Motion-Planning via Contrastive Reinforcement Learning and Gumbel Monte-Carlo Tree Search

Anonymous authors
Paper under double-blind review

Keywords: RLJ, RLC, formatting guide, style file, LaTeX template.

Summary

We consider the generalized movers' problem i.e. finding any path that moves an object to a desired goal while avoiding collisions. Even relaxing optimality requirements, the any-path problem is computationally challenging. Namely, exponential in the degrees of freedom of the object. Due to the *curse of dimensionality*, applying traditional search algorithms to the discretized state space becomes infeasible as the state space grows. This motivated the use of sampling-based methods. These sampling-based methods are *tabula rasa* and require complete re-learning on each problem instance. Existing learning-based methods that attempt to leverage shared structure aim to handle arbitrary changes in the environment. Often, this still requires a significant number of samples and / or expert demonstrations. In practice, many robotics applications or UAV routing do not need to handle these pathological cases, where the environment undergoes drastic change. Rather, they must only be able to avoid a sudden obstacle while their route remains largely unchanged. We allow pre-training in an obstacle free environment and show that combining contrastive reinforcement learning with classical game-inspired search algorithms enables zero shot performance to unseen obstacles.

Contribution(s)

1. Propose formulation for the motion-planning problem that allows pre-training in an obstacle free simulator.
   Context: Existing motion planning (Garrett et al., 2020) focuses on solving planning problems *tabula rasa*. Learning-based approaches typically require expert demonstrations or many sample problem configurations Tamar et al. (2016); Chen et al. (2019).
2. Propose initial combination of combining contrastive reinforcement learning Eysenbach et al. (2023) and gumbel monte-carlo tree search Danihelka et al. (2022) to show the potential for pre-training in an obstacle free environment.
   Context: Recent advancements in goal-conditioned reinforcement learning Eysenbach et al. (2023) enabled learning goal reaching policies for robotics systems with a high number of degrees of freedom. Combining with search, is a promising avenue for path planning with obstacles.

Motion-Planning via Contrastive Reinforcement Learning and Gumbel Monte-Carlo Tree Search

Anonymous authors
Paper under double-blind review

Abstract

We consider the generalized movers' problem i.e. finding any path that moves an object to a desired goal while avoiding collisions. Even relaxing optimality requirements, the any-path problem is computationally challenging. Namely, exponential in the degrees of freedom of the object. Due to the *curse of dimensionality*, applying traditional search algorithms to the discretized state space becomes infeasible as the state space grows. This motivated the use of sampling-based methods. These sampling-based methods are *tabula rasa* and require complete re-learning on each problem instance. Existing learning-based methods that attempt to leverage shared structure aim to handle arbitrary changes in the environment. Often, this still requires a significant number of samples and / or expert demonstrations. In practice, many robotics applications or UAV routing do not need to handle these pathological cases, where the environment undergoes drastic change. Rather, they must only be able to avoid a minor mismatch with the training environment while their route remains largely unchanged. We allow pre-training in an obstacle free environment and show that combining contrastive reinforcement learning with classical game-inspired search algorithms enables zero shot performance to unseen obstacles.

1 Introduction

The generalized movers' motion-planning problem (Lozano-Pérez & Wesley, 1979) aims to find any path that moves an object to a desired goal while avoiding collisions. The problem has a wide range of applications, including self-driving (Teng et al., 2023), robotics (LaValle, 2006; Kunchev et al., 2006; Orthey et al., 2023), and UAVs (Quan et al., 2020). Due to the curse of dimensionality, traditional path-planning such as A^* (Hart et al., 1968) becomes infeasible as any-path-motion-planning is exponential in the degrees of freedom (Kozen & Yap, 1985) and shortest-path is NP-hard Canny & Reif (1987). If the obstacles are moving, then even the any-path problem is NP-hard and PSPACE-hard (Reif, 1979). Therefore, for high-dimensional problems, sampling based planning algorithms (Orthey et al., 2023) have become the standard in practice. Notably, traditional sampling-based algorithms (Williams et al., 2016; Durrant-Whyte et al., 2012; Kavraki et al., 1996) are *tabula rasa*, meaning that they must find a suitable path from scratch for each new configuration. To remedy this, there has been substantial work on learning priors or policies to help guide the search (Zucker et al., 2008; Kim et al., 2017; Ichter et al., 2018; Huh & Lee, 2018). These methods rely on handcrafted features or expert demonstration and typically do not generalize to changes in obstacle configurations.

Instead we observe that (1) many real world applications have access to a "map" of the deployment environment without obstacles and (2) these applications primarily require the agent to be able to avoid sparse obstacles. For example, robots in a factory setting need to avoid occasional humans in their path, or UAVs need to avoid a fallen tree. For these applications, the first observation motivates us to leverage advancements in unsupervised reinforcement learning, particulary, contrastive reinforcement learning (Eysenbach et al., 2022). The second motivates us to avoid sample inefficient trajectory based techniques and instead utilize heuristic based search methods common in reinforcement learning for games (Silver et al., 2017; 2018; Schrittwieser et al., 2020; Danihelka et al., 2022). This also provides the added advantages of these heuristics, such as support for stochastic / changing dynamics and non-stationary obstacles over time.

