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 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-Planningnaturally follows expectation maximization (EM): the E step corresponds to planning an optimal sequence of waypoints using graph search, while the M step corresponds to learning 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 describes the problem of learning to reach goals. It is of particular import to the design of autonomous agents. 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.

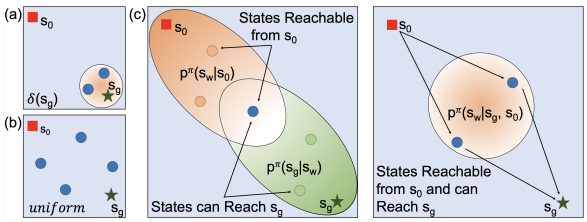

Figure 1: **Curriculum via Waypoint Sampling**: Whereas *(a)* standard goal-conditioned RL commands the policy to reach the desired goal, *(b)* sample a random goal in state space, *(c)* our method will first command an intermediate waypoint before commanding the desired goal. Our waypoint distribution maximizes a lower bound on goal-reaching, selects waypoints the policy can navigate from the initial state. and can also reach the desired goal state.

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 (see Fig. 1). Collecting data in this manner improves the quality of data, allowing the agent to learn to reach distant goals. Importantly, our method does not perform planning at test time, decreasing the computational demands for deploying the learned agent.

Importantly, our curriculum does not require manual engineering, but rather emerges automatically from a reformulation of goal-conditioned RL as variational inference. Our resulting method performs expectation maximization (EM) on a variational lower bound. The M step corresponds to a prior method for goal-conditioned RL, while the E step corresponds to graph search. Thus, EM presents an elegant marriage between previous goal-conditioned RL algorithms and graph-search methods. This method provably optimizes a lower bound on a well-established goal-reaching objective.

The work's main contribution is an algorithm, C-Planning, for goal-conditioned RL that excels at reaching distant goals. The main idea behind our method is to use a curriculum of subgoals during training, and our curriculum automatically 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, our method not only matches but surpasses the performance of these prior search-based methods, suggesting that our method is not just amortizing the cost of graph search. We empirically evaluate C-Planningon temporally-extended 2D navigation tasks and complex robotic manipulation tasks. C-Planningimproves the sample efficiency of prior goal-conditioned RL algorithms and manages to solve more difficult manipulations tasks such as rearranging multiple objects in sequence.

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, as 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. This tradeoff is desirable in many real-world settings where training can be done offline but the agent must make decisions in real-time at deployment. In this respect, our method is related to RIS (Chane-Sane et al., 2021), which also performs search at

[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.

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 biasing the learned policy and unifies the objectives for RL and graph search. We demonstrate the importance of this difference in our experiments.

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; Pitis et al., 2020). 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 $p(\Delta) = \textsc{NegBinom}(p = 1-\gamma, n)$ 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, $p^{\pi(\cdot|\cdot,\boldsymbol{s}_g)}_{\textsc{Geom}(1-\gamma)}$ 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:

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

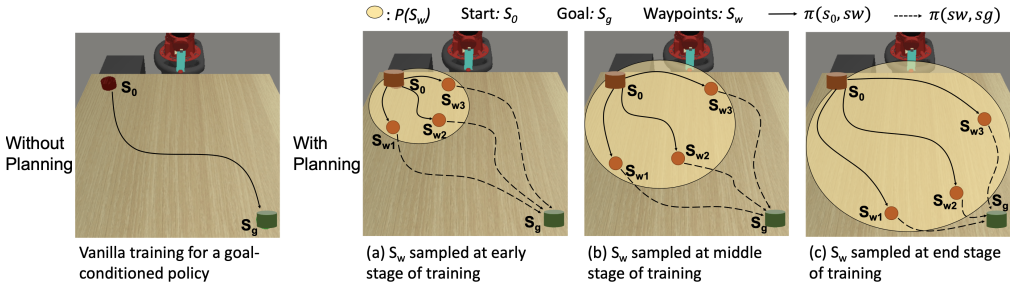

Figure 2: **C-Planning Curriculum**: Illustration of planning over waypoints distribution $p(\boldsymbol{s}_w)$. Goal-conditioned RL directly commands the agent to the final goal. C-planning first samples an intermediate waypoint $\boldsymbol{s}_w$ from $p(\boldsymbol{s}_w)$, directs the agent to that 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.

