}_{(s^f,a^f,\cdot)\sim D^L_g}[\mathbb{D}^L_{\varepsilon_L}(a^f)-1]^2 + \frac{1}{2}\mathbb{E}_{(s^f,\cdot,\cdot)\sim D^L_g, a\sim\pi^L_{\theta_L}(\cdot|s^f,s^e_g)}[\mathbb{D}^L_{\varepsilon_L}(\pi^L_{\theta_L}(\cdot|s^f,s^e_g))-0]^2. \tag{2}$$

4.3 Joint Optimization

To train both hierarchical levels stably, we combine off-policy RL with IRL regularization. While the off-policy RL objective enables agents to autonomously explore and learn from their interactions with the environment, the IRL objective allows them to leverage expert demonstrations for more sample-efficient and guided skill acquisition. The high-level policy is optimized to generate subgoals which, when provided to the lower-level primitive, maximize the expected sum of discounted rewards for each task.

Let T denote the episode horizon and g the sampled episodic goal. We denote the standard RL objectives for high and low levels as $J^H_{\theta_H}$ and $J^L_{\theta_L}$, respectively. Both objectives are augmented with their respective IRL losses, weighted by a regularization factor ψ. The joint optimization objectives are:

$$\max_{\theta_H}\left[J^H_{\theta_H} + \psi\min_{\varepsilon_H} J^H_D(\theta_H,\varepsilon_H)\right], \tag{3}$$

$$\max_{\theta_L}\left[J^L_{\theta_L} + \psi\min_{\varepsilon_L} J^L_D(\theta_L,\varepsilon_L)\right]. \tag{4}$$

The lower-level policy is regularized using expert demonstration data, while the upper-level policy leverages the subgoal transition dataset generated by PIP. ψ trades off between pure task reward and IRL-guided imitation: $\psi = 0$ yields a vanilla HRL agent with no regularization; high values emphasize imitation and may risk overfitting to demonstrations.

See Algorithm 2 for the full training pseudocode. This joint framework enables continual adaptation at both levels: the lower-level primitive explores via RL to reach higher-level subgoals, while its IRL regularizer keeps its behavior close to expert demonstrations, ensuring reliable skill learning. Simultaneously, the higher-level policy uses IRL guidance to produce subgoals attainable by the evolving primitive, promoting task progress without inducing non-stationarity and thereby stabilizing hierarchical training.

5 Experiments

We perform experiments to answer the following questions:

- **Q1.** Does CRISP's primitive-informed parsing outperform fixed-window parsing approaches?
- **Q2.** Does CRISP improve sample efficiency and stability over state-of-the-art HRL baselines?
- **Q3.** Does CRISP mitigate HRL non-stationarity?
- **Q4.** Does IRL regularization generate a curriculum of achievable subgoals?
- **Q5.** Can CRISP policies transfer to real-world robots?
- **Q6.** What is the impact of each design choice?

Environment and Implementation details. We perform experiments on six complex robotic environments with continuous state and action spaces that require long term planning: (i) maze navigation, (ii) pick and place, (iii) bin, (iv) hollow, (v) rope manipulation, and (vi) franka kitchen. We use the off-policy Soft Actor Critic (Haarnoja et al. 2018) as RL algorithm with Adam (Kingma and Ba 2014) optimizer. Extensive implementation and environment details are provided in Supplementary Sections 5 and 6. We collect 28 expert demonstrations in kitchen and 100 demonstrations in all other tasks. The datasets are collected such that they cover a large enough distribution of initial state conditions. For each baseline, we conducted a comprehensive hyper-parameter grid search within the ranges recommended by the original publications to ensure optimal performance. We provide the hyper-parameter list in Supplementary Section 4, and demonstrations generation approach in Supplementary Section 7. The implementation code is also provided in the supplementary.

