C-PLANNING: AN AUTOMATIC CURRICULUM FOR LEARNING GOAL-REACHING TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Goal-conditioned reinforcement learning (RL) has shown great success recently at solving a wide range of tasks(e.g., navigation, robotic manipulation). However, learning to reach distant goals remains a central challenge to the field, and the task is particularly hard without any offline data, expert demonstrations, and reward shaping. In this paper, we propose to solve the distant goal-reaching task by using search at training time to generate a curriculum of intermediate states. Specifically, we introduce the algorithm Classifier-Planning (C-Planning) by framing the learning of the goal-conditioned policies as variational inference. C-Planning naturally follows expectation maximization (EM): the E-step corresponds to planning an optimal sequence of waypoints using graph search, while the M-step aims to learn a goal-conditioned policy to reach those waypoints. One essential difficulty of designing such an algorithm is accurately modeling the distribution over waypoints to sample from. In C-Planning, we propose to sample the waypoints using contrastive methods to learn a value function. Unlike prior methods that combine goal-conditioned RL with graph search, ours performs search only during training and not testing, significantly decreasing the compute costs of deploying the learned policy. Empirically, we demonstrate that our method not only improve the sample efficiency of prior methods, but also successfully solves temporally-extended navigation and manipulation tasks, where prior goal-conditioned RL methods (including those based on graph search) fail to solve.

1 INTRODUCTION

Compared to general RL, which tries to maximize the accumulated reward, goal-conditioned RL methods aim to learn to reach any goal. Many tasks can be specified as goal-reaching RL problems, and prior work has shown how such goal-reaching agents can be learned in a self-supervised manner. While many goal-conditioned RL algorithms successfully learn policies that reliably reach nearby goals, learning to achieve distant goals remains a complex problem. Some prior methods approach this problem by performing search or optimization over subgoals at test time. However, these test-time planning methods either rely on graph search (Savinov et al., 2018; Eysenbach et al., 2019), which scales poorly with dimensionality (Hsu et al., 2006), or continuous optimization over subgoals (Nasiriany et al., 2019), which is expensive and can result in model exploitation.

In this paper, we take a different tack and instead use search at training time to automatically generate a curriculum. Our method effectively teaches a standard goal-conditioned policy to reach even distant goals. When training the agent to reach one goal, our method first samples intermediate waypoints enroute to that goal. Then it commands the agent to reach those waypoints before navigating to the final destination. Running policy in this manner improves the quality of data, allowing the agent to learn to reach distant goals. Importantly, such a method does not perform planning at test time, decreasing the computational demands for deploying the learned agent.

Our curriculum does not require manual engineering or prior knowledge of the tasks but rather emerges automatically from formulating of goal-conditioned RL as variational inference. Our resulting method performs expectation maximization (EM) to maximize a lower bound on the probability of reaching the goal. The M-step corresponds to a prior method for goal-conditioned RL, while the E-step corresponds to graph search. Thus, EM presents a marriage between previous goal-conditioned RL algorithms and graph-search methods.

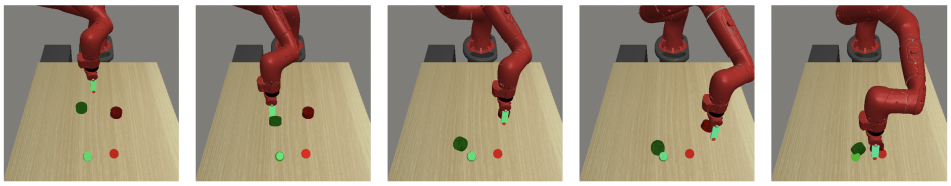

Figure 1: **C-Planning** is an algorithm for goal-conditioned RL that uses an automatic curriculum of waypoints to learn policies that can solve complex tasks, such as manipulating multiple objects in sequence to achieve a goal, without requiring any reward functions, manual distance functions, or human demonstrations.

The main contribution of this work is a goal-conditioned RL algorithm, C-Planning, that excels at reaching distant goals. Our method uses an automatic curriculum of subgoals to accelerate training. We show that our curriculum emerges from applying variational inference to the goal-reaching problem. Unlike prior work, our method does not require performing graph search or optimization at test-time. Empirically, C-Planning not only matches but surpasses the performance of these prior search-based methods, suggesting that it is not just amortizing the cost of graph search. We empirically evaluate C-Planning on temporally-extended 2D navigation tasks and complex robotic manipulation tasks. C-Planning improves the sample efficiency of prior goal-conditioned RL algorithms and manages to solve more difficult manipulations tasks such as rearranging multiple objects in sequence (see Fig. 1 for some snapshots from a policy trained by C-Planning). To the best of our knowledge, such complex behavior of manipulating multiple objects has not been shown in prior methods without any human demonstrations or reward shaping.

2 Related Work

The problem of learning to achieve goals has a long history, both in the control community (e.g., Lyapunov (1992)) and the RL community (Kaelbling, 1993). Many prior papers approach goal-conditioned RL as a reward-driven, multi-task learning problem, assuming access to a goal-conditioned reward function. One unique feature of goal-conditioned RL, compared to other multi-task RL settings, is that it can also be approached using reward-free methods. Reward-free approaches include goal-conditioned behavior cloning (Ding et al., 2019; Lynch et al., 2020; Gupta et al., 2020; Savinov et al., 2018) and RL methods that employ hindsight relabeling (Lin et al., 2019; Eysenbach et al., 2020; Schroecker & Isbell, 2020; Kaelbling, 1993). While goal-conditioned behavior cloning methods are simple to implement and have shown excellent results on a number of real-world settings (Shah et al., 2020; Meng et al., 2020), they are not guaranteed to recover the optimal policy without additional assumptions (e.g., determinism, online data collection). While both methods excel at certain control tasks, both often struggle to solve tasks with longer horizons.

