A PROOFS AND ADDITIONAL ANALYSIS

A.1 DERIVING THE LOWER BOUND (PROOF OF LEMMA 1)

We first formulate the sampling procedure on starting states s_0, waypoints s_w, goals s_g and the corresponding time horizon variable t_1 and t_2. Then we derive a lower bound on the target log density of Eq. 2. We then show that optimizing the lower bound through an EM procedure is equivalent to breaking the goal-reaching task into a sequence of easier sub-problems. Finally, we wrapped up this section with a practical algorithm.

Data Generation Process. The generative model for which inference corresponds to our planning procedure can be formulated as follows. The episode starts by sampling an initial state $s_0 \sim p_0(s_0)$. Then it samples a geometric random variable $t_1 \sim Geom(1-\gamma)$ and roll out the policy $\pi(a \mid s, s_g)$ for exactly t_1 steps, starting from state s_0. We define s_w to be the state where we end up (i.e., $s_w \triangleq s_{t_1}$). Thus, s_w is sampled $s_w \sim p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_g)}(s_{t+} \mid s_0)$. We then sample another geometric random variable $t_2 \sim Geom(1-\gamma)$ and roll out the policy $\pi(a \mid s, s_g)$ for exactly t_1 steps, starting from state s_w. We define s_g to be the state where we end up (i.e., $s_g \triangleq s_{t_1+t_2}$). Thus, s_g is sampled $s_g \sim p_{Geom}^{\pi(\cdot|\cdot,s_g)}(s_{t+} \mid s_w)$. Note that the time index of the final state s_g is a sample from a negative binomial distribution: $t_1 + t_2 \overset{d}{=} NB(p = 1-\gamma, n = 2)$. We can equivalently express the sampling of s_g as $s_g \sim p_{NegBinom}^{\pi(\cdot|\cdot,s_g)}(s_{t+} \mid s_0)$.

Inference process. Under the formulation of the data generation process above, we then aim to answer the following question in the inference procedure: what intermediate states would a policy visit if it eventually reached the goal state s_g? Formally, we will estimate a distribution $q(s_w \mid s_0, s_g) \approx p(s_w \mid s_0, s_g)$.

We learn $q(s_w \mid s_0, s_g)$ by optimizing a evidence lower bound on our main objective (Eq. 2).

$$\log p_{\text{NEGBINOM}}^{\pi(\cdot|\cdot,s_g)}(s_{t+} = s_g \mid s_0) \tag{5}$$

$$= \log \int p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_g)}(s_{t+} = s_g \mid s_w) p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_g)}(s_w \mid s_0) ds_w \tag{6}$$

$$= \log \int p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_g)}(s_{t+} = s_g \mid s_w) p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_g)}(s_w \mid s_0) \frac{q(s_w \mid s_g, s_0)}{q(s_w \mid s_g, s_0)} ds_w \tag{7}$$

$$\geq \int q(s_w \mid s_g, s_0) \Big(\log p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_g)}(s_{t+} = s_g \mid s_w) + \tag{8}$$

$$\log p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_g)}(s_w \mid s_0) - \log q(s_w \mid s_g, s_0) \Big) ds_w \tag{9}$$

$$\triangleq \mathcal{L}(\pi, q(s_w \mid s_g, s_0)). \tag{10}$$

Note that s_g is conditionally independent of s_0 given s_w, so the $p^\pi(s_{t+} = s_g \mid s_w)$ terms on the RHS need not be conditioned on s_0. The evidence lower bound, $\mathcal{L}$, depends on two quantities: the goal-conditioned policy and the distribution over waypoints. The objective for the goal-conditioned policy is to maximize the probabilities of reaching the waypoint and reaching the final state. The objective for the waypoint distribution is to select waypoints s_w that satisfy two important properties: the current policy should have a high probability of successfully navigating from the initial state to the waypoint and from the waypoint to the final goal. Note that the optimal choice for the waypoint distribution automatically depends on the current capabilities of the goal-conditioned policy.