Evaluating two CRISP regularizers. Along with our RL objective, We evaluate CRISP using two imitation learning regularizers: IRL regularization (CRISP-IRL) and BC regularization (CRISP-BC). Evaluating both BC (behavior cloning) and IRL (inverse-RL) variants offers complementary insights due to their fundamentally different approaches to imitation learning (IL). BC directly learns a mapping from states to expert actions, making it sample-efficient and effective when demonstrations are abundant, high-quality, and cover the relevant state distribution. However, BC is prone to compounding

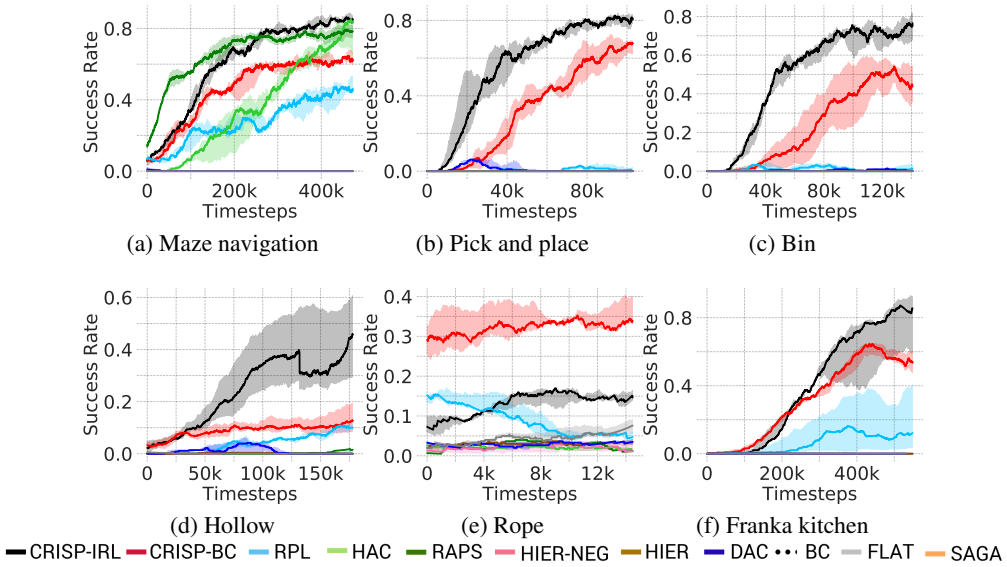


Figure 2. Success rate comparison. This figure compares the methods on six robotic tasks. The solid line represents the mean and shaded region represents the standard deviation across 5 seeds. We compare CRISP-IRL and CRISP-BC against multiple baselines: RPL (Relay Policy learning (Gupta et al. 2019)), HAC (Hierarchical Actor Critic (Levy, Jr., and Saenko 2017)), SAGA (Wang et al. 2023), RAPS (Dalal, Pathak, and Salakhutdinov 2021), HIER-NEG (HRL implementation using SAC (Haarnoja et al. 2018) where higher policy is negatively rewarded when lower primitive fails to achieve the subgoal), HIER (HRL implementation using SAC), DAC (Kostrikov et al. 2018), and FLAT (Single-level RL policy implemented using SAC). CRISP achieves more than 40% higher success rates, shows better convergence speed and training stability over the baselines.

errors and struggles with distributional shift, which can degrade performance in complex or noisy environments. IRL, in contrast, infers the underlying reward function guiding the expert's behavior, enabling the agent to generalize beyond demonstrations by optimizing policies that reflect the expert's intent, even in unseen states. This often leads to better robustness and the ability to surpass imperfect experts but can require more complex training. By separately implementing both IRL and BC variants, we gain a clearer understanding of their relative strengths and weaknesses across different tasks, and also renders insights into which IL approach is better suited for specific robotic control tasks.

In Figure 2 we report the success rates of CRISP alongside all baselines, averaged over five independent random seeds. Training conditions (network architecture, optimizer, batch size, replay buffer size, etc.) were kept identical across methods unless stated otherwise; each baseline was re-implemented from scratch and its hyper-parameters grid-searched within the ranges recommended by the original papers to guarantee a fair comparison.