To solve longer-horizon tasks, prior work has combined goal-conditioned RL with graph search, noting that goal-conditioned value functions can be interpreted as dynamical distances (Savinov et al., 2018; Eysenbach et al., 2018; Nasiriany et al., 2019; Huang et al., 2019). These prior methods typically proceed in two phases: the first phase learns a goal-conditioned policy, and the second phase combines that policy with graph search. While these methods have demonstrated excellent results on a number of challenging control tasks, including real-world robotic navigation (Shah et al., 2020; Meng et al., 2020), the stage-wise approach has a few limitations. First, the post-hoc use of graph search means that graph search cannot improve the underlying goal-conditioned policy. It is well known that the performance of RL algorithms is highly dependent on the quality of the collected data (Kakade & Langford, 2002; Levine et al., 2020; Kumar et al., 2020). By integrating search into the learning of the underlying goal-conditioned policy, our method will improve the quality of the data used to train that policy. A second limitation of prior methods that defer graph search to test time is the cost of deployment: choosing actions using graph search requires at least $\mathcal{O}(|\mathcal{V}|)$ queries to a neural network.[1] In contrast, our method performs search at training time rather than test time. While this design decision does increase the computational complexity of training, it significantly decreases the latency at deployment. Our method is also related to RIS (Chane-Sane et al., 2021), which similarly performs search during training instead of deployment. Our method differs from RIS in how search is used: whereas RIS modify the *objective*, our method uses search to modify the *data*, a subtle distinction that avoids favoring the learned policy and unifies the objectives for RL and graph search. We demonstrate the importance of this difference in our experiments.

[1] While computing all pairwise distances requires $\mathcal{O}(|\mathcal{V}|^2)$ time, only $|\mathcal{V}|$ edges change at each time step. The cost of computing the remaining edges can be amortized across time.

Effectively solving goal-conditioned RL problems requires performing good exploration. In the goal-conditioned setting, the quality of exploration depends on how goals are sampled, a problem studied in many prior methods (Florensa et al., 2018; 2017; Ren et al., 2019; Eysenbach et al., 2018; Pong et al., 2020; Zhang et al., 2020; Pitis et al., 2020; Zhang et al., 2021). These methods craft objectives that try to optimize for learning progress, and the resulting algorithms achieve good results across a range of environments. Our method differs from these methods in that the method for optimizing the goal-conditioned policy and the method for sampling waypoints are jointly optimized using the same objective. We demonstrate the importance of this difference in our experiments.

In terms of its theoretical foundations, our method builds on the idea introduced in many prior works that reinforcement learning can be cast as inference problems (Todorov, 2007; Levine, 2018; Ziebart, 2010; Rawlik et al., 2013; Theodorou et al., 2010; Attias, 2003). The observation that variational inference for certain problems corresponds to graph search is closely related to Attias (2003). Our work extends this inferential perspective to hierarchical models.

3 PRELIMINARIES

We introduce the goal-conditioned RL problem and a recent goal-conditioned RL algorithm, C-learning, upon which our method builds.

Goal-Conditioned RL. We focus on controlled Markov processes defined by states $\boldsymbol{s}_t \in \mathcal{S}$ and actions $\boldsymbol{a}_t$. The initial state is sampled $\boldsymbol{s}_0 \sim p_0(\boldsymbol{s}_0)$ and subsequent states are sampled $\boldsymbol{s}_{t+1} \sim p(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{a}_t)$. Our aim is to navigate to the goal states, $\boldsymbol{s}_g \in \mathcal{S}$, which are sampled from $\boldsymbol{s}_g \sim p_g(\boldsymbol{s}_g)$. Define a policy $\pi(\boldsymbol{a}_t \mid \boldsymbol{s}_t, \boldsymbol{s}_g)$ conditioned on both the current state and the goal. The objective is to maximize the probability (density) of reaching the goal in the future.

To define the probability, we must indicate *when* we would like the agent to reach the goal. Formally, let $\Delta \in \mathbb{N}^+$ be a random integer indicating when we reach the goal. The user specifies a prior $p(\Delta)$. Note that *all* RL papers implicitly choose a prior, with most (implicitly) using $p(\Delta) = \textsc{Geom}(1-\gamma)$. This geometric distribution is ubiquitous because it is easy to incorporate into temporal difference learning, not because it is a particularly reasonable prior belief on when the agent will solve the task or when the episode will terminate. Prior work has also considered non-geometric distributions (Fedus et al., 2019). In this paper, we propose a method where the prior $p(\Delta) = \textsc{NegBinom}(p = 1-\gamma, n = 2)$ is a negative binomial distribution. Given a prior $p(\Delta)$, we define the distribution over future states as

$$p^{\pi(\cdot|\cdot,\boldsymbol{s}_g)}_{p(\Delta)}(\boldsymbol{s}_{t+} = \boldsymbol{s}_{t+} \mid \boldsymbol{s}_t, \boldsymbol{a}_t) = \mathbb{E}_{p(\Delta)}\left[p^{\pi(\cdot|\cdot,\boldsymbol{s}_g)}_{\Delta}(\boldsymbol{s}_{t+\Delta} = \boldsymbol{s}_{t+} \mid \boldsymbol{s}_t, \boldsymbol{a}_t)\right], \tag{1}$$

where $p^{\pi}_{\Delta}(\boldsymbol{s}_{t+\Delta} \mid \boldsymbol{s}_t, \boldsymbol{a}_t)$ is the probability density of reaching state $\boldsymbol{s}_{t+\Delta}$ exactly Δ steps in the future when sampling actions from $\pi(\boldsymbol{a}_t \mid \boldsymbol{s}_t, \boldsymbol{s}_g)$. Note that, $p^{\pi(\cdot|\cdot,s_g)}_{\textsc{Geom}(1-\gamma)}$ denotes the distribution over future states that are sampled according to a Geometric distribution over time, corresponding to the discounted state occupancy measure used in prior work (Ho & Ermon, 2016; Nachum et al., 2019). Our objective for goal-reaching is to maximize the probability of reaching the desired goal:

$$\max_{\pi} \mathbb{E}_{p_g(\boldsymbol{s}_g)}\left[p^{\pi(\cdot|\cdot,\boldsymbol{s}_g)}_{\textsc{NegBinom}(p=1-\gamma,n)}(\boldsymbol{s}_{t+} = \boldsymbol{s}_g)\right]. \tag{2}$$