Before optimizing the lower bound, we introduce a subtle modification to the lower bound:

$$\mathcal{L}_2(\pi, q(s_w \mid s_g, s_0)) \triangleq \int q(s_w \mid s_g, s_0) \Big(\log p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_g)}(s_{t+} = s_g \mid s_w) + \tag{11}$$

$$\log p_{\text{GEOM}}^{\pi(\cdot|\cdot,\color{orange}{s_w})}(s_w \mid s_0) - \log q(s_w \mid s_g, s_0) \Big) ds_w. \tag{12}$$

The difference, highlighted in orange, is that the probability of reaching the waypoint is computed for a goal-conditioned policy that is commanded to reach that waypoint, rather than the final goal. We show that this new objective is also an evidence lower bound on the same goal-reaching objective

(Eq. 2), but modified such that the sequence of *commanded* goals is treated as an additional latent variable.

Assume that the initial state s_0 and the goal state s_g are given. As before, we want to find a policy that maximizes the probability of reaching s_g. However, we now consider jointly optimizing over both the policy and the sequence of goals we command for that policy. We use $s_c^{(t)}$ to denote the goal commanded at time t. We can write this optimization problem as follows:

$$\max_{\pi, s_c^{(1:\infty)}} \mathcal{F}(\pi, s_c^{(1:\infty)}) \triangleq \log p^{\pi(\cdot|\cdot, s_c^{(1:\infty)})}(s_{t+} = s_g \mid s_0). \tag{13}$$

Applying Jensen's inequality, we obtain a lower bound that looks similar to before:

$$\mathcal{F}(\pi, s_c^{(1:\infty)}) \geq \mathbb{E}_{q(s_w|s_g,s_0)} \left[\log p_{\text{GEOM}}^{\pi(\cdot|\cdot, s_c^{(1:\infty)})}(s_{t+} = s_g \mid s_w) + \log p_{\text{GEOM}}^{\pi(\cdot|\cdot, s_c^{(1:\infty)})}(s_w \mid s_0) - \log q(s_w \mid s_g, s_0) \right] \tag{14}$$

$$= \mathbb{E}_{q(s_w|s_g,s_0)} \left[\log p_{\text{GEOM}}^{\pi(\cdot|\cdot, s_c^{(t_w+1:\infty)})}(s_{t+} = s_g \mid s_w) + \log p_{\text{GEOM}}^{\pi(\cdot|\cdot, s_c^{(1:t_w)})}(s_w \mid s_0) - \log q(s_w \mid s_g, s_0) \right]. \tag{15}$$

In the second line, we introduce t_w as the time when the waypoint is reached. This allows us to clarify that the probability of reaching the waypoint only depends on the commanded goals through time t_w, $s_c^{(1:t_w)}$. This lower bound holds for any choice of the commanded waypoints, $s_c^{(1:\infty)}$. Directly optimizing for the sequence of waypoints is challenging for two reasons. First, it requires estimating the probability of commanding one goal but reaching any other state. Second, if we command one goal when trying to reach a different goal, then the goal conditioned policy may not learn to associate the commanded goal with the desired outcome. For both these reasons, we choose to not optimize the lower bound with respect to the commanded goals, but rather manually specify the commanded goals as follows:

$$s_c^{(1:t_w)} = s_w, s_c^{(t_w+1:\infty)} = s_g. \tag{16}$$

Thus, we have recovered the objective in Eq. 12. As our lower bound holds for any choice of commanded waypoint, it also holds for this choice.

A.2 The Optimal Waypoint Distribution (Proof of Lemma 2)

This section proves Lemma 2.

Proof. Recall that our goal is to solve the following maximization problem:

$$\max_{q(s_w|s_g,s_0)} \mathbb{E}_{q(s_w|s_g,s_0)} \left[\log p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_g)}(s_{t+} = s_g \mid s_w) + \log p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_w)}(s_w \mid s_0) - \log q(s_w \mid s_g, s_0) \right]. \tag{17}$$