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 present our method C-Planning for learning to reach distant goals. 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 signal for the easier goal-reaching tasks. The sequence of easier training tasks acts as a curriculum. Such an idea of the decomposition is shown in Fig. 2. This section derives our method using the tools of variational inference and results in a curriculum learning method that resembles graph search.

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 particular states that the agent will visit are unknown latent variables. This problem now 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. 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:*

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

See Appendix A.1 for a full derivation. We illustrate this waypoint distribution in Fig. 1. 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.

To summarize, the only difference between our method and prior goal-conditioned RL methods is how data is collected, not in the updates performed using that data. While this difference is small and easy to implement (see next section), our experiments show that it leads to substantially improved performance across a range of challenging control tasks.

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 $d(\boldsymbol{s}_1, \boldsymbol{s}_2) = \log p_{\text{GEOM}}^{\pi(\cdot|\cdot,\boldsymbol{s}_2)}(\boldsymbol{s}_{t+} = \boldsymbol{s}_2 \mid \boldsymbol{s}_1)$. 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. 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.

4.2 A Practical Implementation

We now describe how we implement the policy updates (M-step) and the 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 state. We implement waypoint sampling using importance sampling. Let $b(\cdot)$ be the background distribution, which we take to be the replay buffer in our experiments. We first sample a batch of waypoints from the buffer, and then sample one waypoint from within that batch using the normalized importance weights, $\frac{q(\boldsymbol{s}_w|\boldsymbol{s}_g,\boldsymbol{s}_0)}{b(\boldsymbol{s}_w)}$. We estimate these importance weights using the classifier learned by C-learning (Eq. 3):

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.

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

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

1: $\mathcal{D} \leftarrow \emptyset, N_g, \epsilon_d$
2: **for** $0 \leq i \leq N$ **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: $\boldsymbol{s}_w, n_g \leftarrow$ 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: $\pi, C \leftarrow$ CLEARNING$(\mathcal{D}, \pi, C)$

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

1: n_g: number of waypoints reached in episode
2: **function** CPLANNING(n_g, C)
3: Set $\boldsymbol{s}_w \leftarrow \boldsymbol{s}_g$
4: **if** ($t = 0$ or $d(\boldsymbol{s}_t, \boldsymbol{s}_w) \leq \epsilon_d$) and $n_g \leq N_g$ **then**
5: $n_g \leftarrow n_g + 1$
6: Sample M points $\boldsymbol{s}_w \sim p(\boldsymbol{s}_w)$
7: $\boldsymbol{s}_w \propto \text{SoftMax}(\log \frac{C(F=1|\boldsymbol{s}_w,\boldsymbol{s}_g)}{C(F=0|\boldsymbol{s}_w,\boldsymbol{s}_g)} + \log \frac{C(F=1|\boldsymbol{s}_0,\boldsymbol{s}_w)}{C(F=0|\boldsymbol{s}_0,\boldsymbol{s}_w)})$
8: **return** $\boldsymbol{s}_w, n_g$

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. 2.

5 EXPERIMENTS

We evaluate C-Planning extensively on 2D navigation tasks and challenging robotic manipulation tasks. We first conduct an oracle experiment, confirming a simple result: suppose we can sample from an optimal policy's state density distribution (e.g., fixed classifier of the optimal policy), then performing goal-conditioned RL can result in faster learning than vanilla goal-conditioned RL. We then show results of our algorithm with learning the classifier and policy jointly, followed by an ablation study on the number of waypoints we choose for the prior negative binomial distribution. To provide some more intuition on why C-Planning works much better: we also plot the norm of the gradient for actor and critic functions, showing that C-Planning enjoys larger norm of gradient for critic function and smaller variance of the gradient of the actor function. We finally show a the test-time computation cost comparison between C-Planning and SoRB (Eysenbach et al., 2019). This reveals that our method enjoys a much better computation efficiency and thus easy to employ.