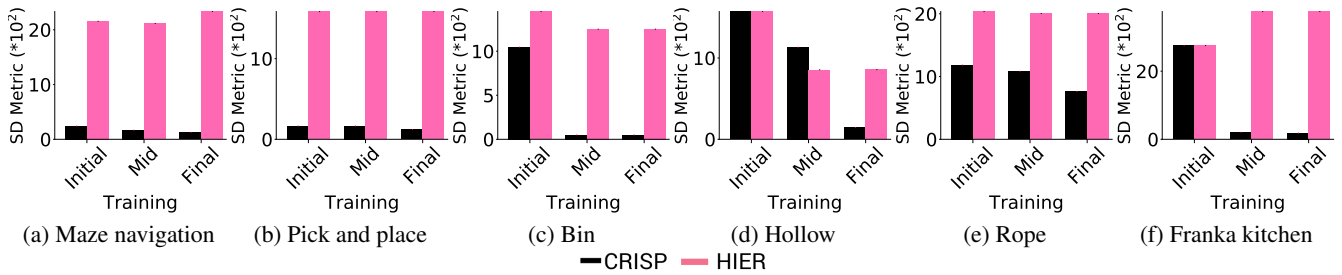
The subsections that follow present empirical results that systematically address each of the questions listed above.

Q1. Does CRISP's adaptive primitive-informed parsing outperform fixed-window parsing approaches? In Figure 2, we compare CRISP with *RPL* (Relay Policy Learning) (Gupta et al. 2019), to demonstrate the efficacy of primitive-informed parsing compared to fixed window based parsing. RPL first uses supervised pre-training from undirected demonstrations, and then fine-tunes the policy using RL. To ensure fair comparisons, we use a variant of *RPL* without pre-training. Thus, the only difference between CRISP and RPL is that CRISP uses primitive-informed parsing to select subgoals, and RPL uses fixed window based parsing.

As illustrated in Figure 2, CRISP consistently outperforms *RPL* across all tasks, highlighting the effectiveness of primitive-informed parsing over fixed window parsing. Although both approaches integrate RL and IL, they primarily differ in subgoal assignment. CRISP employs primitive-informed parsing to dynamically identify subgoals feasible for the current lower-level primitive, enabling automatic adaptation of subgoal difficulty without manual tuning.

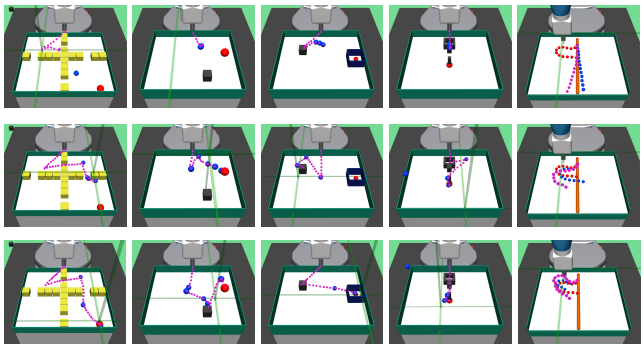
This adaptability proves especially valuable when perturbations or random exploration cause the agent to deviate from the demonstration trajectory. In such cases, CRISP's high-level policy responds by generating future subgoals that steer the primitive back toward success, enabling recovery from off-manifold states where RPL typically fails. For example, in rope manipulation experiments, agents using fixed window parsing often become stuck after a poor poke, unable to recover. In contrast, CRISP promptly proposes a reachable intermediate rope configuration, allowing the agent to proceed and ultimately succeed.

We also elucidate the importance of primitive-informed parsing by considering with a variant of CRISP that uses the subgoal dataset collected using fixed window based parsing. We compare this variant (CRISP-RPL) with our approach



(a) Maze navigation (b) Pick and place (c) Bin (d) Hollow (e) Rope (f) Franka kitchen

Figure 3. Non-stationarity metric comparison. This figure compares the subgoal distance (SD) metric between the subgoals predicted by the higher level policy and the states achieved by the lower level policy during training. (The columns represent Initial: when training begins, Mid: half-way during training, Final: when training ends, e.g. since maze navigation is trained for 4.7E6 timesteps, the values are Initial: iteration 1, Mid: iteration 2.35E6, and Final: iteration 4.7E6). As seen in figure, CRISP consistently produces efficient subgoals leading to low distances between the predicted and achieved subgoals throughout the training process. This mitigates non-stationarity in HRL.