Note that the waypoint distribution must integrate to one. The Lagrangian can be written as

$$\mathbb{E}_{q(s_w|s_g,s_0)} \left[\log p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_g)}(s_{t+} = s_g \mid s_w) + \log p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_w)}(s_w \mid s_0) - \log q(s_w \mid s_g, s_0) \right] + \tag{18}$$

$$\lambda \left(\int q(s_w \mid s_0, s_g) ds_w - 1 \right), \tag{19}$$

where λ is a Lagrange multiplier. We then take the derivative with respect to $q(s_w \mid s_g, s_0)$:

$$\frac{d}{dq(s_w \mid s_0, s_g)} = \frac{-q(s_w \mid s_0, s_g)}{q(s_w \mid s_0, s_g)} + \log p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_g)}(s_{t+} = s_g \mid s_w) + \tag{20}$$

$$\log p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_g)}(s_w \mid s_0) - \log q(s_w \mid s_g, s_0) + \lambda \tag{21}$$

$$= -1 + \log p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_g)}(s_{t+} = s_g \mid s_w) + \log p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_w)}(s_w \mid s_0) - \tag{22}$$

$$\log q(s_w \mid s_g, s_0) + \lambda. \tag{23}$$

We then set this derivative equal to zero and solve for $q(s_w \mid s_g, s_0)$:

$$q(s_w \mid s_g, s_0) = e^{\lambda - 1} p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_g)}(s_{t+} = s_g \mid s_w) p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_w)}(s_w \mid s_0).$$

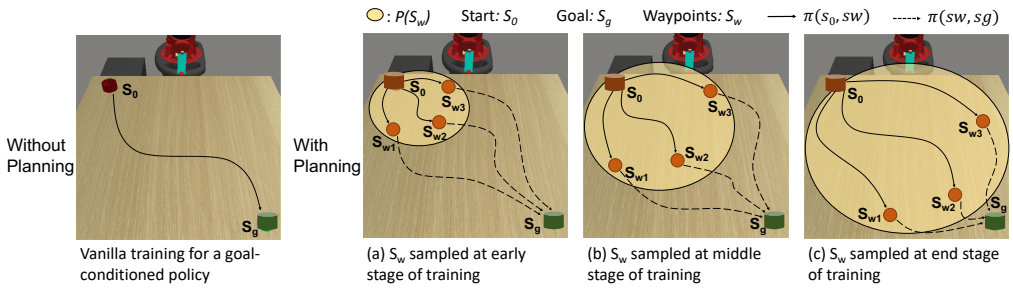


Figure 8: **C-Planning Curriculum**: Illustration of planning over waypoints distribution $p(\boldsymbol{s}_w)$. Goal-conditioned RL directly commands the agent to the final goal. C-planning first samples an intermediate waypoint $\boldsymbol{s}_w$ from $p(\boldsymbol{s}_w)$, directs the agent to that waypoint, and then commands the final goal after the agent has reached the waypoint. Note that the $p(s_w)$ is proportional to the state density of the current policy, so the high probability region will expand from starting state s_0 to the goal state s_g, as the agent explores the environment.

Finally, we determine the value of λ such that $q(s_w \mid s_0, s_g)$ integrates to one. We can then express the optimal waypoint distribution as follows:

$$q^*(s_w \mid s_g, s_0) = \frac{p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_g)}(s_g \mid s_w) p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_w)}(s_w \mid s_0)}{\int p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_g)}(s_g \mid s'_w) p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_w)}(s'_w \mid s_0) ds'_w}.$$

□

A.3 Estimating Importance Weights (Proof of Lemma 3)

This section proves Lemma 3.

Proof. Define the normalizing constant as follows

$$Z(s_0, s_g) = \frac{b(s_g)}{\int p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_g)}(s_g \mid s'_w) p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_w)}(s'_w \mid s_0) ds'_w}.$$

Substituting $Z(s_0, s_g)$ into the RHS of Eq. 4 and simplifying the result, we show that it equals the LHS of Eq. 4:

$$\frac{C_\theta(s_w, s_g)}{1 - C_\theta(s_w, s_g)} \frac{C_\theta(s_0, s_w)}{1 - C_\theta(s_0, s_w)} Z(s_0, s_g) \tag{24}$$

$$= \frac{C_\theta(s_w, s_g)}{1 - C_\theta(s_w, s_g)} \frac{C_\theta(s_0, s_w)}{1 - C_\theta(s_0, s_w)} \frac{b(s_g)}{\int p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_g)}(s_g \mid s'_w) p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_w)}(s'_w \mid s_0) ds'_w} \tag{25}$$

$$= \frac{p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_g)}(s_{t+} = s_g \mid s_w)}{\cancel{b(s_g)}} \frac{p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_w)}(s_{t+} = s_w \mid s_0)}{b(s_w)} \frac{\cancel{b(s_g)}}{\int p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_g)}(s_g \mid s'_w) p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_w)}(s'_w \mid s_0) ds'_w} \tag{26}$$