C-Learning. Our method builds upon a prior method for goal-conditioned RL, C-Learning (Eysenbach et al., 2020). C-Learning learns a classifier for predicting whether a state $\boldsymbol{s}_t$ comes from a future state density $p^{\pi(\cdot|\cdot,\boldsymbol{s}_g)}_{p(\Delta)}(\boldsymbol{s}_{t+} = \boldsymbol{s}_{t+} \mid \boldsymbol{s}_t, \boldsymbol{a}_t)$ or a marginal state density $p^{\pi(\cdot|\cdot,\boldsymbol{s}_g)}_{p(\Delta)}(\boldsymbol{s}_{t+} = \boldsymbol{s}_{t+}) = \int p^{\pi(\cdot|\cdot,\boldsymbol{s}_g)}_{p(\Delta)}(\boldsymbol{s}_{t+} = \boldsymbol{s}_{t+} \mid \boldsymbol{s}_t, \boldsymbol{a}_t)\, p(\boldsymbol{s}_t, a_t)\, d\boldsymbol{s}_t\, d\boldsymbol{a}_t$. We can write the Bayes-optimal classifier:

$$C_\theta(\boldsymbol{s}, \boldsymbol{s}_g) = \frac{p^{\pi(\cdot|\cdot,\boldsymbol{s}_g)}_{\textsc{Geom}}(\boldsymbol{s}_g \mid \boldsymbol{s})}{p^{\pi(\cdot|\cdot,\boldsymbol{s}_g)}_{\textsc{Geom}}(\boldsymbol{s}_g \mid \boldsymbol{s}) + p^{\pi(\cdot|\cdot,\boldsymbol{s}_g)}_{\textsc{Geom}}(\boldsymbol{s}_g)}. \tag{3}$$

In C-Learning, this classifier acts as a value function for training the policy. Our method will use this classifier not only to update the policy, but also to sample waypoints.

4 C-PLANNING: GOAL-CONDITIONED RL WITH PLANNING

We now present our method, C-Planning. The main idea behind our method is to decompose the objective of Eq. 2 into a sequence of easier goal-reaching problems. This makes the training easier since we could have more training signals for the more manageable goal-reaching tasks. The series of easier training tasks act as a curriculum. Such an idea of the decomposition is shown in Fig. 2. Note that the distribution of waypoints to sample from is policy dependent, thus changing over training iterations.

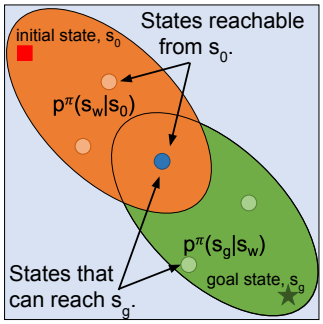

Figure 2: **Waypoint Sampling**: Our method samples waypoints that are reachable from the initial state, and from which the agent can reach the goal. This maximizes a lower bound on goal reaching objective.

To provide a high-level algorithm summary, our method consists of two steps: sampling a waypoint and command the agent to that; then we perform standard policy optimization based on the data collected. The main challenge lies in the first step, as modeling the distribution over waypoints is difficult. We adopt the idea of importance sampling: we first sample a batch of waypoints from the replay buffer and then put an importance weight based on the classifier trained by the procedure similar to C-Learning. The complete algorithm is presented in Alg. 1 and we will focus on derives our method using the tools of variational inference in this section.

4.1 Planning and Variational Inference

If we want a sub-optimal agent to navigate from some initial state $\boldsymbol{s}_0$ to a distant goal state $\boldsymbol{s}_g$, having the agent try to reach that state again and again will likely prove futile. Instead, we can command a sequence of goals, like a trail of breadcrumbs leading to the goal $\boldsymbol{s}_g$. To select these waypoints, we will command the policy to reach those states it would visit if it were successful at reaching $\boldsymbol{s}_g$. Sampling subgoals in this way can ease the difficulty of training the agent.

To formally derive our method for waypoint sampling, we cast the problem of goal-reaching as a latent variable problem. We assume that the agent starts at state $\boldsymbol{s}_0$ and ends at state $\boldsymbol{s}_g$, but the intermediate states that the agent will visit are unknown latent variables. This problem resembles standard latent variable modeling problems (e.g., VAE), where the intermediate state serves the role of the inferred representation. We can thus derive an evidence lower bound on the likelihood of reaching the desired goal:

Lemma 1. *The objective $\mathcal{L}$, defined below, is a lower bound on the goal-reaching objective (Eq. 2):*

$$\log p^{\pi(\cdot|\cdot,\boldsymbol{s}_g)}_{\text{NegBinom}}(\boldsymbol{s}_{t+} = \boldsymbol{s}_g \mid \boldsymbol{s}_0) \geq \mathbb{E}_{q(\boldsymbol{s}_w|\boldsymbol{s}_g,\boldsymbol{s}_0)}\Big[\log p^{\pi(\cdot|\cdot,\boldsymbol{s}_g)}_{\text{Geom}}(\boldsymbol{s}_{t+} = \boldsymbol{s}_g \mid \boldsymbol{s}_w) + \log p^{\pi(\cdot|\cdot,\boldsymbol{s}_g)}_{\text{Geom}}(\boldsymbol{s}_w \mid \boldsymbol{s}_0) - \log q(\boldsymbol{s}_w \mid \boldsymbol{s}_g, \boldsymbol{s}_0)\Big] \triangleq \mathcal{L}(\pi, q(\boldsymbol{s}_w \mid \boldsymbol{s}_g, \boldsymbol{s}_0)).$$

See Appendix A.1 for a full proof. The lower bound, $\mathcal{L}$, depends on two quantities: the goal-conditioned policy and an *inferred* distribution over waypoints, $q(\boldsymbol{s}_w \mid \boldsymbol{s}_0, \boldsymbol{s}_g)$. This bound holds for any choice of $q(\boldsymbol{s}_w \mid \boldsymbol{s}_0, \boldsymbol{s}_g)$, and we can optimize this lower bound with respect to this waypoint distribution. We will perform expectation maximization (Dempster et al., 1977) to optimize the lower bound, alternating between computing the optimal waypoint distribution and optimizing the goal-conditioned policy using this waypoint distribution. We then describe these two steps in detail.