1.1 Additional Related Work

Model Predictive Control: Similar to our approach, in MPC they consider a finite time horizon to avoid the exponential explosion in the time horizon. Our method can be seen as using the goal conditioned value function as a surrogate cost (Lowrey et al., 2019). However, in traditional MPC, the optimization either has a closed form solution, or is differentiable and can perform gradient-based optimization. We do not assume differentiable dynamics and therefore are most similar to data-driven MPC, where they instead optimize with sampling based methods, such as (Williams et al., 2016; Durrant-Whyte et al., 2012). However, these methods cannot fully leverage pre-trained policies, so we suspect that they will be less sample efficient as we increase the degrees of freedom of the system.

Goal-conditioned Planning: There has been a line of work concerned with using goal-conditioned reinforcement learning with a hierarchical planner (Nasiriany et al., 2019; Dubey et al., 2021; Chane-Sane et al., 2021), where in addition to a goal conditioned policy they learn a hierarchical planner to plan intermediate subgoals. These works still focus on the same environment and aim to solve the problem of horizon generalization (Park et al., 2025; Myers et al., 2025). Alternatively, in Eysenbach et al. (2019), they learn a goal-conditioned reinforcement learning to assign weights for use in Djikstra's algorithm (Dijkstra, 1959). However, this relies on low-dimensionality much like the other previously discussed methods.

Learning-based motion planning: One approach to leverage structure across planning problems is to learn a "work-space" conditioned policy (Tamar et al., 2016; Oh et al., 2017; Qureshi et al., 2019; Chen et al., 2019). Here, a work-space is a 2D birds-eye view of the environment. These methods can be seen as a *representation learning* methods, where they aim to jointly learn a representation for the workspace and a policy on this latent representation. The final policy is used to guide existing sampling-based motion-planning methods. These works require either expert demonstrations or be trained incrementally interpolating from pure sample-based methods.

Model-based safe RL: Safe reinforcement learning (Gu et al., 2024) is primarily concerned with finding an optimal policy to a constrained MDP (Altman, 2021). Analogous to constrained optimization, this often consists of a primal-dual (Bai et al., 2022; Paternain et al., 2022) or trust region optimization (Achiam et al., 2017). More similar to our setting is model-based safe RL, where they use a model to plan over the unsafe states (Efroni et al., 2020; Thomas et al.). However, in this setting, the model is still over the training environment, and the training and implementation environments are the same, so they still recover a policy that they use without search.

Unsupervised RL: Typically, unsupervised reinforcement learning consists of pre-training in a reward-free environment with the hope of accelerating fine-tuning on the downstream task. Early work aims to learn intrinsic rewards, which induce *generally useful behavior* (Pathak et al., 2017; Eysenbach et al., 2018; Pathak et al., 2019; Zhao et al., 2022). More related, recent work (Touati & Ollivier, 2021; Machado et al., 2023; Touati et al., 2023; Carvalho et al., 2024; Agarwal et al., 2024) is concerned with learning representations that linearly span all possible rewards. These methods usually are related to the successor representation (Dayan, 1993) and aim to provide more stable alternatives to successor features (Barreto et al., 2017; Borsa et al., 2018). Contrastive reinforcement learning (Eysenbach et al., 2022) can be seen as an unsupervised reinforcement learning algorithm. However, we immediately recover the goal-reaching policy without needing any additional learning on the downstream task.

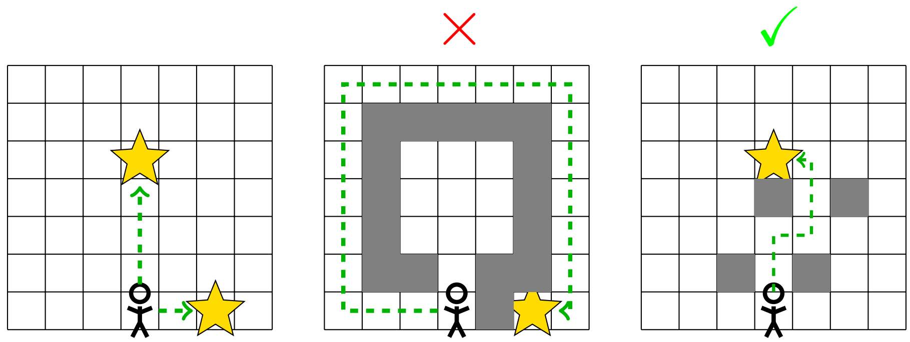

Figure 1: Train environment without obstacles (left) vs. eval environment with obstacles (right). Similar to MPC, we assume that the task is solvable by only observing a finite horizon. Therefore, we are more concerned with the second configuration, which more closely models real-world changes.

2 Problem

As previously discussed, as the state space grows (i.e. robots with a many degrees of freedom) tabula rasa motion-planning algorithms become computationally intractable. Due to this, we must allow for some inductive bias. In this work, we make the relaxation that the agent can undergo a pre-training phase, where they can learn in an environment without perfect knowledge of the downstream deployment configuration.

Formally, we define the MDP problem studied in the paper below.

