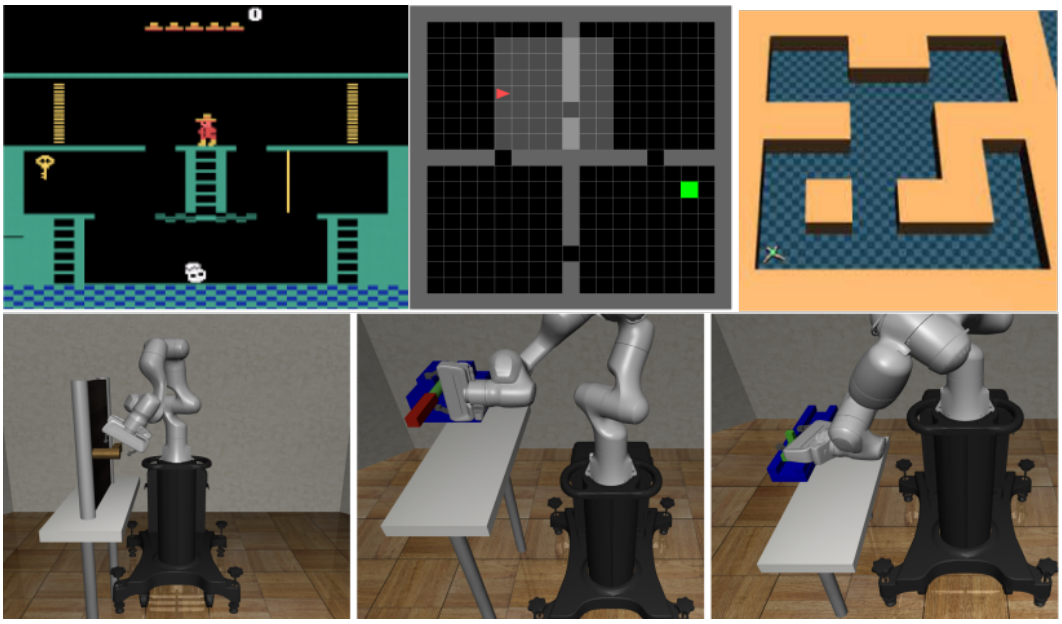


Figure 6: Domains used for our experiments in Section 4. Enumerating left to right from the top-left: first screen of Montezuma's Revenge, MiniGrid FourRooms, Ant Medium Maze, Robosuite Door, Robosuite Lever, and Robosuite Slide.

A Task Descriptions

Four Rooms. This task, which is part of the MINIGRID suite [Chevalier-Boisvert et al., 2018], is an adaptation of the four rooms problem presented in the paper introducing the Options framework [Sutton et al., 1999]. Observations are 84×84 images; the underlying state-space contains approximately 19×19 grid locations and 4 possible orientations of the player. We defined 4 options whose termination conditions (subgoals) were to navigate to the center of the 4 rooms. $\mathcal{S}_0$ was the set of all empty grid locations.

Montezuma's Revenge. As is standard in ALE [Bellemare et al., 2013], observations are 84×84 images, action space is a set of 18 discrete actions [Machado et al., 2018]. We defined start states scattered across the first room where the player was on the ground (not jumping or falling), was not too close to the skull and did not already have the key in its possession. We also defined 5 options whose termination conditions are reaching the left door, right door, bottom-right of the first screen, bottom-left of the first screen and getting the key.

Robosuite manipulation tasks. Three constrained manipulation tasks were used to study the task-oriented grasping performance of our initiation set learning algorithms: opening a door, flipping a lever, and sliding a knob. The door task was originally implemented in ROBOSUITE [Zhu et al., 2020]; the others were implemented and described in the work of Rosen et al. [2022]. All three are 1-DoF articulated objects which require making and sustaining contact to manipulate. The 52-dimensional observation space consists of the robot's proprioceptive state (joint position, joint velocity, end-effector position, gripper state, tactile data) as well as the object state (object pose, joint position, handle position). The action space employed is operational space control with variable impedance [Martín-Martín et al., 2019]: the agent controls the 6-DoF change in position and orientation of the end-effector, the 6-DoF change in stiffness, and 1-DoF gripper state. Episodes have a maximum length of 250 steps. In each task, $\mathcal{S}_0$ was a set of arm configurations establishing contact with the object; see Section B.2.