E-Step. The E-step estimates the waypoint distribution $q(\boldsymbol{s}_w \mid \boldsymbol{s}_0, \boldsymbol{s}_g)$. The three terms in the lower bound indicate that the sampled waypoint should be reachable from the initial state, the goal state should be reachable from the sampled waypoint, and that the waypoint distribution should have high entropy. Thus, waypoint sampling resembles planning: we aim to minimize the length of the path to the goal, where distances are measured using log probabilities. Importantly, reachability is defined in terms of the capabilities of the current policy. We can analytically solve for the optimal waypoint distribution:

Lemma 2. *The optimal choice for the waypoint distribution satisfies:*

$$q^*(\boldsymbol{s}_w \mid \boldsymbol{s}_g, \boldsymbol{s}_0) = \frac{p^{\pi(\cdot|\cdot,\boldsymbol{s}_g)}_{\text{Geom}}(s_g \mid \boldsymbol{s}_w) p^{\pi(\cdot|\cdot,\boldsymbol{s}_w)}_{\text{Geom}}(s_w \mid \boldsymbol{s}_0)}{\int p^{\pi(\cdot|\cdot,\boldsymbol{s}_g)}_{\text{Geom}}(\boldsymbol{s}_g \mid \boldsymbol{s}'_w) p^{\pi(\cdot|\cdot,\boldsymbol{s}_w)}_{\text{Geom}}(\boldsymbol{s}'_w \mid \boldsymbol{s}_0) d\boldsymbol{s}'_w}.$$

See Appendix A.1 for a full derivation. We illustrate this waypoint distribution in Fig. 2. In general, accurately estimating the optimal distribution q^* is challenging. However, in the next section, we develop a simple algorithm that only requires learning a classifier instead of a generative model.

M-step. The M-step optimizes the lower bound with respect to the goal-conditioned policy. We ignore the $-\log q(\boldsymbol{s}_w \mid \boldsymbol{s}_0, \boldsymbol{s}_g)$ term, which does not depend on the policy. This lower bound looks like the sum of two goal-conditioned RL objectives, with a subtle but important difference in how the trajectories are sampled. When collecting data, a standard goal-conditioned RL algorithm initially samples a goal and then collects a trajectory where the policy attempts to reach that goal. Our method collects data in a different manner. The agent samples a goal, *then samples an intermediate waypoint that should lead to that goal.* The agent attempts to reach the waypoint *before* attempting to reach the goal. After the trajectory has been collected, the policy updates are the same as a standard goal-conditioned RL algorithm.

Two mental models. The algorithm sketch we have presented, which will be fleshed out in the subsequent section, can be interpreted in two ways. One interpretation is that the method performs a soft version of graph search during exploration, using a distance function of $d(\boldsymbol{s}_1, \boldsymbol{s}_2) = \log p_{\text{GEOM}}^{\pi(\cdot \mid \cdot, \boldsymbol{s}_2)}(\boldsymbol{s}_{t+} = \boldsymbol{s}_2 \mid \boldsymbol{s}_1)$. Thus, our waypoints sampling procedure corresponds to performing search using distance function $d(\boldsymbol{s}_0, \boldsymbol{s}_g) = \log p_{\text{GEOM}}^{\pi(\cdot \mid \cdot, \boldsymbol{s}_w)}(\boldsymbol{s}_{t+} = \boldsymbol{s}_w \mid \boldsymbol{s}_0) + \log p_{\text{GEOM}}^{\pi(\cdot \mid \cdot, \boldsymbol{s}_g)}(\boldsymbol{s}_{t+} = \boldsymbol{s}_g \mid \boldsymbol{s}_w)$. From this perspective, the method is similar to prior work that performs graph search during evaluation (Savinov et al., 2018; Eysenbach et al., 2019). However, whereas these prior methods perform search at test-time, ours will perform search at training time. We did extensive ablation on this in Sec. 5, showing our method is significantly better than test-time search.

The second interpretation focuses on the task of reaching the goal from different initial states. The choice of waypoint distribution, and the policy's success at reaching those waypoints, determines the initial state distribution for this task. Intuitively, the best initial state distribution is one that sometimes starts the agent at the true initial state $\boldsymbol{s}_0$, sometimes starts the agent at the final state $\boldsymbol{s}_g$, and sometimes starts the agent in between. In fact, prior work has formally shown that the optimal initial state distribution is the marginal state distribution of the optimal policy (Kakade & Langford, 2002). Under somewhat strong assumptions,[2] this is precisely what our waypoint sampling achieves. We refer the reader for the discussion and experiments in Appendix. A.

4.2 A Practical Implementation

We describe how we implement the policy updates (M-step) and waypoint sampling (E-step). For the policy updates, we simply apply C-learning (Eysenbach et al., 2020). The main challenge is waypoint sampling, which requires sampling a potentially high-dimensional waypoint distribution. Naively, one could fit a density model for estimating this with the observed states and goals. Alternatively, we implement waypoint sampling using importance sampling and a learned classifier.

Let $b(\cdot)$ be the background distribution, which we take to be the replay buffer. Our algorithm corresponds to the following two-step procedure: First sample a batch of waypoints from the buffer; Then sample one waypoint from within that batch using the normalized importance weights, $\frac{q(\boldsymbol{s}_w \mid \boldsymbol{s}_g, \boldsymbol{s}_0)}{b(\boldsymbol{s}_w)}$. We estimate these importance weights using the classifier learned by C-learning (Eq. 3). In this way, we avoid training an extra density model for fitting the distribution of waypoints.

Lemma 3. *We can write the importance weights in terms of this value function*

$$\frac{q(\boldsymbol{s}_w \mid \boldsymbol{s}_g, \boldsymbol{s}_0)}{b(\boldsymbol{s}_w)} = \frac{C_\theta(\boldsymbol{s}_w, \boldsymbol{s}_g)}{1 - C_\theta(\boldsymbol{s}_w, s_g)} \frac{C_\theta(\boldsymbol{s}_0, \boldsymbol{s}_w)}{1 - C_\theta(\boldsymbol{s}_0, \boldsymbol{s}_w)} Z(\boldsymbol{s}_0, \boldsymbol{s}_g), \tag{4}$$

where $Z(\boldsymbol{s}_0, \boldsymbol{s}_g)$ is a normalizing constant.

See Appendix A.3 for a full derivation. Since we will eventually normalize these importance weights (over $\boldsymbol{s}_w$), we do not need to estimate $Z(\boldsymbol{s}_0, \boldsymbol{s}_g)$. We call our complete algorithm *C-Planning*, and summarize it in Alg. 1. Note that our algorithm only yields a single line change in C-Learning at training time, commanding the agent to the intermediate waypoint $\boldsymbol{s}_w$ instead of the final goal $\boldsymbol{s}_g$. The sampling procedure in Alg. 2 is scalable and easy to implement.

The core idea of our method is to sample waypoints enroute to the goal. Of course, at some point we must command the agent to directly reach the goal. We introduce a hyperparameter n_g to indicate

[2] The assumptions are that *(1)* the policy always reaches the commanded waypoint, and that *(2)* that the goal-reaching probabilities $p_{\text{GEOM}}^{\pi(\cdot \mid \cdot, \cdot)}$ reflect the probability that the *optimal* policy reach some state.