Definition 2.1. *(MDP) We define an MDP as $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, p, p_0, r, \gamma \rangle$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ are the transition dynamics, where $p(s, a, s')$ represents the probability of transitioning to state s' given you are in state s and take action a, $p_0 : \mathcal{S} \rightarrow [0, 1]$ is the starting state distribution, $r : \mathcal{S} \rightarrow \mathbb{R}$ is reward function, and γ is the discount factor. We say the MDP is reward-free if it does not have a corresponding reward function.*

Definition 2.2. *(Obstacle-MDP) Given a set of obstacles $\mathcal{S}_{obs} \subset \mathcal{S}$ and an MDP $\mathcal{M}$. The corresponding Obstacle-MDP is the MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, p_{obs}, p_0, r, \gamma \rangle$, where p_{obs} are the same transition dynamics as p except all obstacles states $s \in S_{obs}$ are now absorbing.*

Definition 2.3. *(Goal-Parametrized Family of MDPs) Given a reward-free MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, p_0, \gamma \rangle$ the goal-parametrized family of $\mathcal{M}$ is given by*

$$\mathcal{G}(\mathcal{M}) = \{ \langle \mathcal{S}, \mathcal{A}, p, p_0, r_g, \gamma \rangle \mid g \in \mathcal{S} \},$$

where the MDPs of the family only differ from $\mathcal{M}$ in the reward function r_g given by $r_g(s) = \mathbf{1}_{\{s=g\}}$

Definition 2.4. *(Goal-Obstacle-Parametrized Family of MDPs) Given a reward-free MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, p_0, \gamma \rangle$ the goal-obstacle-parametrized family of $\mathcal{M}$ is given by*

$$\mathcal{GO}(\mathcal{M}) = \{ \langle \mathcal{S}, \mathcal{A}, p_{obs}, p_0, r_g, \gamma \rangle \mid g \in \mathcal{S}, \mathcal{S}_{obs} \subset \mathcal{S} \}.$$

Goal-Conditioned Motion Planning Problem (GCMP): Given a reward-free MDP $\mathcal{M}$, we want to find the policy π^*, such that

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{M \in \mathcal{GO}(\mathcal{M})}[V_M^{\pi}],$$

our first insight can be seen as taking the expectation over a particular distribution on $\mathcal{GO}(\mathcal{M})$. Note that we assume that the training and eval MDPs share the same dynamics outside of the states containing obstacles. This formalizes how we hope to *cache* goal-reaching policies for the downstream task. Intuitively, the goal-parametrized MDPs correspond to having a map of the environment,

Algorithm 1: `CRL`

Input: Critic parameters ψ, ϕ, policy parameters θ, reward-free MDP $\mathcal{M}$
$\langle \mathcal{S}, \mathcal{A}, p, p_0, \gamma \rangle \leftarrow \mathcal{M}$
while *not converged* **do**
  $g \sim p(g)$
  $s_0 \sim p_0$
  foreach *environment step* **do**
    $a_t \sim \pi_\theta(s_t, g)$
    $s_{t+1} \sim p(s_t, a_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup (a_t, s_t, r_t(s_t, a_t))$
  end
  foreach *gradient step* **do**
    $\psi \leftarrow \psi - \alpha \nabla_\psi \mathcal{L}_{\text{RL InfoNCE}}(f_{\psi,\phi}(\mathcal{D}))$
    $\phi \leftarrow \phi - \alpha \nabla_\phi \mathcal{L}_{\text{RL InfoNCE}}(f_{\psi,\phi}(\mathcal{D}))$
    $\theta \leftarrow \theta - \alpha \nabla_\theta L_\pi(\theta)$
  end
end

without obstacles, where the agent will live. The family additionally parametrized by obstacles represents the possible obstacle configurations the agent may encounter when deployed in the real world. This is seen in Figure 1, where the goal-parametrized MDP is given by the left gridworld environment, showing two possible goals without obstacles. On the right, are examples of possible obstacle configurations.

3 Methodology

3.1 Preliminaries

First, we must discuss the key methods underlying our approach.

Goal-Conditioned RL: In goal conditioned reinforcement learning (Liu et al., 2022), one aims to solve the multi-task reinforcement learning problem (Schaul et al., 2015; Borsa et al., 2016; Vithayathil Varghese & Mahmoud, 2020) where tasks refer to reaching states in the environment. As presented in Eysenbach et al. (2022), goal-conditioned RL can be seen as RL with the reward function

$$r_g(s_t, a_t) = (1 - \gamma) p(s_{t+1} = g \mid s_t, a_t).$$

Contrastive RL: In traditional contrastive learning (Gutmann & Hyvärinen, 2010; Ma & Collins, 2018; Oord et al., 2019), one tries to learn the underlying data distribution by learning to distinguish between positive (true) and negative (arbitrarily) generated samples. Explicitly, given distribution of $p_\mathcal{X}(x), p_\mathcal{Y}(y)$ over data $x \in \mathcal{X}$, $y \in \mathcal{Y}$ and the conditional distribution of positive pairs $p_{\mathcal{Y}|\mathcal{X}}(y \mid x)$ over $\mathcal{X} \times \mathcal{Y}$, the InfoNCE loss (Oord et al., 2019) is

$$\mathcal{L}_{\text{InfoNCE}}(f) = \mathbb{E}_{\substack{x \sim p_\mathcal{X}(x), y^+ \sim p_{\mathcal{Y}|\mathcal{X}}(y|x) \\ y_{1:N}^- \sim p_\mathcal{Y}(y)}} \left[\log \frac{e^{f(x,y^+)}}{\sum_{i=1}^N e^{f(x,y_i^-)}} \right]. \tag{1}$$