Ant Medium Maze. The goal location is small region around $(20, 20)$. A state is considered to satisfy a goal if the two have a euclidean distance of 0.5 units or less $R(s, g) = ||s - g||_2 < 0.5$. The agent is evaluated by rolling out the learned policy once every 10 episodes; during evaluation,

the agent starts from a small region around $(0, 0)$, during training it starts at a location randomly sampled from the open locations in the maze. The task reward function is -1 for non-goal transitions and a terminating reward of 0 for reaching the goal. Episodes last a maximum of 1000 steps. The training and evaluation protocol is identical to Bagaria et al. [2021a], except for the fact that we learn initiation sets over the full state.

B Details about Learning Algorithm

B.1 Accuracy Experiment

Algorithm 1 is the pseudocode used for the experiments described in Section 4.1. Every episode, every option is executed from every start state in $\mathcal{S}_0$. The result of that execution is recorded as ground-truth $Y_{s,o}(t)$ and stored to later compute the size of the true initiation set $|Y| = \frac{\sum_{s,o,t} Y_{s,o}(t)}{|Y_{s,o}(t)|}$. If the start was predicted to be inside the initiation set by the learning algorithm, then the trajectory generated by rolling out the option policy is used to update the policy and the initiation learner (e.g, IVF, classifier). We report the agreement between the predicted initiations and the ground-truth as an accuracy measurement for that state-option pair. The fraction of start states in $\mathcal{S}_0$ that lead to success is reported as a measurement of the size of the true initiation set.

Algorithm 1 Accuracy/Size Experiment Procedure

Inputs: Option termination conditions $\beta_o, \forall o \in \mathcal{O}$, start states $\mathcal{S}_0$, number of episodes `n_episodes`.
Outputs: Accuracy table A and Initiation Size table S; both map state-option pairs to a list of booleans.

```
Initialize goal-conditioned policy π_θ : S × G → a ∈ A.
Initialize Initiation Value Function (IVF) V_φ : S × G → ℝ.
Initialize binary classifier for each option I_o(s; ψ) → {0, 1}.
Initialize replay buffers for Rainbow B_R and IVF B_I.
Initialize buffers to store positive and negative examples for each option's initiation classifier.
Initialize tables A and S as mapping each state-option pair to an empty list.
for episode ∈ range(n_episodes) do
  for start state s_0 ∈ S_0 do
    for option o ∈ O do
      Reset simulator to s_0.
      Record option o's initiation decision X = I_o(s_0; ψ).
      Rollout option policy π_o(s_0, g ~ β_o) to get trajectory τ and next state s'.
      Record whether the option policy reached the goal Y = s' ∈ β_o.
      Record accuracy A[s_0][o].append(𝟙(X = Y)).
      Update ground-truth size table S[s_0][o].append(Y).
      if predicted initiation X = 1 then
        Add trajectory τ to policy's replay B_R.
        Relabel trajectory τ with initiation cumulant c_0 : S → {0, 1}.
        Add relabeled trajectory to IVF's replay B_I.
        Add trajectory τ to o's positive/negative example buffer.
      end if
    end for
  end for
  Sample minibatch and update π_o using Rainbow.
  Sample minibatch and update V_φ using TD(0).
  for option o ∈ O do
    Compute weights w(s) for all training examples using Equation 1.
    Update o's initiation classifier by minimizing weighted cross-entropy loss.
  end for
end for
```