Algorithm 1 C-Planning performs planning in data collection, modifies C-learning by $L5 \to L6-7$. The update for the policy and classifier ($L9$) is the same.

1: $\mathcal{D} \leftarrow \emptyset, N_g, \epsilon_d$
2: **for** $0 \le i \le N$ **do**
3: Set $n_g \leftarrow 0$ if new episode
4: $\boldsymbol{s}_0 \sim p_0(\boldsymbol{s}_0), \boldsymbol{s}_g \sim p_g(\boldsymbol{s}_g)$
5: $\cancel{\tau \sim p^\pi(\tau \mid \boldsymbol{s}_0, \boldsymbol{s}_g)}$
6: $\boldsymbol{s}_w, n_g \leftarrow \text{CPLANNING}(n_g, C)$
7: $\tau \leftarrow (\tau_1 \sim p^\pi(\tau \mid \boldsymbol{s}_0, \boldsymbol{s}_w),$ $\tau_2 \sim p^\pi(\tau \mid \boldsymbol{s}_0, \boldsymbol{s}_w))$
8: $\mathcal{D} \leftarrow \mathcal{D} \cup \{\tau\}$
9: $\pi, C \leftarrow \text{CLEARNING}(\mathcal{D}, \pi, C)$

Algorithm 2 C-Planning samples the intermediate waypoints, then command the agent to reach them.

1: n_g: number of waypoints agent has reached in an episode
2: **function** CPLANNING(n_g, C)
3: **if** ($t = 0$ or $d(\boldsymbol{s}_t, \boldsymbol{s}_w) \le \epsilon_d$) and $n_g \le N_g$ **then**
4: $n_g \leftarrow n_g + 1$
5: Sample M waypoints $\boldsymbol{s}_w^{(i)} \sim \text{Buffer}(\boldsymbol{s}_w)$
6: Compute distances for each candidate waypoint: $d^{(i)} \leftarrow \log \frac{C(F=1|\boldsymbol{s}_w,\boldsymbol{s}_g)}{C(F=0|\boldsymbol{s}_w,\boldsymbol{s}_g)} + \log \frac{C(F=1|\boldsymbol{s}_0,\boldsymbol{s}_w)}{C(F=0|\boldsymbol{s}_0,\boldsymbol{s}_w)})$
7: $\boldsymbol{s}_w \propto \text{SOFTMAX}(d^{(i)})$
8: **else**
9: Set $\boldsymbol{s}_w \leftarrow \boldsymbol{s}_g$
10: **return** $\boldsymbol{s}_w, n_g$

how many times we resample the intermediate waypoint, before commanding the agent to reach the goal. Note that this parameter is not the planning horizon, which is always set to 2

5 EXPERIMENTS

Our experiments not only study whether C-Planning can compete with prior goal-conditioned RL methods on benchmark tasks but also investigate tasks that are purposefully designed to pose a significant planning and exploration challenge. The benefits of C-Planning, compared with prior methods, are especially pronounced on these challenging tasks. To provide more intuition for why C-Planning outperforms prior goal-conditioned RL methods, we plot the gradient norm for the actor and critic functions, finding that C-Planning enjoys larger norm gradients for the critic and smaller variance of the gradient for the actor. Ablation experiments study the number of waypoints used in training. We finally measure the inference time of choosing actions, finding that C-Planning is up to $4.7\times$ faster than prior methods that perform search at test time.[3]

Environments. We investigate environments of various difficulty, some of which we visualize in in Fig. 3. The first set of environments is taken directly from the Metaworld benchmark suite (Yu et al., 2020), a commonly used benchmark for goal-conditioned RL. These tasks are challenging because of the manipulation skills required to push and reorient objects. While prior goal-conditioned RL methods do solve some of these tasks (e.g., `Push` and `Reach`), they stand at the edge of the capabilities of current methods. The second set of environments, 2D navigation mazes, are designed to stress-test the planning capabilities of different methods. Successfully solving these tasks requires episodes lasting up to 50 steps. These are challenging because, unlike many benchmarks, they require non-greedy exploration, with greedy strategies getting stuck in local optima. Our final set of environments combine the challenges of these environments: we extend the tasks in Metaworld to involve sequential manipulation of multiple objects. The tasks that we design (e.g., `Push-Two` and `Obstacle-Drawer-Close`), are inspired by prior work (Singh et al., 2020). These tasks require the robotic arm to manipulate multiple objects in sequence (e.g., first pushing an object, then opening a drawer). These tasks are quite difficult, as the agent is given no demonstrations, no reward shaping, and no manually specified distance metrics.

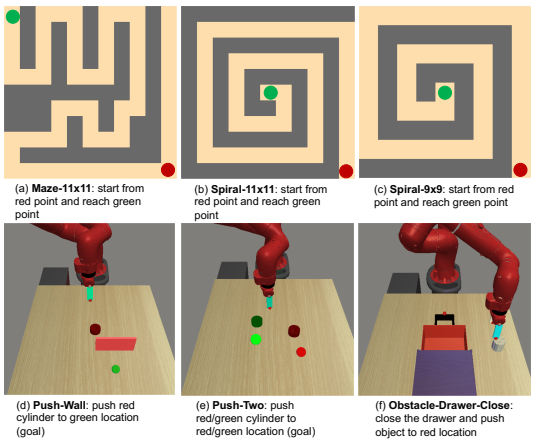
(a) **Maze-11x11**: start from red point and reach green point
(b) **Spiral-11x11**: start from red point and reach green point
(c) **Spiral-9x9**: start from red point and reach green point
(d) **Push-Wall**: push red cylinder to green location (goal)
(e) **Push-Two**: push red/green cylinder to red/green location (goal)
(f) **Obstacle-Drawer-Close**: close the drawer and push object to red location

Figure 3: **Environments**: Visualization of the 2D navigation maze environments (top row) and robotics manipulation tasks (bottom row).