The key insight in Eysenbach et al. (2022) is that we can learn a goal-reaching policy by maximizing the probablity of reaching the goal under the discounted state occupancy measure (Puterman, 1994; Sutton et al., 1999), where the discounted state occupancy measure is given by

$$p^\pi(s^+ \mid s, a) = (1 - \gamma) \sum_{t=1}^{\infty} \gamma^{t-1} p_t^\pi(s^+ \mid s, a).$$

Algorithm 2: `Gumbel :: Selection`

Input: policy π_θ, value v_ϕ
Input: state s, transition kernel p, reward r
Output: A_{t+1}
Sample k Gumbel variables
Sample $\mathcal{A}_{topn}$ according to (4)
$\mathcal{A}_{next} = \mathcal{A}_{topn}$
while $m > 1$ **do**
  foreach $a \in \mathcal{A}_{next}$ **do**
    $visits = \lfloor \frac{n}{\log_2(m)m} \rfloor$
    $\mathcal{T} = \texttt{Simulate}(s, a, visits)$
  end
  $v \leftarrow \texttt{Backup}(\mathcal{T})$
  $m \leftarrow m/2$
  $\mathcal{A}_{next} \leftarrow$ top m of $\mathcal{A}_{topn}$
end
Select A_{n+1} according to (5)

Algorithm 3: `Gumbel :: Expansion`

Input: Tree $\mathcal{T} = (\mathcal{V}, \mathcal{E}, \mathcal{N})$, s, a s'
Output: Tree $\mathcal{T}$
$\mathcal{V} \leftarrow \mathcal{V} \cup s'$
$\mathcal{E} \leftarrow \mathcal{E} \cup (s, a, s')$

Algorithm 4: `Gumbel :: Simulate`

Input: Tree $\mathcal{T} = (\mathcal{V}, \mathcal{E}, \mathcal{N})$
Input: state s_0, action a_0, transition kernel p
Output: Tree $\mathcal{T}$
$s \leftarrow s_0$
$a \leftarrow a_0$
while *true* **do**
  $\mathcal{N}(s, a) \leftarrow \mathcal{N}(s, a) + 1$
  $s' \sim p(s, a)$
  if $s' \notin \mathcal{V}$ **then**
    $\texttt{Expansion}(\mathcal{T}, s, s')$
    return
  end
  $s \leftarrow s'$
  $a \sim \pi_D(s)$ from (6)
end

Algorithm 5: `Gumbel :: Backup`

Input: Tree $\mathcal{T} = (\mathcal{V}, \mathcal{E}, \mathcal{N})$
Output: Tree $\mathcal{T}$
foreach $(s_{par}, a, s_{child}) \in \mathcal{E}$ **do**
  $v(s_{par}) = \frac{\mathcal{N}(s_{par}, a)v(s_{par}) + v(s_{child})}{\mathcal{N}(s_{par}, a) + 1}$
end

Figure 2: Gumbel in the formulation presented for planning algorithms in Sutton et al. (1998).

Here, $p_t^\pi(s^+ \mid s, a)$ is the probability density of reaching state s^+ after exactly t steps, starting at state s, taking action a, and following policy $\pi(a \mid s)$. Using a goal-conditioned policy $\pi(a \mid s, g)$ we get an analogous goal-conditioned state occupancy $p^\pi(s^+ \mid s, a, g)$. We then want to optimize the following objective

$$\max_\pi \mathbb{E}_{p_g(g), p_0(s), \pi(a|s,g)}[p^\pi(s^+ = g \mid s, a, g)].$$

Explicitly, we want to maximize the probability of reaching the goal state g under the policy $\pi(\cdot \mid s, g)$. To do so, we can modify (1). We introduce $p_{\mathcal{X}}(x) = p(s, a)$ as the data distribution and $p_{\mathcal{Y}}(y) = p(s^-)$ as the distribution over the replay buffer i.e. sampling randomly picking some previously visited state. Then

$$\mathcal{L}_{\text{RL InfoNCE}}(f) = \mathbb{E}_{\substack{(s,a)\sim p(s,a), s^+ \sim p^\pi(s^+|s,a) \\ s^-_{1:N} \sim p(s^-)}} \left[\log \frac{e^{f(s,a,s^+)}}{\sum_{i=1}^N e^{f_\theta(s,a,s_i^-)}} \right]. \tag{2}$$

Solving this for f^*, we have that f^* satisfies

$$\exp(f^*(s, a, g)) = \frac{p^\pi(g \mid s, a)}{p(g)c(s, a)}.$$

Therefore, we can learn a goal conditioned policy as

$$\pi^*(s, a, g) = \arg\max_a \exp(f^*(s, a, g)). \tag{3}$$

This can be done using classical RL methods for continuous action spaces, such as DDPG (Lillicrap et al., 2015) or SAC (Haarnoja et al., 2018). Namely, in Algorithm 1, taking L_π as the corresponding policy loss. Crucially, this gives us an efficient continuous time analog to multi-task RL methods