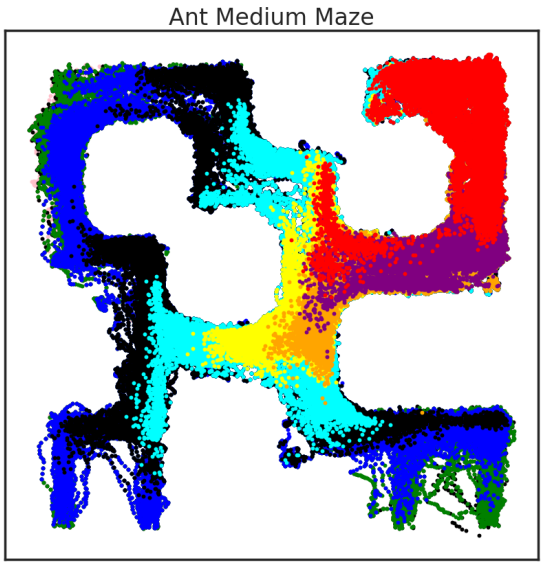


Figure 7: Initiation sets learned by deep skill chaining using the weighted classification approach in ANT MEDIUM MAZE. The task involves navigating the ant from the bottom-left to the top-right. Each color in the scatter plot denotes the initiation set of a different option; although the plot shows the location of the ant in the maze, the initiation set is learned using the full 30-dimensional state.

B.2 Robot Manipulation

The task-specific grasping problem is typically phrased as identifying grasp poses $g \in \mathbb{SE}(3)$ that afford task success. In practice, the difficulty of this problem is compounded by the fact that, for redundant manipulators, each grasp pose g yields an infinite number of corresponding arm configurations (solutions to the inverse kinematics problem). Explicitly, this relation is governed by the manipulator's forward kinematics $f : \mathcal{C} \to \mathbb{SE}(3)$ which maps (typically) 7-DoF configurations of the arm $q \in \mathcal{C}$ to poses in Cartesian space. In practice, only a subset of these configurations for a given grasp pose enable successful manipulation [Schiavi et al., 2022]. As a result, we task the initiation set learning algorithm with choosing start states directly from the space of arm configurations $\mathcal{C}$.

We generate collision-free grasp poses on each object using off-the-shelf grasp generation method GPG [Ten Pas et al., 2017] and corresponding arm poses using IKFLOW[Ames et al., 2022]. We chose to generate 50 grasp poses with 5 random inverse kinematics solutions each yielding a total of 250 starting configurations for each task.

Reward function. The reward functions are implemented as progress toward 1-DoF object joint position goals. The agent receives reward when the current joint position exceeds its previous maximum in a given episode.

Parameterization. As described in Section A, the agent receives proprioceptive and object-state observations and controls the manipulator's end-effector pose and impedance. The learning algorithm employed is TD3 [Fujimoto et al., 2018]. Goal-conditioning is omitted in these experiments as they have a single goal and a single option.

B.3 Deep Skill Chaining

Deep skill chaining (DSC) [Konidaris and Barto, 2009, Bagaria and Konidaris, 2020] proceeds recursively backward from the goal: the termination region of the first option is the task goal (Line 2, Algorithm 3); the agent first learns an option policy and initiation set for that option. Then, it learns

another option whose termination region is the initiation set of the first option; this process continues until there is some option whose initiation set covers the start state. The result is a collection of options that can be sequenced to reliably reach the goal.

Since the initiation sets of options are the subgoals for other options, the entire process is sensitive to the way in which the initiation sets are learned: poorly estimated initiation sets can lead to subgoals that do not improve the agent's abilty to reach the task goal.

Details about line 12 of Algorithm 2 differ based on the method used to learn the initiation set. When using the pure GVF approach, we perform as many minibatch gradient updates as the length of the option rollout; when using weighted classification, we recompute weights using Eq 1 for all training examples and then proceed to minimize weighted cross-entropy loss (3 epochs, batch size 128). When using classification (weighted or unweighted), we boost the contribution of the minority class by the ratio of the size of the majority class to that of the minority class.