(a) Maze (b) Pick (c) Bin (d) Hollow (e) Rope

Figure 4. Subgoal curriculum. This figure depicts the progression of CRISP subgoal generation with training phases (Row 1: initial training, Row 2: mid training, Row 3: end training) in maze navigation, pick and place, bin, hollow, rope and kitchen environment, (walls:yellow, final goal:red, subgoals:blue, and the trajectories are shown in pink). As seen from figure, the subgoals get progressively harder, generating a curriculum for training higher level policy.

(CRISP-IRL) in Supplementary Figure 1. (CRISP-IRL) consistently outperforms this baseline, showing that primitive-informed parsing is crucial for improved performance.

Q2. Does CRISP improve sample efficiency and stability over state-of-the-art HRL baselines? We benchmark CRISP against diverse hierarchical and flat baselines to gauge sample efficiency and training stability. HAC (Levy, Jr., and Saenko 2017) assumes an optimal low-level primitive, which is often untrue in practice. Thus, HAC may produce sub-optimal subgoals. As Figure 2 shows, HAC fares well on the simple maze task but fails on harder tasks. In contrast, CRISP's primitive-informed parsing dynamically aligns the subgoal difficulty with the evolving capabilities of the lower-level primitive, yielding stable HRL learning and higher success rates performance. SAGA (Wang et al. 2023) uses a discriminator generate subgoals but still suffers when those goals remain unreachable. CRISP consistently surpasses SAGA on harder tasks due to primitive-informed parsing and IRL regularization, which ensure subgoals remain achievable and mitigate non-stationarity.

In *RAPS* (Dalal, Pathak, and Salakhutdinov 2021), handcrafted action primitives drive the low-level policy, leaving the high level to choose their sequence. Designing these primitives is labour-intensive, and except for the maze navigation task, RAPS performs poorly. We attribute this to navigation reducing the high-level choice to a simple direction selection, whereas other domains demand more complex primitives. Further, our sparse-reward re-implementation of tasks accounts for RAPS's poor performance. In contrast, CRISP outperforms RAPS on all tasks without requiring any domain-specific primitives. We do not evaluate RAPS on the rope task because creating suitable primitives there proved impractical.

We also compare our approach with two other hierarchical baselines: *HIER* and *HIER-NEG*, which are hierarchical off-policy SAC (Haarnoja et al. 2018) based baselines that do not leverage expert demonstrations. In *HIER-NEG*, the higher level policy is negatively rewarded if the lower primitive is unable to reach the predicted subgoal. CRISP outperforms these hierarchical baselines, indicating that improvements stem not only from hierarchical abstraction but from our primitive-informed parsing and primitive regularization, that directly mitigates non-stationarity and enforces subgoal reachability.

Finally, we compare CRISP against flat baselines, including (DAC) Discriminator Actor Critic (which leverages expert demonstrations), single-level RL (FLAT), and behavior cloning (BC). These baselines fail to reliably solve the tasks tested, reinforcing the critical importance of both hierarchical abstraction and curriculum-based subgoal adaptation. Thus, our comprehensive evaluation demonstrates that CRISP significantly improves sample efficiency and stabilizes learning compared to several existing hierarchical and flat baselines.

Q3. How well does CRISP mitigate non-stationarity? We illustrate CRISP's ability to mitigate non-stationarity in HRL in Figure 3. To quantify this, we compare CRISP to the *HIER* baseline by measuring the average distance during several stages of training (Initial: when training begins, Mid: half-way during training, Final: when training ends, e.g. since maze navigation is trained for 4.7E6 timesteps, the values are Initial: iteration 1, Mid: iteration 2.35E6, and Final: iteration 4.7E6) between the subgoals issued by the higher-level policy

and the actual states reached by the lower-level primitive. Lower distance values indicate that the higher-level policy is generating subgoals that are well-aligned with the capabilities of the lower-level primitive, which leads to more consistent goal achievement and reduces non-stationarity within the hierarchical learning process. As shown, CRISP consistently produces achievable subgoals throughout training, resulting in superior stability and effective non-stationarity mitigation.