(Borsa et al., 2016; Bai et al., 2025) by allowing us to leverage shared structure across goals. We do so by taking

$$f_{\psi,\phi}(s,a,g) = \langle \phi(s,a), \psi(g) \rangle, \quad \phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d, \ \psi : \mathcal{S} \to \mathbb{R}^d.$$

Planning with Gumbel: With the remarkable success of AlphaGo (Silver et al., 2017), the application of MCTS-based search (Coulom, 2006; Chaslot et al., 2008) in combination with reinforcement learning has been successfully applied to increasingly challenging domains (Silver et al., 2018; Schrittwieser et al., 2020; Brown et al., 2020; Hubert et al., 2021; Antonoglou et al., 2022). However, these domains usually benefit from access to incredibly fast simulation, or the ability to spend a large amount of time searching. In particular, when the number of simulations is less than the number of actions, the MCTS algorithm proposed in Silver et al. (2017) is not necessarily a policy improvement. To remedy this, in Danihelka et al. (2022) they utilize the gumbel trick (Gumbel, 1954; Maddison et al., 2014) i.e. for a categorical distribution $\pi(\cdot \mid s) \in \mathbb{R}^k$, we can sample from $\pi(\cdot \mid s)$ by taking

$$a = \arg\max_a \left(\text{logits}_{\pi(\cdot|s)}(a) + g(a)\right),$$

where $\mathbf{g}$ is a vector of k Gumbel variables with $g(a) \sim \text{Gumbel}(0)$ and $\text{logits}_{\pi(\cdot|s)}(a)$ corresponds to the logit of the ath entry of $\pi(\cdot \mid s)$. Similarly, we can sample n times from $\pi(\cdot \mid s)$ by taking the top n denoted $\text{argtop}(\cdot, n)$ i.e.

$$\mathcal{A}_{topn} = \text{argtop}(\text{logits}_{\pi(\cdot|s)}(a) + g(a), n). \tag{4}$$

The key insight is that for any increasing function $\sigma : \mathbb{R} \to \mathbb{R}$

$$\mathbb{E}_\pi[Q(s,a)] \le \mathbb{E}_{\mathbf{g}}\left[Q\Big(\arg\max_{a \in \mathcal{A}_{topn}} \big(\text{logits}_{\pi(\cdot|s)}(a) + g(a) + \sigma(Q(s,a))\big), s\Big)\right].$$

This leads to the selection

$$A_{n+1} = \arg\max_{a \in \mathcal{A}_{topn}} \big(\text{logits}_{\pi(\cdot|s)}(a) + g(a) + \sigma(Q(s,a))\big). \tag{5}$$

Additionally, we can use the information from computing the estimated Q-values to extract what is hopefully a better policy via the *completed* Q-values defined as

$$\text{Completed}(Q) = \begin{cases} Q(s,a), & \text{if } \mathcal{N}(s,a) > 0 \\ v_\pi(s), & \text{otherwise} \end{cases}$$

and $\pi'(s) = \text{softmax}(\text{logits} + \sigma(\text{Completed}))$. For non-root nodes, we want to ensure the Q-function estimate correctly corresponds to our new policy. To do this, they utilize a deterministic policy to minimize variance

$$\pi_D(s) = \arg\max_a \left[\pi'(a|s) - \frac{\mathcal{N}(s,a)}{\sum_b \mathcal{N}(s,b)}\right]. \tag{6}$$

The derivation can be found in Danihelka et al. (2022, Appendix E). We give pseudo code for the complete algorithm in the form of traditional planning algorithms in 2.

3.2 Contrastive RL + Gumbel Planning

Traditionally, value function based MCTS is focused on ensuring a policy improvement within the *same* environment. However, in this work, we combine Gumbel MCTS with contrastive reinforcement learning and show that this enables zero-shot adaptation in motion-planning problems.

Algorithm 6: `Contrastive Gumbel MCTS`

Input: Reward-free mdp $\mathcal{M}$, evaluation MDP $\mathcal{M}_E \in \mathcal{GO}(\mathcal{M})$

θ, ϕ, ψ

$\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, p_0, \gamma \rangle \leftarrow \mathcal{M}$

while *pre-training* **do**

  $\pi_\theta, f_{\psi,\phi} \leftarrow \texttt{CRL}(\theta, \psi, \phi, \mathcal{M})$

end

$q = \exp f_{\psi,\phi}$

foreach $M \in \mathcal{GO}(\mathcal{M})$ **do**

  $M \leftarrow \langle \mathcal{S}, \mathcal{A}, p_{obs}, r_g, p_0, \gamma \rangle$