$$= \frac{p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_g)}(s_{t+} = s_g \mid s_w) p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_w)}(s_{t+} = s_w \mid s_0)}{\int p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_g)}(s_g \mid s'_w) p_{\text{GEOM}}^{\pi(\cdot|\cdot,s_w)}(s'_w \mid s_0) ds'_w} \frac{1}{b(s_w)} \tag{27}$$

$$= \frac{q(s_w \mid s_0, s_g)}{b(s_w)}. \tag{28}$$

□

B Additional Experiments

B.1 An Oracle Experiment

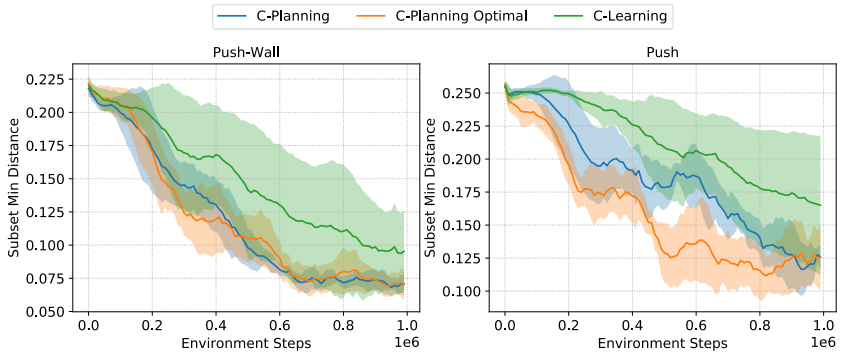
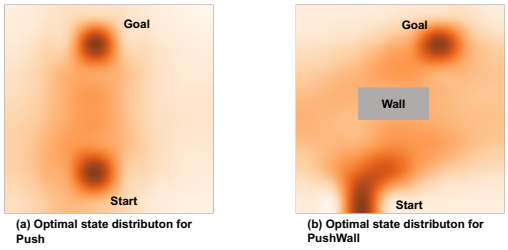


Figure 9: **Oracle experiment with the optimal generative model**: Goal-conditioned RL learns the task much faster if we can sample the waypoints from the expert policy's marginal state distribution (C-Planning Optimal). Compared to planning with optimal state density distribution, planning using a learned classifier (C-Planning) shows comparable performance.



(a) Optimal state distributon for Push

(b) Optimal state distributon for PushWall

Figure 10: *(Left)*An agent must navigate from the `start` state to the `goal` state. The heatmap visualizes the marginal state distribution of the optimal policy.

We run an experiment to confirm the well-known result [21] that the optimal initial state distribution for learning a task is the state distribution of that task's optimal policy. Intuitively, if we could sample exactly from the state distribution of the optimal policy, then an RL agent could learn to reach goals more quickly. This is not surprising as it resembles behavioral cloning of the optimal policy. In this oracle experiment, it is assumed we are given the access to an optimal policy. We achieve this by following the sampling procedure from Algorithm. 1, but replace the classifier C with that learned from the optimal policy C^*. We then perform RL to reach waypoints and then the final goal. Results in Fig. 9 show that using the optimal policy as the initial state distribution results in faster learning than using the original state distribution. We also visualize the state density distribution of the optimal policy to provide more qualitative intuition on how the algorithm works. Please refer to Appendix B.2 for more details.

In addition, we are also interested in measuring how well does planning through a learned model (C-Planning) performs if we don't have access to the optimal generative model. Results in Fig. 9 also show that C-Planning achieves performance comparable to planning via optimal generative model in the Push and the Push-Wall environment.

B.2 Visualization of State Density Map of Optimal policy

We conduct this experiment on a 2D navigation task shown in Fig. 9 (left), where we have also visualized the original initial state distribution, the state distribution of an optimal policy, and the goal state. To conduct this experiment, we apply a state-of-the-art goal-conditioned RL algorithm (C-learning) in the two settings with different initial state distributions. For fair evaluation, we evaluate the policies learned in both settings using the original initial state distribution. The results shown in Fig. 9 (right) show that starting from the optimal initial state distribution results in 2.4x faster learning.