Q4. Does PIP and IRL regularization generate a curriculum of achievable subgoals? Primitive-Informed Parsing (PIP) and IRL regularization plays a crucial role in generating a curriculum of subgoals that dynamically adapt to the lower-level primitive's evolving capabilities. PIP periodically segments expert demonstrations by identifying subgoals that the current low-level controller can realistically achieve within a given horizon, effectively ensuring that each subgoal is feasible. Subsequently, IRL regularization regularizes the higher level policy to predict such achievable subgoals according to the current lower level primitive. This adaptive parsing leads to an implicit curriculum: early in training, subgoals are simpler and localized, while later stages naturally involve more complex and distant targets as the primitive improves. This is evident in Figure 4, where early on, subgoals cluster close to the agent (Row 1); mid-training (Row 2) they move outward and diversify; by the final phase (Row 3) they span the full task horizon and often coincide with, or lie just before, the final goal. The increasing subgoal difficulty across rows demonstrates CRISP's ability to generate a curriculum of achievable subgoals for the lower primitive.

Q5. Can CRISP transfer to real-world robots? To evaluate real-world deployability, we conducted experiments on pick-and-place, bin packing, and rope manipulation tasks using a four-axis Dobot Magician desktop arm (Supplementary Figure 6). The robot was controlled via the manufacturer's Python SDK at a fixed 20 Hz command rate ($\delta t = 50$ ms). Each episode lasted approximately 40 seconds for pick-and-place and bin tasks, and 60 seconds for rope manipulation—allowing sufficient time for joint-space commands to achieve the goal or time out. After each episode, a "Home" macro returned the arm to its calibrated start pose, followed by a brief scripted motion to position the end-effector above the workspace. This automatic reset took roughly 10 seconds on average, minimizing downtime and ensuring consistent initial conditions. We used a Realsense D435 depth camera to track the robot, block, bin, and rope cylinder positions. Due to the difficulty of precisely estimating linear and angular velocities in real settings, we assigned them small fixed values, which proved effective in practice. We performed five sets of ten trials per task and report the average success rates.

In real-robot experiments, the ability to structure the task into feasible subgoals is essential: both CRISP-IRL and CRISP-BC successfully decompose complex tasks into reachable stages, leading to notably higher real-world success rates. Further, by ensuring that subgoals correspond to attainable states, these methods inherently promote safer operation, as the robot avoids attempting unsafe or unreachable maneuvers. CRISP-IRL achieves an accuracy of 0.6, 0.6 and 0.5, whereas CRISP-BC achieves accuracy of 0.8, 0.3, 0.3 on pick and place, bin and rope tasks respectively. We also deployed the next best performing baseline *RPL* on the tasks, however it is unable to generate good subgoals and the agent gets stuck, thus failing to show any progress in the tasks.

Q6. What is the impact of each design choice? We also perform ablations to select the population hyperparameter p (Supplementary Figure 3), learning rate ψ (Supplementary Figure 2), RPL window size ablation (Supplementary Figure 5), and the optimal number of expert demos (Supplementary Figure 4) in Supplementary Section 1. Extensive ablation studies show that CRISP maintains strong performance across a wide range of hyper-parameter settings, indicating that the method is not unduly sensitive to precise hyper-parameter tuning. We empirically found that very large values of p are unable to generate good curriculum of subgoals. Further, when ψ is too high, the method might overfit to expert data, whereas if ψ is too small, CRISP is unable to utilize IL regularization. We perform ablations to deduce the optimal number of expert demos required for each task. If the number of expert demos is too small, the policy may overfit. Although the number of expert demos are subject to availability, we increase the number until there is no significant improvement. We analysed the effect of varying the quality of expert data, and found direct co-relation between quality of expert data and performance. Finally, we provide qualitative visualizations for all tasks in Supplementary Section 8.