  $s \sim p_0(s)$

  while *goal not reached* **do**

    $a \sim \texttt{Selection}(\pi_\theta, q, s, p_{obs}, r_g)$

    $s \sim p_{obs}(s, a)$

  end

end

Algorithm 7: `Ours` :: `Selection`

Input: policy π_θ, value v_ϕ

Input: state s, transition kernel p, reward r

Output: A_{t+1}

foreach $i \in [1, 4]$ **do**

  $\mathcal{A}_{next} \leftarrow k/4$ actions from $\pi(s|g^{(i)})$

end

while $m > 1$ **do**

  foreach $a \in \mathcal{A}_{next}$ **do**

    $visits = \lfloor \frac{n}{\log_2(m) m} \rfloor$

    $\mathcal{T} = \texttt{Simulate}(s, a, visits)$

  end

  $v \leftarrow \texttt{Backup}(\mathcal{T})$

  $m \leftarrow m/2$

  $\mathcal{A}_{next} \leftarrow \text{argtop}_{a \in \mathcal{A}_{next}}(\sigma(\hat{Q}(s, a), m)$

end

$A_{n+1} \leftarrow \arg\max_a \sigma(\hat{Q}(s, a)$

Continuous action space: We first must adapt the discrete Gumbel method presented in Danihelka et al. (2022) to a continuous action space. To do so, we follow a similar procedure to Hubert et al. (2021). Namely, we first sample n actions $\mathcal{A}_{total}$ according to the policy network $\pi_\theta(\cdot|s, g)$. We then use the actions as if it were the complete set of discrete actions according to the previous section. The value and logits at the root node are computed as

$$\tilde{v}(s) = \frac{1}{n} \sum_{a \in \mathcal{A}_{total}} Q_{\psi,\phi}(s, a, g), \quad \text{logits}_{\pi(\cdot|s,g)} = \text{Unif}(n).$$

Since we are sampling from π_θ with replacement,

$$\mathbb{E}_{\mathcal{A}_{total} \sim \pi_\theta}[\tilde{v}(s)] = v_\pi(s),$$

and sampling with (4) still corresponds to π_θ. Since the logits are uniform, they can be omitted from the algorithm as seen in Algorithm 7.

Gumbel Without the Gumbel: In Danihelka et al. (2022), they add the Gumbel variables as noise to ensure sufficient coverage because they distill the resulting policy back into the policy network. Since we are not learning, we instead want to ensure we take the best action at each time step. Therefore, we do not add the exploratory noise but do find it necessary to utilize sequential halving along with the non-root action selection. We tried the sampling rule from traditional Alpha-zero (Silver et al., 2018) and saw significant degradation in performance.

Exploratory actions: We found that sampling from $\pi(\cdot|s, g)$ even when adding noise did not provide diverse enough actions to properly avoid obstacles. Therefore, we added "fallback" actions, where we divide the number of actions by four and sample that many actions from the goal-conditioned policy conditioned on each of the cardinal directions with respect to the goal. This is similar to safe reinforcement learning, where it is often the case that they have a safe "fallback" policy in the event that they cannot produce a safe action (Wagener et al., 2021; Liu et al., 2023). Similar to the reasoning for deterministic non-root action selection, it is important that the Q-values in the search correspond to a policy that the agent will actually take. Therefore, we found it best to not randomly sample the goals but keep them fixed to ensure that the agent would have similar actions available as during simulation. Additionally, the value of each node is initialized as the average of the action produced *soley* by the actions generated from the true goal conditioned policy. This is because in our search we still want states to be valued based on their potential for reaching the goal and this is best conveyed through the value function corresponding to the goal-reaching policy.

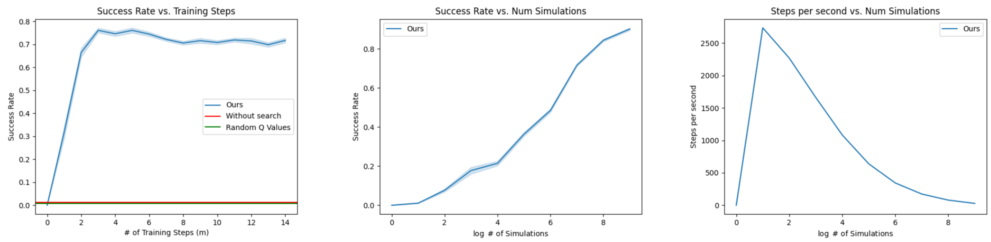

Figure 3: (1): Success rate as a function of 15 different checkpoints taken throughout pre-training the goal-conditioned policy and Q-function for 64 simulations. Random Q-values still uses the final policy to ensure reasonable actions are sampled. Results are averaged over 10 seeds for pre-training and 256 evaluation environments. (2)-(3): We take the final checkpoint of our Q-function and scale the number of simulations. In (2), we see as the number of simulations increases so does the success rate. In (3), we see that this comes at the cost of steps per second.

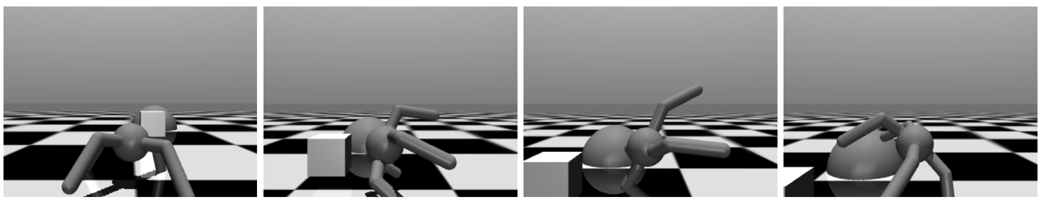

Figure 4: A sample environment configuration and subsequent trajectory where the ant is able to properly avoid the obstacle and reach the goal.