B.3 Ablations to HER, SkewFit and SoRB (C-Planning)

We perform additional experiments, comparing C-Planning with HER [2], SkewFit [34] and SoRB [10] on C-Planning. HER barely works on only simple experiments (e.g., Reach) but fails on harder ones. Similar results are also been observed in the C-Learning [11] paper. We implement

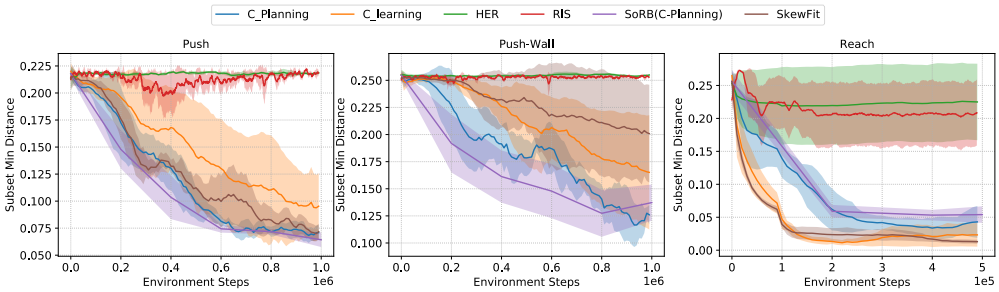


Figure 11: **Ablation with HER, SkewFit and SoRB**: Comparison of C-Planning to more baselines like HER, SkewFit and SoRB. HER only shows good performance on easy task like Reach but fails on harder ones. SkewFit shows a little better performance in Reach task but is a little worse in Push and Push-Wall. Additional ablation study on SoRB(C-Planning) shows small benefits are gained by adding SoRB at test time.

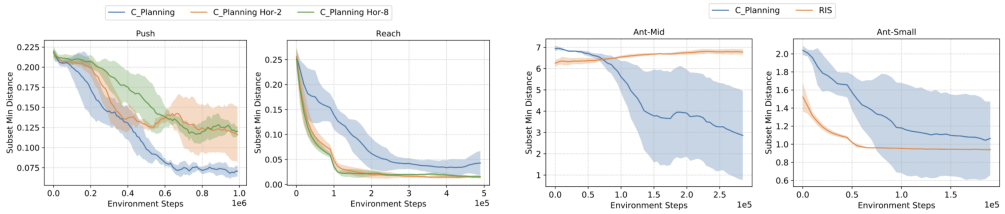


Figure 12: **Ablation on the Planning Horizon and Experiments on Ant-Maze**: (*left*) Albation study of the planning horion. A Larger planning horizon will sometimes help the performance of tasks, but the improvement is not consistent. (*right*) Experiments on small scale Ant-Maze. RIS manage to reach the goal quickly in Ant-Small environment but fails in the Ant-Mid.

a version of SkewFit algorithm on top of our framework. The only change it did is SkewFit samples sswaypoints according to the inverse of a state density model. We see that SkewFit has better performance in Reach task but fails in PushWall task. We also compare C-Planning with SoRB(C-Planning) which does SoRB at test time. Results show that SoRB(C-Planning) performs as well as C-Planning at test time.

B.4 Ablations on Planning Horizon

We perform an ablation study on the planning horizon (number of waypoints used when doing sampling). We implement this in a way of iterative planning: take horizon of 4 for example, we first planning the middle point s_{w2} via starting point and goal of (s_0, s_g); then planning s_{w1} via starting point and goal (s_0, s_{w2}). Results show that planning for a more fine-grained fashion (larger number of waypoints) is not always helping. In the Fig. 12 left, in Reach task, planning over more waypoints helps but it decreases the performance in Push task. Note that here the number of waypoints is used for planning, while in Fig. 5 the number of waypoints is how many waypoints to reach before changing to the final goal. These two are different quantities.

B.5 Additional Experiments on Ant-Maze

We performance an additional experiments in the Ant-Maze environment. We design a very simple environment: command the ant to a specific goal position with a short (Ant-Small) or a long (Ant-Mid) distance. The distance between starting point and goal is 2.5 and 7.5 repectively for Ant-Small and Ant-Mid. The only two modifications we did comparing to the original RIS [5] is change the maximum time horizon to 300 (RIS uses max time steps of 600) and reset the agent to a fixed area (RIS resets the position of the agent uniformly). In Fig. 12 right, RIS achieves better performance in Ant-Small environment while fails in Ant-Mid environment. Note that C-Planning doesn't use any termination function and reward function while RIS heavily relies on them.