6 Discussion

Limitations. Although primitive-informed parsing introduces some additional overhead, especially in long-horizon tasks, our empirical results indicate that this cost remains marginal. In future, we plan to also analyze using undirected demonstrations. Further, we recognize that CRISP uses environment resets to parse expert demonstrations, which is often impractical or costly in real-world environments without simulations. One promising solution is to integrate a learned reset controller, such as backward policies that return the system to previous states, allowing CRISP to function in the absence of manual resets (Eysenbach et al. 2017). Another alternative is to generate curricula using only states naturally visited during training or by dynamically identifying subgoals reachable from the current policy's exploration.

Conclusion. We present CRISP, a hierarchical curriculum based framework to address non-stationarity in HRL. We leverage PIP to parse expert demonstrations and construct a curriculum of achievable subgoals according to the low-level policy. We evaluate CRISP on complex, sparse robotic navigation and manipulation tasks in simulation, demonstrating substantial performance gains over existing baselines. In addition, we show that CRISP shows strong performance on real-world robotic tasks. These findings underscore the promise of hierarchical curriculum learning as an efficient alternative for robust real-world robotic systems.

Acknowledgments This work was partially supported by Research-I Foundation of the CSE Deptt. at IIT Kanpur.

References

Andrychowicz, M.; Wolski, F.; Ray, A.; Schneider, J.; Fong, R. H.; Welinder, P.; McGrew, B.; Tobin, J.; Abbeel, P.; and

Zaremba, W. 2017. Hindsight Experience Replay. In *NIPS*.

Barto, A. G.; and Mahadevan, S. 2003. Recent Advances in Hierarchical Reinforcement Learning. *Discrete Event Dynamic Systems*, 13: 341–379.

Bengio, Y.; Louradour, J.; Collobert, R.; and Weston, J. 2009. Curriculum Learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, 41–48. New York, NY, USA: Association for Computing Machinery. ISBN 9781605585161.

Dalal, M.; Pathak, D.; and Salakhutdinov, R. 2021. Accelerating Robotic Reinforcement Learning via Parameterized Action Primitives. *CoRR*, abs/2110.15360.

Dayan, P.; and Hinton, G. E. 1993. Feudal Reinforcement Learning. In *Advances in Neural Information Processing Systems 5, [NIPS Conference]*, 271–278. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. ISBN 1-55860-274-7.

Dietterich, T. G. 1999. Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. *CoRR*, cs.LG/9905014.

Ding, Y.; Florensa, C.; Abbeel, P.; and Phielipp, M. 2019. Goal-conditioned Imitation Learning. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Eysenbach, B.; Gu, S.; Ibarz, J.; and Levine, S. 2017. Leave no Trace: Learning to Reset for Safe and Autonomous Reinforcement Learning. arXiv:1711.06782.

Florensa, C.; Held, D.; Geng, X.; and Abbeel, P. 2018. Automatic goal generation for reinforcement learning agents. In *International conference on machine learning*, 1515–1528. PMLR.

Florensa, C.; Held, D.; Wulfmeier, M.; Zhang, M.; and Abbeel, P. 2017. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*, 482–495. PMLR.

Foster, D.; and Dayan, P. 2002. Structure in the Space of Value Functions. *Machine Learning*, 49(2-3): 325–346.

Fournier, P.; Sigaud, O.; Chetouani, M.; and Oudeyer, P.-Y. 2018. Accuracy-based curriculum learning in deep reinforcement learning. *arXiv preprint arXiv:1806.09614*.

Fox, R.; Krishnan, S.; Stoica, I.; and Goldberg, K. 2017. Multi-Level Discovery of Deep Options. arXiv:1703.08294.

Gupta, A.; Kumar, V.; Lynch, C.; Levine, S.; and Hausman, K. 2019. Relay Policy Learning: Solving Long-Horizon Tasks via Imitation and Reinforcement Learning. *CoRR*, abs/1910.11956.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *CoRR*, abs/1801.01290.

Hester, T.; Vecerík, M.; Pietquin, O.; Lanctot, M.; Schaul, T.; Piot, B.; Sendonaris, A.; Dulac-Arnold, G.; Osband, I.; Agapiou, J. P.; Leibo, J. Z.; and Gruslys, A. 2017. Learning from Demonstrations for Real World Reinforcement Learning. *CoRR*, abs/1704.03732.