Obstacle penalty: Additionally, along the lines of safe RL, we also observe that inducing a large obstacle hitting penalty is esssential in order to properly guide the search. This is consistent with what has been found in safe reinforcement learning (Massiani et al., 2023). In our experiments, hitting an obstacle induces a reward of -50.

The complete algorithm is given in Algorithm 6.

4 Experiments

We build upon JAX (Bradbury et al., 2021), MCTX (DeepMind et al., 2020), JaxGCRL (Bortkiewicz et al.), and Stoix (Toledo, 2024).

4.1 Environments

For our environment, we use the `ant` (Fu et al., 2020) BRAX (Freeman et al., 2021), Mujoco (Todorov et al., 2012) environment. It has an 8-dimensional continuous action space and 29-dimensional continuous state space. The reward is only on success, where success is the same as prior work (Bortkiewicz et al.; Eysenbach et al., 2022; Zheng et al., 2024). The training environment also follows existing goal-conditioned RL, selecting a random goal each episode.

Task: In the `static ant` environment, both the goals and the obstacles are randomly generated. We generate obstacles between the agent and the goal by randomly perturbing the obstacle by a small margin after it has been placed directly between the ant and the goal. An example configuration can be seen in Figure 4.

4.2 Results

Across 10 seeds, we train Q-functions with contrastive RL and then do evaluation `static ant` environment. Only 10 seeds were ran because little variation was observed. We run 256 evaluations and take the percentage of success. In Figure 3 (1), we see that with only 64 samples, we are able to reach 70% success rate, where the pretrained goal-conditioned policy or pure search both fail dramatically. Furthermore, in (2) and (3), we see that as we increase the number of samples to 256 we observe a greater than 90% success rate while still taking less than 10 ms to make a decision.

5 Discussion

Most real-world applications do not necessitate the level of generality and optimality longed for by the academic community. In this work, we show that designing our algorithm with these applications in mind enables incredibly sample efficiency and adaptibility to rapidly changing environments. Firstly, recent advancements in goal-conditioned reinforcement learning have enabled controlling high degree of freedom robotics to reach states in the environment. We apply this in motion-planning by noting that in many real world applications the map of the environment is available prior to deployment. We pre-train a goal-conditioned Q-function using contrastive reinforcement learning (Eysenbach et al., 2022) and demonstrate its ability to guide search for arbitrary downstream configurations.

Secondly, existing approaches attempt to generalize between entirely different environment configurations, such as new mazes. In practice, the agent rarely has to adapt to such drastic changes. We exploit this insight by using a search heuristic from Danihelka et al. (2022) rather than less efficient control-based algorithms optimized for worst-case configurations. Due to using less samples, we can re-plan at each time step. This enables adaption to obstacle changes within a trajectory along with planning over stochastic dynamics and changes in dynamics in the downstream task. However, fully general representation learning based approaches are not necessarily independent of our approach. In Chen et al. (2019), they require doing full RRT based sampling to generate successful trajectories in order to train their representation. In the future, we could combine these approaches to use faster, less robust methods to generate successful paths in order to accelerate training of their representations.

Future Work: We believe that significant improvements can be made to the search component of this work. Specifically, there should be some way to further guide the search via information about the obstacles, such as our value function estimate. Along with this, due to train-test environment mismatch, the search algorithm should most likely utilize more aggressive exploration. The discussed RL-based search algorithms primarily focus on improving the starting policy on the *same* environment. This generally involves conservative rollout policies, such as (6), where as we must take some actions that would be severely sub-optimal in the training environment. Our heuristic exploratory actions in the work are almost certainly severely suboptimal. Alternatively, one could try to leverage existing goal-conditioned planning to produce subgoals (Nasiriany et al., 2019; Dubey et al., 2021; Chane-Sane et al., 2021) that avoid obstacles. We also hope to extend to more realistic models potentially learning a world model in the obstacle free environment in combination with a map as defined in Chen et al. (2019), or obstacle detection.

Ultimately, we hope that this work introduces a new paradigm for motion-planning problems, where we use obstacle-free information via unsupervised reinforcement learning to accelerate the search in the presence of real-world obstacles.

References

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 22–31. JMLR.org, 2017.

Siddhant Agarwal, Harshit Sikchi, Peter Stone, and Amy Zhang. Proto Successor Measure: Representing the Space of All Possible Solutions of Reinforcement Learning, November 2024. URL http://arxiv.org/abs/2411.19418. arXiv:2411.19418 [cs].

Eitan Altman. *Constrained Markov Decision Processes: Stochastic Modeling*. Routledge, Boca Raton, 1 edition, December 2021. ISBN 978-1-315-14022-3. DOI: 10.1201/9781315140223. URL https://www.taylorfrancis.com/books/9781315140223.

Ioannis Antonoglou, Julian Schrittwieser, Sherjil Ozair, Thomas K Hubert, and David Silver. Planning in stochastic environments with a learned model. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=X6D9bAHhBQ1.

Qinbo Bai, Amrit Singh Bedi, Mridul Agarwal, Alec Koppel, and Vaneet Aggarwal. Achieving zero constraint violation for constrained reinforcement learning via primal-dual approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 3682–3689, 2022.