We visualize some of the environments in Fig. 3. The 2D navigation tasks are based on a simple continuous maze environment, and the robotic manipulation tasks are based on MetaWorld (Yu et al., 2020) (a commonly used benchmark for goal-conditioned RL). Some tasks, such as `Obstacle-Drawer-Close` (close drawer and push block) and `Push-Two` (push two blocks) are considerably more complex than tasks used in prior work on goal-conditioned RL. This is because these tasks require the robotic arm to perform manipulation (e.g., push an object, open a drawer) to more than one object in a sequential manner. Prior methods that look at similar tasks(COG (Singh et al., 2020), Actionable Models (Chebotar et al., 2021) typically assume access to expert demonstrations or shaped rewards. Our setting is considerably more challenging as the agent is given no demonstrations, no reward shaping, and no manually-specified distance metrics.

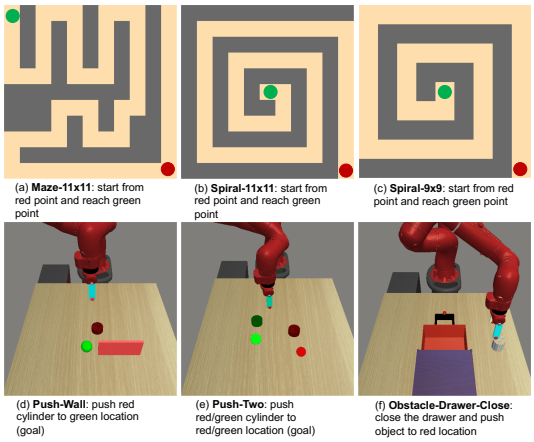
(a) **Maze-11x11**: start from red point and reach green point
(b) **Spiral-11x11**: start from red point and reach green point
(c) **Spiral-9x9**: start from red point and reach green point
(d) **Push-Wall**: push red cylinder to green location (goal)
(e) **Push-Two**: push red/green cylinder to red/green 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).

5.1 PLANNING VIA OPTIMAL GENERATIVE MODEL

Our first experiment confirms the well-known result (Kakade & Langford, 2002) 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 behavior 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. 2, 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. 4 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 for more details.

In addition to that, we are also interested in measuring how well does planning through a learned model (C-Planning) perform if we don't have access to the optimal generative model. Results in Fig. 4 also shows that C-Planning achieves performance comparable to planning via optimal generative model in the Push and the Push-Wall environment.

5.2 Planning via Learned Classifier

Our second set of experiments compare C-Planning to prior goal-conditioned RL methods on simple navigation tasks and on more complex robot manipulation tasks. We compare C-Planning with the following baselines: (1) The most natural comparison is to use the exact same goal-conditioned RL algorithm (C-Learning (Eysenbach et al., 2020)) but to not perform search during exploration (red lines in Alg. 2). (2) Another baseline, which is an extension of the previous one, is planning at test time (SoRB (Eysenbach et al., 2019) combined with C-Learning). This ablation corresponds to standard goal-conditioned RL (without planning) at training time, but incorporates waypoint sampling at test-time. (3) We also compare against an ablation of our method that is designed to resemble RIS (Chane-Sane et al., 2021). This ablation samples waypoints using the same method as our method, but uses those waypoints differently. Precisely, the policy for reaching goal s_g is regularized to produce similar actions to the policy for reaching waypoint s_w. An essential difference between our method and this ablation is that this ablation modifies the RL objective. Because our method only modifies the data and not the objective, we expect our method to avoid bias and result in better policies. (4) In addition, we compare against Hindsight Experience Replay (Andrychowicz et al., 2017), a well-known goal-conditioned RL algorithm that leverages the goal-relabeling of the previous data.

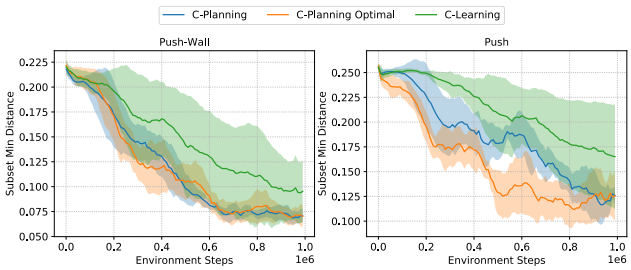

Figure 4: **Oracle experiment with the optimal generative model**: Goal-conditioned RL learns the task much faster if we can sample the waypoints from the expert policy's marginal state distribution (C-Planning Optimal). Compared to planning with optimal state density distribution, planning using a learned classifier (C-Planning) shows comparable performance.