Ho, J.; and Ermon, S. 2016. Generative Adversarial Imitation Learning. *CoRR*, abs/1606.03476.

Kaelbling, L. P. 1993. Learning to Achieve Goals. In *IN PROC. OF IJCAI-93*, 1094–1098. Morgan Kaufmann.

Kim, S.; Lee, K.; and Choi, J. 2023. Variational curriculum reinforcement learning for unsupervised discovery of skills. *arXiv preprint arXiv:2310.19424*.

Kingma, D. P.; and Ba, J. 2014. Adam: A Method for Stochastic Optimization. Cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Kipf, T.; Li, Y.; Dai, H.; Zambaldi, V.; Sanchez-Gonzalez, A.; Grefenstette, E.; Kohli, P.; and Battaglia, P. 2019. Compile: Compositional imitation learning and execution. In *International Conference on Machine Learning*, 3418–3428. PMLR.

Kostrikov, I.; Agrawal, K. K.; Dwibedi, D.; Levine, S.; and Tompson, J. 2018. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. *arXiv preprint arXiv:1809.02925*.

Krishnan, S.; Fox, R.; Stoica, I.; and Goldberg, K. 2017a. DDCO: Discovery of Deep Continuous Options for Robot Learning from Demonstrations. arXiv:1710.05421.

Krishnan, S.; Fox, R.; Stoica, I.; and Goldberg, K. 2017b. DDCO: Discovery of Deep Continuous Options forRobot Learning from Demonstrations. *CoRR*, abs/1710.05421.

Krishnan, S.; Garg, A.; Liaw, R.; Thananjeyan, B.; Miller, L.; Pokorny, F. T.; and Goldberg, K. 2019. SWIRL: A sequential windowed inverse reinforcement learning algorithm for robot tasks with delayed rewards. *The International Journal of Robotics Research*, 38(2-3): 126–145.

Kulkarni, T. D.; Narasimhan, K.; Saeedi, A.; and Tenenbaum, J. B. 2016. Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation. *CoRR*, abs/1604.06057.

Levine, S.; Finn, C.; Darrell, T.; and Abbeel, P. 2015. End-to-End Training of Deep Visuomotor Policies. *CoRR*, abs/1504.00702.

Levy, A.; Jr., R. P.; and Saenko, K. 2017. Hierarchical Actor-Critic. *CoRR*, abs/1712.00948.

Mao, X.; Li, Q.; Xie, H.; Lau, R. Y. K.; and Wang, Z. 2016. Multi-class Generative Adversarial Networks with the L2 Loss Function. *CoRR*, abs/1611.04076.

Nachum, O.; Gu, S.; Lee, H.; and Levine, S. 2018. Data-Efficient Hierarchical Reinforcement Learning. *CoRR*, abs/1805.08296.

Nachum, O.; Tang, H.; Lu, X.; Gu, S.; Lee, H.; and Levine, S. 2019. Why does hierarchy (sometimes) work so well in reinforcement learning? *arXiv preprint arXiv:1909.10618*.

Nair, A.; McGrew, B.; Andrychowicz, M.; Zaremba, W.; and Abbeel, P. 2017. Overcoming Exploration in Reinforcement Learning with Demonstrations. *CoRR*, abs/1709.10089.

Nasiriany, S.; Liu, H.; and Zhu, Y. 2021. Augmenting Reinforcement Learning with Behavior Primitives for Diverse Manipulation Tasks. *CoRR*, abs/2110.03655.

Parker-Holder, J.; Jiang, M.; Dennis, M.; Samvelyan, M.; Foerster, J.; Grefenstette, E.; and Rocktäschel, T. 2022. Evolving curricula with regret-based environment design. In *International Conference on Machine Learning*, 17473–17498. PMLR.

Parr, R.; and Russell, S. 1998. Reinforcement Learning with Hierarchies of Machines. In Jordan, M.; Kearns, M.; and Solla, S., eds., *Advances in Neural Information Processing Systems*, volume 10. MIT Press.