[3]Code and videos of our results: `https://sites.google.com/view/cplanning`

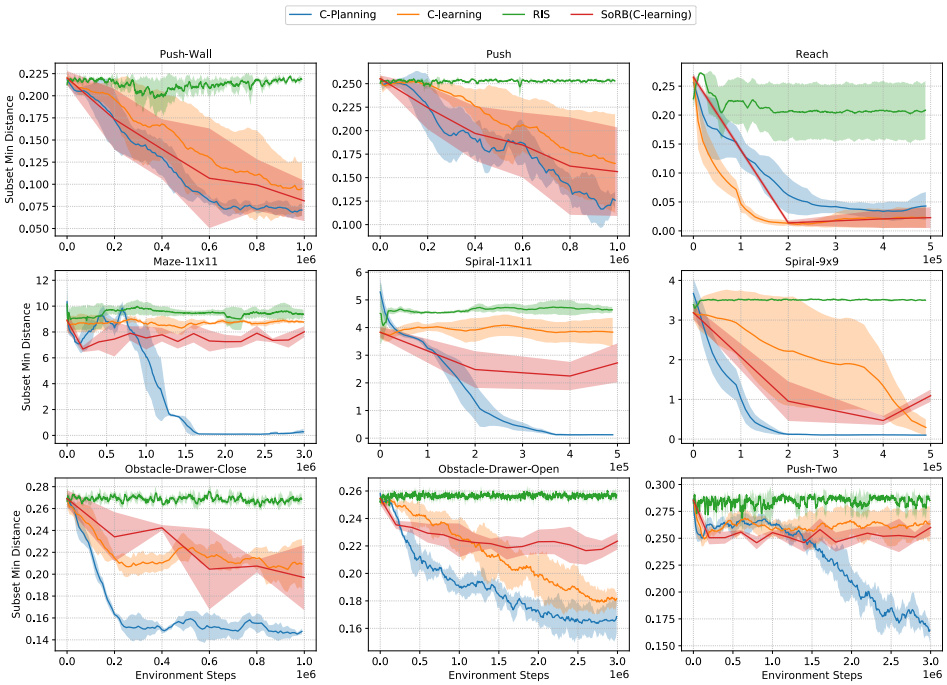

Figure 4: **Comparison of goal-conditioned RL methods**: We compare C-Planning to prior goal-conditioned RL algorithms on various tasks: *(top)* 2D maze navigation, *(middle)* benchmark manipulation tasks from Meta-world, and *(bottom)* robot manipulation tasks that require long horizon planning. For each task, we record the Euclidean distance to the goal, taking the minimum distance within an episode. Our method shows great sample efficiency on easy tasks like `Push` and `Obstacle-Drawer-Open`. It also solves the long-horizon tasks prior methods all fail (`Obstacle-Drawer-Close` and `Push-Two`).

Baselines. We will compare C-Planning to a number of baselines. C-Learning is a recent goal-conditioned RL method that does not perform planning. Because C-Learning performs the same gradient updates as C-Planning, and only differs in how experience is collected, this baseline will allow us to identify the marginal contribution of our curriculum of waypoints. The second baseline is SoRB (Eysenbach et al., 2019), a goal-conditioned RL method that performs search at test-time, rather than training time. While SoRB was originally implemented using Q-learning, we find that this version based on C-learning worked substantially better, so we use this stronger baseline in our experiments. Comparing against SoRB will allow us to study the tradeoffs between performing search during training versus testing. Because SoRB performs search at testing, it is considerably more expensive to deploy in terms of computing. Thus, even matching the performance of SoRB, without incurring the computational costs of deployment would be a useful result. The third baseline is an ablation of our method designed to resemble RIS (Chane-Sane et al., 2021). Like C-Planning, RIS performs search during training and not testing, but the search is used differently from C-Planning. Whereas C-Planning uses search to collect data, leaving the gradient updates unchanged, RIS modifies the RL objective to include an additional term that resembles behavior cloning. Our comparison with RIS thus allows us to study how our method for using the sampled waypoints compares to alternative methods to learn from those same sampled waypoints.

Results. We start with three tasks from the MetaWorld suite: `Reach`, `Push`, `Push-Wall`. On these tasks, we observe that in the easiest one `Push`, C-Planning does not perform as well as C-Learning; However, on tasks that require reasoning over longer horizons to solve, such as `Push-Wall`, C-Planning outperforms prior methods. This observation suggests that that the way-point sampling performed by C-Planning might be especially beneficial for solving tasks that require planning over long horizons.

To test the planning capabilities of C-Planning, we design a series of 2D navigation mazes. While the underlying locomotion motions are simple, these tasks post challenges for planning and are designed so that a greedy planning algorithm would fail. On all three mazes, C-Planning learns faster than all baselines and achieves a lower asymptotic distance to the goal, as compared to the baselines. On

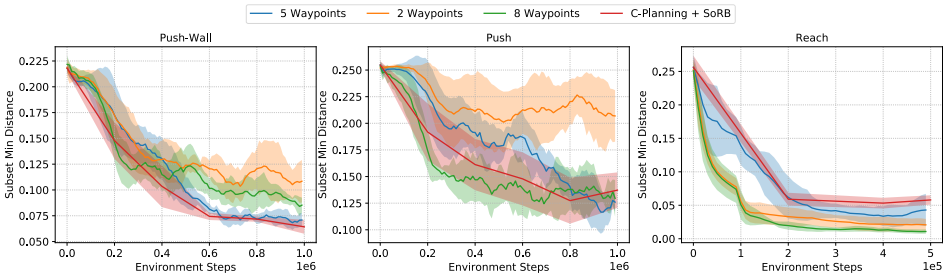

Figure 5: **Planning at Training Versus Testing:** We ablate the number of intermediate waypoints used to reach the goal, and compare against a variant of our method that performs planning during both training and testing ("C-Planning + SoRB"). This variant does not improve performance, indicating that the feedforward policy learned by C-Planning has already "internalized" these planning capabilities.

the most challenging maze, `Maze-11x11`, only our method is able to make any learning progress. Of particular note is the comparison with SoRB, which uses the same underlying goal-conditioned RL algorithm as C-Planning, but differs in that it performs search at testing, rather than training. We observe that SoRB does perform better than C-learning (which does not perform planning) on all three mazes but consistently performs worse than C-Planning. This observation suggests that C-Planning is not just amortizing the cost of performing search. Rather, these results suggest that incorporating search into training can produce significantly larger gains than incorporating search into the deployed policy.