We also emphasize that prior works on nonparametric planning (Savinov et al., 2018; Eysenbach et al., 2019; Shah et al., 2020; Meng et al., 2020) focus solely on navigation, where the planning problem is 2D even when observations are high-dimensions. We are unaware of similar approaches being applied to tasks like manipulation where the dimension of the system is much larger than two.

We show results in Fig. 5 and all the experiments are averaged across four random seeds. We evaluate C-Planning on a range of environments. The first set of environments are taken directly from the Metaworld benchmark suite (Yu et al., 2020). 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 are 2D navigation mazes, which are designed to stress-test the planning capabilities of different methods. Successfully solving these tasks requires episodes lasting up to 200 steps. These tasks are especially challenging because, unlike many benchmark tasks, they require non-greedy exploration, with greedy strategies getting study in local optima. Our final set of environments combine the challenges of these other environments: we extend the benchmark tasks in Metaworld to involve sequential manipulation of multiple objects (e.g., `Push-Two` and `Obstacle-Drawer-Close`). These tasks require both planning over long hori-

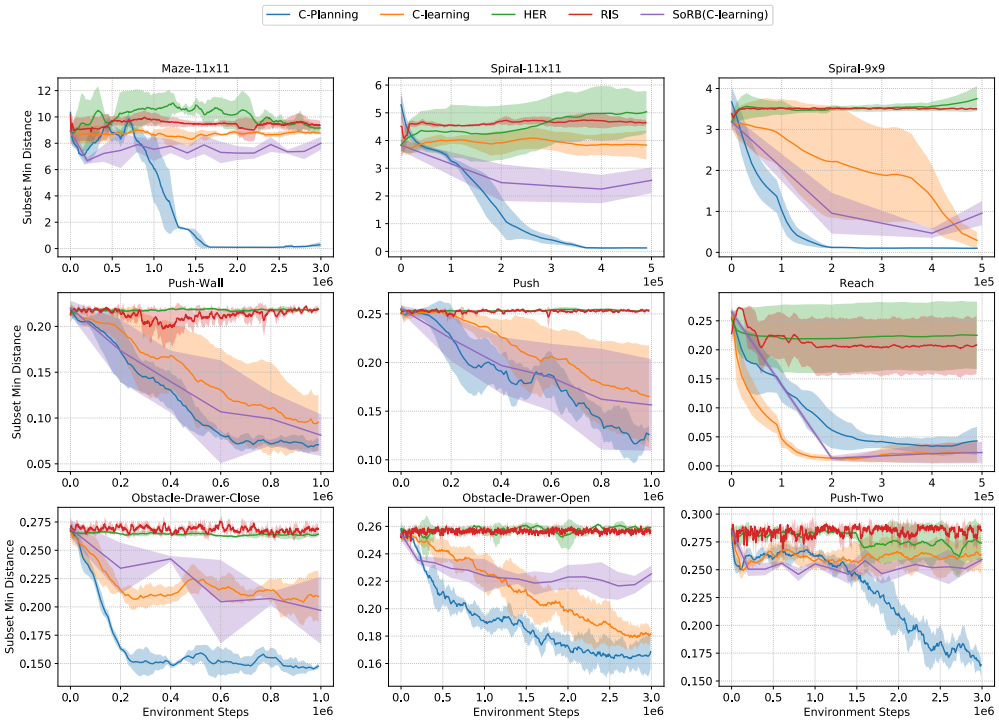

Figure 5: **Comparison of goal-conditioned RL methods**: We compare C-Planning to prior goal-conditioned RL algorithms on various tasks: *(top)* 2D maze navigation, *(middle)* benchmark manipulation tasks from Meta-world, and *(bottom)* robot manipulation tasks that require long horizon planning. For each task, we record the Euclidean distance to the goal, taking the minimum distance within an episode.

zons and learning complex manipulation skills. These tasks are inspired by prior work (COG (Singh et al., 2020), Actionable Models (Chebotar et al., 2021)), but prior methods that address tasks of this difficulty typically rely on additional supervision, such as human demonstrations or offline dataset. To the best of our knowledge, tasks of comparable difficulty have not been solved by prior RL methods without additional supervision or reward shaping.