C EXPERIMENTAL DETAILS

In this section, We provide the essential hyperparameters for reproducing our experiments in this section. We also introduce the hyperparameters used in baselines and provide a detailed description of environmental design.

C.1 IMPLEMENTATION DETAILS

We introduce the hyperparameters used in C-Planning. Note that in C-Learning, only a classifier on $(\boldsymbol{s}_t, \boldsymbol{s}_{t+}, \boldsymbol{a}_t)$ is needed. In C-Planning, in order to sample waypoints from the distribution, an additional classifier on $(\boldsymbol{s}_t, \boldsymbol{s}_{t+})$ is needed. We also introduce two hyperparameters: Maximum Steps Reaching Goal (N_g in Alg.1) forces the agent to change the intermediate goal if the original goal hasn't been reached for some steps; Distance Threshold Reaching Goal (ϵ_d in Alg.1) for determing whether a goal is reached or not. The rest are the standard hyperparameters for SAC algorithm and we list here for reference.

Table 1: Hyperparameters used for C-Planning in all the environments in MetaWorld.

	Hyperparameter Value
Actor lr	0.0003
Action-State Critic lr	0.0003
State Critic lr	0.00003
Actor Network Size	(256, 256, 256)
Critic Network Size	(256, 256, 256)
Maximum Steps Reaching Goal	20
Distance Threshold Reaching Goal	0.05
Actor Loss Weight	1.0
Critic Loss Weight	0.5
Discount	0.99
Target Update Tau	0.005
Target Update Period	1
Number Waypoints	5
Goal Relabel Next	0.3
Goal Relabel Future	0.2

Note that we use a slightly different hyperparameters for 2D maze environment and we list below, all the other hyperparameters remains the same. Note that we follow the goal relabeling changes by C-Learning [11].

Table 2: Hyperparameters used for C-Planning in all the environments in 2D navigation maze.

	Hyperparameter Value
Distance Threshold Reaching Goal	1.0
Number Waypoints	8
Goal Relabel Next	0.5
Goal Relabel Future	0.0

C.2 ENVIRONMENTS

We follow the envioronment design of [11] with only one noticeable difference: in the original environments of C-Learning, for the ease of training, the author set the initial position of objects to be relatively near the arm so the arm can easily push the object, getting a better learning signal. We intentionally set the initial state of object to be far away from the arm. This significantly increase the diffuculty of learning. We'll release these environment with the code.

Table 3: Max number of time steps used for each environment.

	Max Time Steps
Spiral 9x9	200
Spiral 11x11	200
Maze 11x11	200
Sawyer Push	50
Sawyer Reach	50
Sawyer Push-Wall	50
Obstacle-Drawer-Close	150
Obstacle-Drawer-Open	150
Sawyer Push-Two	150

C.3 BASELINES

We also provide the hyperparameters associated with the baselines. The two baselines we care about: C-Learning and RIS. We summarize their hyperparameters in the table:

Table 4: Hyperparameters used for C-Learning in all the environments in MetaWorld.

	Hyperparameter Value
Actor lr	0.0003
Action-State Critic lr	0.0003
Actor Network Size	(256, 256, 256)
Critic Network Size	(256, 256, 256)
Actor Loss Weight	1.0
Critic Loss Weight	0.5
Discount	0.99
Target Update Tau	0.005
Target Update Period	1
Goal Relabel Next	0.3
Goal Relabel Future	0.2

Table 5: Hyperparameters used for RIS in all the environments in MetaWorld.

	Hyperparameter Value
epsilon	0.0001
Replay Buffer Goals	0.5
Distance Threshold	0.05
Alpha	0.1
Lambda	0.1
H lr	0.0001
Q lr	0.001
Pi lr	0.0001
Encoder lr	0.0001

D VISUALIZATION OF THE LEARNED POLICY

In order to more intuitively visualize the behavior of the learned policy and emphasize the importance of the difficulty of the learning task, we plot the visualization of our learned policy for several snap shots in an episode for environment of `Push-Two` and `Obstacle-Drawer-Open`. We see that our method successfully guide the agent to learn the behavior of push the green object first, the push the object; And first open the drawer then push the object to the desired location. We would like to emphasize that prior baselines all fail to demonstrate such behavior without any offline data, expert demonstration and reward shaping. The successfully learning of such behavior enables the robot to do more complex tasks without any human intervention.