As the last set of experiments, we want to see whether our observations generalize to higher dimensional tasks that require long-range planning. We design three environments from MetaWorld: `Obstacle-Drawer-Close`, `Obstacle-Drawer-Open` and `Push-Two`. These tasks have a fairly large dimension and require sequential manipulation of multiple objects. C-Planning outperforms all baselines on all three tasks. While C-learning makes some learning progress, SoRB (which modifies C-learning with search at test time) does not improve the results. RIS, which also samples waypoints during training but uses those waypoints differently, does not make any learning progress on these tasks. Taken together, these results suggest that waypoint sampling (as performed by C-Planning) can significantly aid in the solving of complex manipulation tasks, but only if that waypoint sampling is performed during training and those waypoints are used to collect new data, rather than augment the policy learning objective. To the best of our knowledge, C-Planning is the first method to learn manipulation behavior of this complexity without additional assumptions (such as dense rewards or demonstrations).

Ablations on Number of Waypoints and Comparison to Test-Time Planning Our ablations aim to answer the following two questions: (1) As we have aforementioned, C-Planning only applies planning at training time. Does planning in test time (SoRB with C-Planning) also help the performance? This helps us distinguish between our method and other test-time-planning methods. (2) What is the optimal number of intermediate waypoints for our algorithm? This justifies the design choice of the experiments and studies on whether using a larger number of waypoints in planning helps the performance in long-range tasks.

Fig 5 shows the results of ablation experiments. C-Planning performs the same at test-time with and without SoRB-based planning, suggesting that our training-time planning procedure already makes the policy sufficiently capable of reaching distant goals, such that it does not benefit from additional test-time planning. In addition to that, from Fig. 5, we can conclude that our choice on the prior distribution $p(\Delta)$ (number of waypoints) does exist a sweet spot for each task (2 or 5 in these tasks). However, C-Planning exhibits some amount of robustness to this design choice. As we vary the number of waypoints (e.g., 2, 5, and 8 in the figure), C-Planning still manages to solve all the tasks.

Gradient Analysis and Test Time Computation Cost. We provide an empirical observation why C-Planning is better than vanilla goal-conditioned RL methods. We hypothesize that, by introducing waypoints, C-Planning provides a better learning signal, increasing the gradient norm for training the critic and decreasing the variance of the gradient for the policy. While prior work has theoretically analyzed the importance of gradient norms for RL (Agarwal et al., 2021), we will provide some empirical evidence for the importance of controlling gradient norms. In Fig. 7, we plot the norm of the gradient (mean and variance) of both C-Planning and C-Learning using the Maze-11x11 envi-

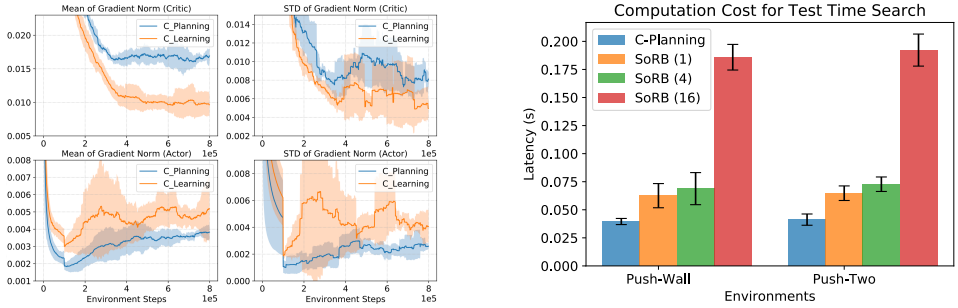

Figure 7: **Gradient Analysis and Computation Cost:** *(Left)* Mean of the gradient norm of the critic network. C-Planning has a larger gradient norm than C-Learning. Standard Deviation of the gradient norm of the actor network. C-Planning shows a smaller variance comparing to C-Learning. *(Right)* Computation cost in latency of C-Planning and SoRB with various number of waypoints.

ronment, finding that our method increases the norm of the critic norm and decreases the variance of the actor gradient norm.

One important advantage of C-Planning is that it enables direct execution at test time, does not require any online planning. At test time, we directly set the agent's goal to the final goal state without commanding any intermediate states. This makes C-Planning considerably different from previous planning methods, such as SoRB (Eysenbach et al., 2019). Not only does our method achieve a higher return, but it does so with significantly less computing at deployment. We measure the execution time at evaluation time for C-Planning and SoRB with a various number of waypoints in Fig. 7. As we see, C-Planning is 1.74× faster than planning over four waypoints and 4.71× faster than planning over 16 waypoints in `Push-Wall` environment.

Visualization of Training Dynamics. To further gain intuition into the mechanics of our method, we visualize how the distribution over waypoints changes during training of the 2D navigation of the four rooms environment. Fig. 6 shows the sampled waypoints. The value functions (i.e., future state classifiers) are randomly initialized at the start of training, so the waypoints sampled are roughly uniform over the state space. As training progresses, the distribution over waypoints converges to the states that an optimal policy would visit enroute to the goal. While we have only shown one goal here, our method trains the policy for reaching all goals. This set of experiments provides intuition of how C-Planning works as the distribution of waypoints shrinks from a uniform distribution to the single path connecting start state and goal state. This also provides experimental support for Eq. 2 as the distributions of waypoints qualitatively match well with the optimal solution.

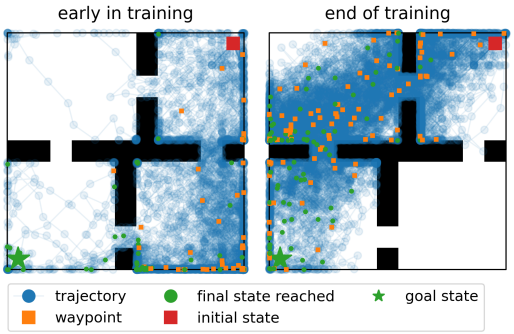

Figure 6: **Learning Dynamics:** *(Left)* Among those states that the agent is able to successfully reach early in training, the states closest to the goal are in the corners of the two rooms on the right. *(Right)* At convergence, waypoints are evenly distributed along states visited by the optimal policy, as predicted by our theory.

6 Conclusion

In this paper, we introduced C-Planning, a method for incorporating search into the training of goal-conditioned agents. The method enables the automatic generation of a curriculum over intermediate waypoints. Unlike prior methods, our approach avoids the computational complexity of performing search at test time. While being simple to implement on top of existing goal-conditioned RL algorithms, our resulting method achieves state-of-the-art results on a range of navigation and manipulation tasks. We want to emphasize that goal-sampling from extremely large dimensions is still challenging and exploration in goal-conditioned RL remains a core problem. Algorithmically, our method can be viewed as a combination of exact dynamic programming (search) and approximate dynamic programming (C-learning). We encourage future work to study the scaling limits of this combination on increasingly complex and long-horizon tasks.