5.3 Ablation Experiments.

Our ablations try 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) Can other standard goal-conditioned RL methods, such as Hindsight Experience Replay (HER), be combined with the learned planner? This shows whether the base algorithm C-Learning has some essential effects on the planning part. (3) What is the optimal number of intermediate waypoints for our algorithm? This not only justifies the design choice of the experiments, but also reflects the choice of n in the prior distribution $p(\Delta)$.

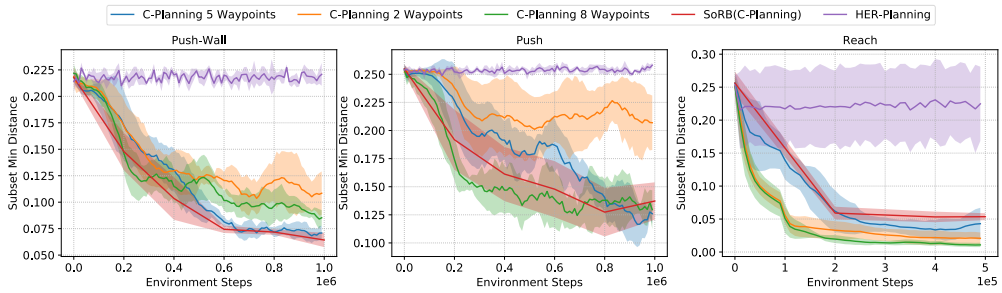

Figure 6: **Empirical analysis:** We compare our method with HER-Planning and SoRB(C-Planning). C-Planning are considerably better than HER-Planning, while the performance between C-Planning with and without test time planning are similar. In addition, there exists a sweat point for the number of waypoints.

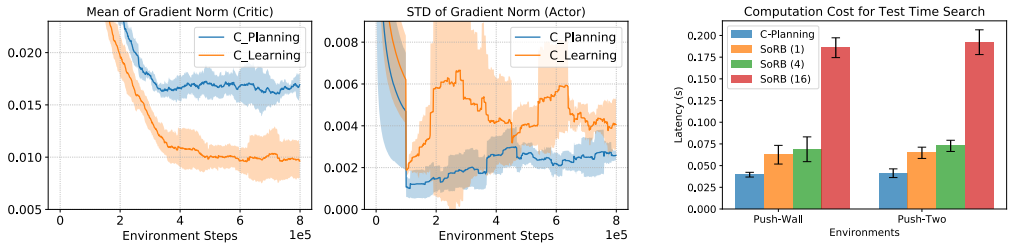

Figure 7: **Gradient Analysis and Computation Cost:** *(Left)* Mean of the gradient norm of the critic network. C-Planning has a larger gradient norm than C-Learning. Standard Deviation of the gradient norm of the actor network. C-Planning shows a smaller variance comparing to C-Learning. *(Right)* Computation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

Fig 6 shows the results of ablation results. We see that C-Planning with or without SoRB at test time show comparable performance. In addition to that, C-Planning significantly outperforms HER-Planning. For HER-Planning, we change the underlying goal-conditioned RL algorithm to HER but keep the same sampling procedure of waypoints as in Algorithm 2. This suggests that planning via goal-conditioned Q-value function doesn't work well, and C-Learning also plays an important role. At last, we see that our prior knowledge on the 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.

5.4 Gradient Analysis

In this section, 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 environment, finding that our method increases the norm of the critic norm and decreases the variance of the actor gradient norm.

5.5 Computation Cost

One important advantage of C-Planning is that it enables direct execution at test time, does not require any online planning. That is, at test time, we directly set the goal of the agent 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 higher return, but it does so with significantly less compute at deployment. We measure the execution time at evaluation time for C-Planning and SoRB with various number of waypoints in Fig. 7. As we see, C-Planning is $1.74\times$ faster than planning over 4 waypoints and $4.71\times$ faster than planning over 16 waypoints in `Push-Wall` environment.

6 Conclusion

In this paper, we introduced a method C-Planning 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. Algorithmically, our method can be viewed as a bias-free 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.

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.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *arXiv preprint arXiv:1707.01495*, 2017.

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.

Yevgen Chebotar, Karol Hausman, Yao Lu, Ted Xiao, Dmitry Kalashnikov, Jake Varley, Alex Irpan, Benjamin Eysenbach, Ryan Julian, Chelsea Finn, et al. Actionable models: Unsupervised offline reinforcement learning of robotic skills. *arXiv preprint arXiv:2104.07749*, 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.

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