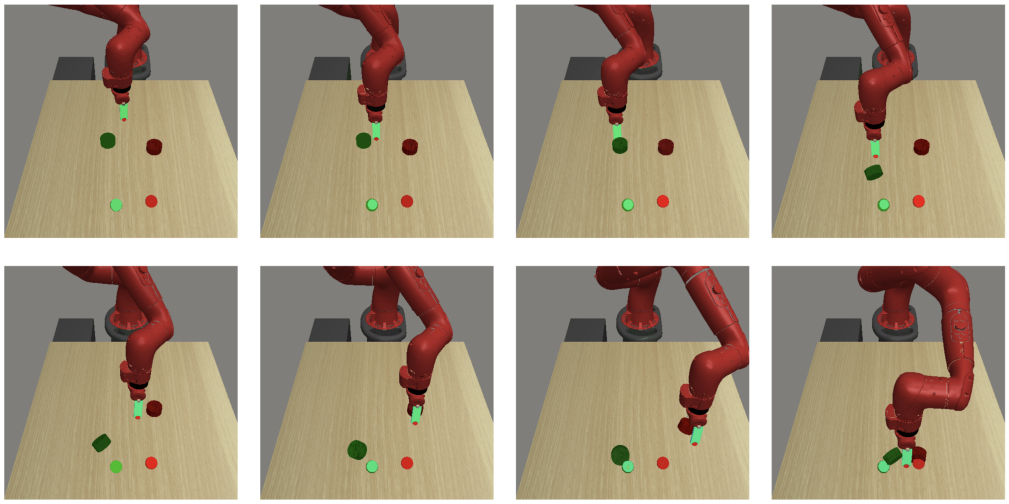


Figure 13: Visualization of the trained policy for the `Push-Two` environment. Our method successfully trains the agent to manipulate the two object in the sequential manner without any offline data, expert demonstration and reward shaping. Prior baselines all fail to demonstrate such behavior.

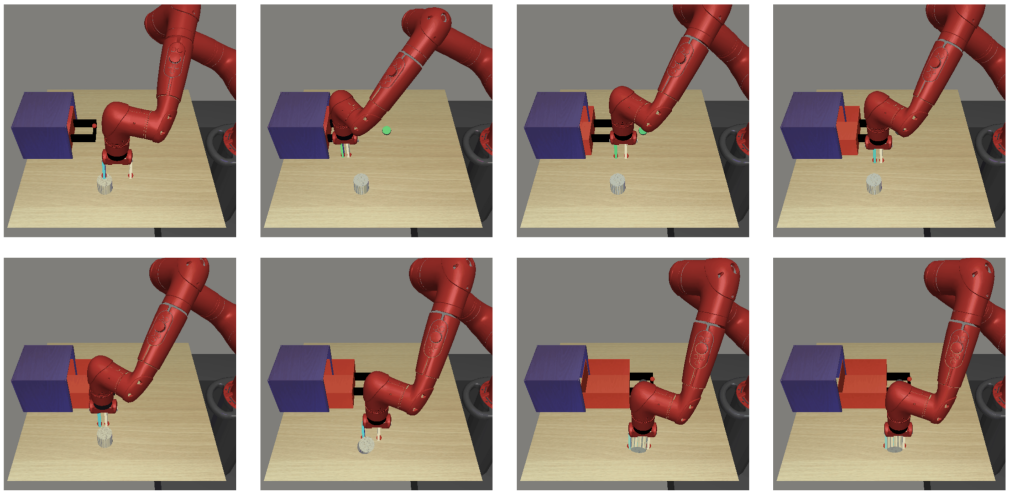


Figure 14: Visualization of the trained policy for the `Obstacle-Drawer-Open` environment. Our method successfully trains the agent to first close the drawer and then push the object to the desired location. We emphasize that such behavior is hard to obtain and most of the prior methods only manage to close the drawer.