Ethics Statement. Our paper focus on one of the core questions in RL, i.e. can we design an search/planning algorithm to help goal-conditioned RL agent learn complex long-range behavior. As a general algorithm for goal-conditioned RL, currently we cannot foresee any positive/negative social impact of our work. As we conduct our experiments on standard environments like 2D navigation maze and robotic manipulation benchmarks (MetaWorld), our experiments should not suffer from any discrimination/bias/fairness concerns.

Reproducibility Statement. Our code is directly adapted from the open-sourced code base from (Eysenbach et al., 2020) and we have introduced our implementation details in Appendix E.2 and Appendix E. We believe our results can be easily reproduced, and our code is available at https://sites.google.com/view/cplanning.

REFERENCES

Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan. On the theory of policy gradient methods: Optimality, approximation, and distribution shift. *Journal of Machine Learning Research*, 22(98):1–76, 2021.

Hagai Attias. Planning by probabilistic inference. In *International Workshop on Artificial Intelligence and Statistics*, pp. 9–16. PMLR, 2003.

Elliot Chane-Sane, Cordelia Schmid, and Ivan Laptev. Goal-conditioned reinforcement learning with imagined subgoals. In *International Conference on Machine Learning*, pp. 1430–1440. PMLR, 2021.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

Yiming Ding, Carlos Florensa, Pieter Abbeel, and Mariano Phielipp. Goal-conditioned imitation learning. *Advances in Neural Information Processing Systems*, 32:15324–15335, 2019.

Benjamin Eysenbach, Shixiang Gu, Julian Ibarz, and Sergey Levine. Leave no trace: Learning to reset for safe and autonomous reinforcement learning. In *International Conference on Learning Representations*, 2018.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. C-learning: Learning to achieve goals via recursive classification. In *International Conference on Learning Representations*, 2020.

William Fedus, Carles Gelada, Yoshua Bengio, Marc G Bellemare, and Hugo Larochelle. Hyperbolic discounting and learning over multiple horizons. *arXiv preprint arXiv:1902.06865*, 2019.

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*, pp. 482–495. PMLR, 2017.

Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *International conference on machine learning*, pp. 1515–1528. PMLR, 2018.

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. In *Conference on Robot Learning*, pp. 1025–1037. PMLR, 2020.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29:4565–4573, 2016.

David Hsu, Jean-Claude Latombe, and Hanna Kurniawati. On the probabilistic foundations of probabilistic roadmap planning. *The International Journal of Robotics Research*, 25(7):627–643, 2006.

Zhiao Huang, Fangchen Liu, and Hao Su. Mapping state space using landmarks for universal goal reaching. *arXiv preprint arXiv:1908.05451*, 2019.

Leslie Pack Kaelbling. Learning to achieve goals. In *IJCAI*, pp. 1094–1099. Citeseer, 1993.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *In Proc. 19th International Conference on Machine Learning*. Citeseer, 2002.

Aviral Kumar, Abhishek Gupta, and Sergey Levine. Discor: Corrective feedback in reinforcement learning via distribution correction. *Advances in Neural Information Processing Systems*, 33, 2020.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Xingyu Lin, Harjatin Singh Baweja, and David Held. Reinforcement learning without ground-truth state. *arXiv preprint arXiv:1905.07866*, 2019.

Aleksandr Mikhailovich Lyapunov. The general problem of the stability of motion. *International journal of control*, 55(3):531–534, 1992.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Conference on Robot Learning*, pp. 1113–1132. PMLR, 2020.

Xiangyun Meng, Nathan Ratliff, Yu Xiang, and Dieter Fox. Scaling local control to large-scale topological navigation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 672–678. IEEE, 2020.

Ofir Nachum, Yinlam Chow, Bo Dai, and Lihong Li. Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections. *Advances in Neural Information Processing Systems*, 32:2318–2328, 2019.

Soroush Nasiriany, Vitchyr Pong, Steven Lin, and Sergey Levine. Planning with goal-conditioned policies. *Advances in Neural Information Processing Systems*, 32:14843–14854, 2019.

Silviu Pitis, Harris Chan, Stephen Zhao, Bradly Stadie, and Jimmy Ba. Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. In *International Conference on Machine Learning*, pp. 7750–7761. PMLR, 2020.

Vitchyr Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. In *International Conference on Machine Learning*, pp. 7783–7792. PMLR, 2020.

Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. In *Twenty-third international joint conference on artificial intelligence*, 2013.

Zhizhou Ren, Kefan Dong, Yuan Zhou, Qiang Liu, and Jian Peng. Exploration via hindsight goal generation. *Advances in Neural Information Processing Systems*, 32:13485–13496, 2019.

Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. In *International Conference on Learning Representations*, 2018.

Yannick Schroecker and Charles Isbell. Universal value density estimation for imitation learning and goal-conditioned reinforcement learning. *arXiv preprint arXiv:2002.06473*, 2020.

Dhruv Shah, Benjamin Eysenbach, Gregory Kahn, Nicholas Rhinehart, and Sergey Levine. Ving: Learning open-world navigation with visual goals. *arXiv preprint arXiv:2012.09812*, 2020.

Avi Singh, Albert Yu, Jonathan Yang, Jesse Zhang, Aviral Kumar, and Sergey Levine. Cog: Connecting new skills to past experience with offline reinforcement learning. *arXiv preprint arXiv:2010.14500*, 2020.

Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. A generalized path integral control approach to reinforcement learning. *The Journal of Machine Learning Research*, 11:3137–3181, 2010.

Emanuel Todorov. Linearly-solvable markov decision problems. In *Advances in neural information processing systems*, pp. 1369–1376, 2007.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pp. 1094–1100. PMLR, 2020.

Tianjun Zhang, Huazhe Xu, Xiaolong Wang, Yi Wu, Kurt Keutzer, Joseph E Gonzalez, and Yuandong Tian. Bebold: Exploration beyond the boundary of explored regions. *arXiv preprint arXiv:2012.08621*, 2020.

Tianjun Zhang, Paria Rashidinejad, Jiantao Jiao, Yuandong Tian, Joseph Gonzalez, and Stuart Russell. Made: Exploration via maximizing deviation from explored regions. *arXiv preprint arXiv:2106.10268*, 2021.

Brian D Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010.