Algorithm 2 Robust DSC Rollout

Inputs. Skill Chain $\mathcal{O}$
Hyperparameters. Option horizon H

1: Initialize empty trajectory buffer $\mathcal{B}$
2: **for** `each timestep` t **do**
3: Select option o using policy over options $\pi_{\mathcal{O}}(s_t)$
4: Sample a goal for selected option: $g \sim \beta_o$
5: Execute option policy $\pi_o(\cdot|g)$ in the environment
6: Add trajectory $\tau = \bigcup_{i=0}^{H-1}(s_i, o, a_i, s_{i+1}, g)$ to $\mathcal{B}$
7: **if** final state s_H reached goal g **then**
8: Add τ to o's list of positive examples
9: **else**
10: Add τ to o's list of negative examples
11: **end if**
12: Refit option o's initiation classifier
13: Add τ to replay buffer and update π_o using TD3
14: **end for**
15: **return** $\mathcal{B} = \bigcup_t (s_t, o_t, a_t, s_{t+1}, g_t)$

Algorithm 3 Robust DSC Algorithm

Inputs. Start state s_0, Goal region g.

1: Initialize global option o_G such that $\mathcal{I}_{o_G}(\cdot) = 1$
2: Initialize goal option o_g such that $\beta_{o_g} = g$
3: Initialize skill chain $\mathcal{O}$ with $\{o_g\}$
4: **for** `each episode` **do**
5: `transitions` $=$ `ROLLOUT`$(\mathcal{O})$
6: **if** $s_0 \notin \mathcal{I}_o, \forall o \in \mathcal{O}$ **then**
7: Create new option ω
8: Add ω to skill chain $\mathcal{O}$
9: **end if**
10: **end for**

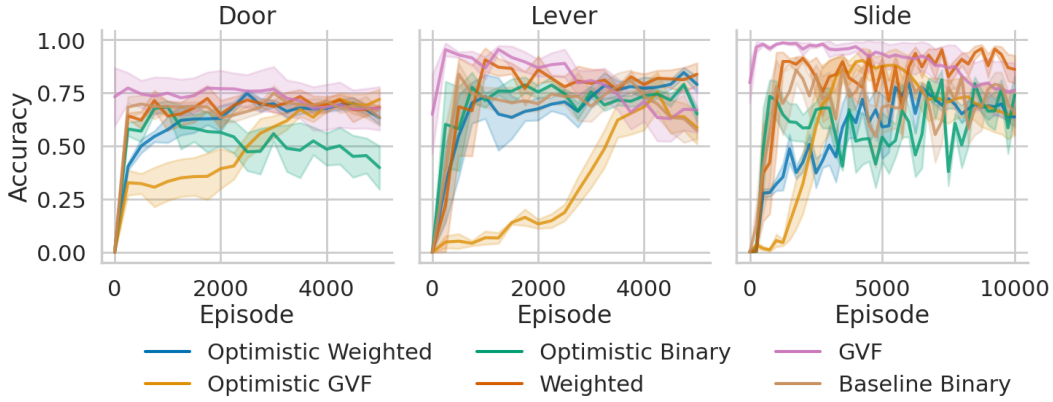
Picking a goal for option execution. Line 4 of Algorithm 2 samples a goal from the option's termination region. To implement this sampling procedure, we consider the option's parent ω in the chain (the parent option is the one whose initiation set is being targeted by the current option o). We enumerate the positive examples used to train $\mathcal{I}_\omega$ and pick the goal with the highest initiation probability under the current option. This process is done iteratively backward from the goal: the first goal is the task goal, the next one is the positive example closest (in terms of highest IVF value) to the task goal and so on.