Peng, X. B.; Abbeel, P.; Levine, S.; and Van de Panne, M. 2018. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions On Graphics (TOG)*, 37(4): 1–14.

Pertsch, K.; Lee, Y.; and Lim, J. J. 2020. Accelerating Reinforcement Learning with Learned Skill Priors. *CoRR*, abs/2010.11944.

Pitis, S.; Chan, H.; Zhao, S.; Stadie, B.; and Ba, J. 2020. Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. In *International Conference on Machine Learning*, 7750–7761. PMLR.

Racaniere, S.; Lampinen, A. K.; Santoro, A.; Reichert, D. P.; Firoiu, V.; and Lillicrap, T. P. 2019. Automated curricula through setter-solver interactions. *arXiv preprint arXiv:1909.12892*.

Rajeswaran, A.; Kumar, V.; Gupta, A.; Schulman, J.; Todorov, E.; and Levine, S. 2017. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. *CoRR*, abs/1709.10087.

Ren, Z.; Dong, K.; Zhou, Y.; Liu, Q.; and Peng, J. 2019. Exploration via hindsight goal generation. *Advances in Neural Information Processing Systems*, 32.

Salimans, T.; and Chen, R. 2018. Learning montezuma's revenge from a single demonstration. *arXiv preprint arXiv:1812.03381*.

Schaul, T.; Horgan, D.; Gregor, K.; and Silver, D. 2015. Universal Value Function Approximators. In Bach, F.; and Blei, D., eds., *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, 1312–1320. Lille, France: PMLR.

Shankar, T.; and Gupta, A. 2020. Learning Robot Skills with Temporal Variational Inference. In *ICML*.

Sharma, A.; Gupta, A.; Levine, S.; Hausman, K.; and Finn, C. 2021. Autonomous reinforcement learning via subgoal curricula. *Advances in Neural Information Processing Systems*, 34: 18474–18486.

Shiarlis, K.; Wulfmeier, M.; Salter, S.; Whiteson, S.; and Posner, I. 2018. TACO: Learning Task Decomposition via Temporal Alignment for Control. In Dy, J.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 4654–4663. PMLR.

Singh, A.; Liu, H.; Zhou, G.; Yu, A.; Rhinehart, N.; and Levine, S. 2020. Parrot: Data-Driven Behavioral Priors for Reinforcement Learning. *CoRR*, abs/2011.10024.

Song, Y.; and Schneider, J. 2022. Robust Reinforcement Learning via Genetic Curriculum. In *2022 International Conference on Robotics and Automation (ICRA)*, 5560–5566. IEEE.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1): 181–211.

Vecerík, M.; Hester, T.; Scholz, J.; Wang, F.; Pietquin, O.; Piot, B.; Heess, N.; Rothörl, T.; Lampe, T.; and Riedmiller, M. A. 2017. Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards. *CoRR*, abs/1707.08817.

Vezhnevets, A. S.; Osindero, S.; Schaul, T.; Heess, N.; Jaderberg, M.; Silver, D.; and Kavukcuoglu, K. 2017. FeUdal Networks for Hierarchical Reinforcement Learning. *CoRR*, abs/1703.01161.

Wang, V. H.; Pajarinen, J.; Wang, T.; and Kämäräinen, J.-K. 2023. State-conditioned adversarial subgoal generation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, 10184–10191.

Wulfmeier, M.; Abdolmaleki, A.; Hafner, R.; Springenberg, J. T.; Neunert, M.; Hertweck, T.; Lampe, T.; Siegel, N. Y.; Heess, N.; and Riedmiller, M. A. 2019. Regularized Hierarchical Policies for Compositional Transfer in Robotics. *CoRR*, abs/1906.11228.

Wulfmeier, M.; Rao, D.; Hafner, R.; Lampe, T.; Abdolmaleki, A.; Hertweck, T.; Neunert, M.; Tirumala, D.; Siegel, N. Y.; Heess, N.; and Riedmiller, M. A. 2020. Data-efficient Hindsight Off-policy Option Learning. *CoRR*, abs/2007.15588.