Learned initiation sets. Figure 7 shows the initiation sets learned by DSC when using the weighted classification approach. Starting from the bottom-left of the maze, the agent successively targets the

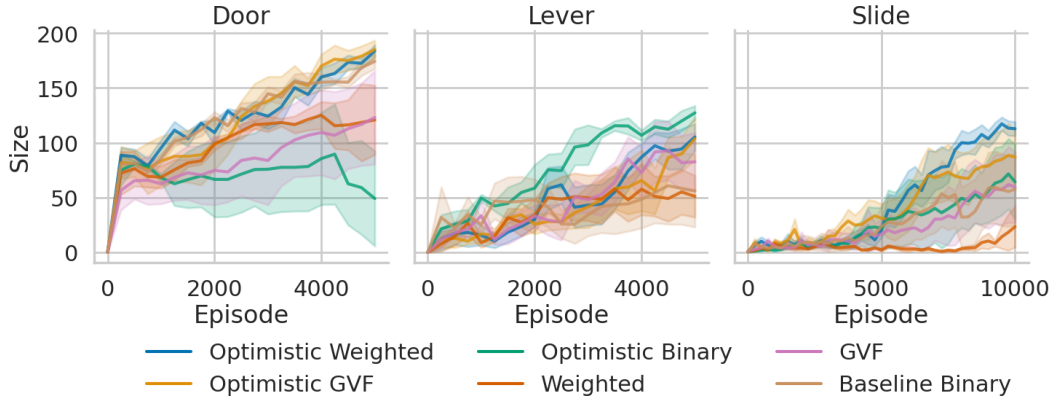
next option's initiation set, until it reaches the task goal at the top-right of the maze. The plot was generated by querying the learned initiation classifiers on the states in the agent's replay buffer at the end of training; only the (x, y) coordinates of those states are visualized.

C Additional Manipulation Experiments

Initiation set accuracy and true size are computed during training by performing an analogous procedure to Algorithm 1. Periodically, the manipulator was reset to each candidate start state and the initiation prediction was compared with the outcome of a policy rollout. Initiation set accuracy is visualized in Figure 8a. The methods generally converge to similar accuracy. True initiation set size is plotted in Figure 8b; size increases with optimism and correlates with success rates.



(a) Initiation set accuracy for manipulation domains.



(b) Initiation set size for manipulation domains (out of 250 start states).

Figure 8: (a) Accuracy of the learned initiation sets in the robot manipulation domains. (b) The size of the "true" initiation sets measured by performing Monte Carlo rollouts of the option policy.

D Hyperparameters

Rainbow was used for policy learning in Section 4.1, TD3 was used in the other experiments. Their hyperparameters (Tables 1 and 4) were not tuned and are either identical to the original paper implementation or borrowed from Bagaria et al. [2021a]. The bonus scale c (described in Sec 3.3) was tuned over the set $\{0.05, 0.1, 0.25, 0.5, 1.0\}$, the best performing hyperparameters are listed in Table 2.

Parameter	Value
Replay buffer size	10^6
Critic Learning rate	$3 \cdot 10^{-4}$
Actor Learning rate	$3 \cdot 10^{-4}$
Optimizer	Adam
Target Update Rate	$5 \cdot 10^{-3}$
Batch size	256
Iterations per time step	1
Discount Factor	0.99
Output Normalization	False

Table 1: TD3 Hyperparameters for Robosuite and DSC Experiments

Method	Bonus Scale
Optimistic Binary	0.1
Optimistic GVF	0.5
Optimistic Weighted	0.5

Table 2: Exploration Hyperparameters for Robosuite Experiments.

Parameter	Value
Replay buffer size	$3 \cdot 10^5$
Replay start size	1024
Learning rate	10^{-4}

Table 3: Rainbow Hyperparameters for Accuracy Experiments

Parameter	Value
Learning rate	10^{-4}
Optimizer	Adam
Replay buffer size	10^5
Batch size	32
Threshold	0.5
Target network update rate	$5 \cdot 10^{-3}$

Table 4: IVF Hyperparamters