Yitao Bai, Sihan Zeng, Justin Romberg, and Thinh T. Doan. Accelerating multi-task temporal difference learning under low-rank representation, 2025. URL https://arxiv.org/abs/2503.02030.

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Diana Borsa, Thore Graepel, and John Shawe-Taylor. Learning shared representations in multi-task reinforcement learning, 2016. URL https://arxiv.org/abs/1603.02041.

Diana Borsa, André Barreto, John Quan, Daniel Mankowitz, Rémi Munos, Hado Van Hasselt, David Silver, and Tom Schaul. Universal successor features approximators. *arXiv preprint arXiv:1812.07626*, 2018.

Michał Bortkiewicz, Władysław Pałucki, Vivek Myers, Tadeusz Dziarmaga, Tomasz Arczewski, Łukasz Kuciński, and Benjamin Eysenbach. Accelerating goal-conditioned reinforcement learning algorithms and research. In *The Thirteenth International Conference on Learning Representations*.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, et al. Jax: Autograd and xla. *Astrophysics Source Code Library*, pp. ascl–2111, 2021.

Noam Brown, Anton Bakhtin, Adam Lerer, and Qucheng Gong. Combining deep reinforcement learning and search for imperfect-information games. *Advances in neural information processing systems*, 33:17057–17069, 2020.

John Canny and John Reif. New lower bound techniques for robot motion planning problems. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pp. 49–60. IEEE, 1987.

Wilka Carvalho, Momchil S. Tomov, William de Cothi, Caswell Barry, and Samuel J. Gershman. Predictive representations: building blocks of intelligence, July 2024. URL http://arxiv.org/abs/2402.06590. arXiv:2402.06590 [cs].

Elliot Chane-Sane, Cordelia Schmid, and Ivan Laptev. Goal-Conditioned Reinforcement Learning with Imagined Subgoals, July 2021. URL http://arxiv.org/abs/2107.00541. arXiv:2107.00541 [cs].

Guillaume Chaslot, Sander Bakkes, Istvan Szita, and Pieter Spronck. Monte-carlo tree search: A new framework for game ai. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 4, pp. 216–217, 2008.

Binghong Chen, Bo Dai, Qinjie Lin, Guo Ye, Han Liu, and Le Song. Learning to plan in high dimensions via neural exploration-exploitation trees. *arXiv preprint arXiv:1903.00070*, 2019.

Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pp. 72–83. Springer, 2006.

Ivo Danihelka, Arthur Guez, Julian Schrittwieser, and David Silver. Policy improvement by planning with gumbel. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=bERaNdoegnO.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993. DOI: 10.1162/neco.1993.5.4.613.

DeepMind, Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Laurent Sartran, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Milo ˇ s Stanojević, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. The DeepMind JAX Ecosystem, 2020. URL http://github.com/deepmind.

Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1: 269–271, 1959.

Rohit K. Dubey, Samuel S. Sohn, Jimmy Abualdenien, Tyler Thrash, Christoph Hoelscher, André Borrmann, and Mubbasir Kapadia. SNAP:Successor Entropy based Incremental Subgoal Discovery for Adaptive Navigation. In *Proceedings of the 14th ACM SIGGRAPH Conference on Motion, Interaction and Games*, MIG '21, pp. 1–11, New York, NY, USA, November 2021. Association for Computing Machinery. ISBN 978-1-4503-9131-3. DOI: 10.1145/3487983.3488292. URL https://dl.acm.org/doi/10.1145/3487983.3488292.

Hugh Durrant-Whyte, Nicholas Roy, and Pieter Abbeel. *Cross-Entropy Randomized Motion Planning*, pp. 153–160. 2012.

Yonathan Efroni, Shie Mannor, and Matteo Pirotta. Exploration-exploitation in constrained mdps. *arXiv preprint arXiv:2003.02189*, 2020.

Ben Eysenbach, Russ R Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Ruslan Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=vGQiU5sqUe3.

Benjamin Eysenbach, Tianjun Zhang, Ruslan Salakhutdinov, and Sergey Levine. Contrastive Learning as Goal-Conditioned Reinforcement Learning, February 2023. URL http://arxiv.org/abs/2206.07568. arXiv:2206.07568 [cs].

C Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax–a differentiable physics engine for large scale rigid body simulation. *arXiv preprint arXiv:2106.13281*, 2021.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning, 2020. URL https://arxiv.org/abs/2010.01083.

Shangding Gu, Long Yang, Yali Du, Guang Chen, Florian Walter, Jun Wang, and Alois Knoll. A review of safe reinforcement learning: Methods, theories, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):11216–11235, 2024. DOI: 10.1109/TPAMI.2024.3457538.

Emil Julius Gumbel. *Statistical theory of extreme values and some practical applications: a series of lectures*, volume 33. US Government Printing Office, 1954.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 297–304. JMLR Workshop and Conference Proceedings, March 2010. URL https://proceedings.mlr.press/v9/gutmann10a.html. ISSN: 1938-7228.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. DOI: 10.1109/TSSC.1968.300136.

Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Mohammadamin Barekatain, Simon Schmitt, and David Silver. Learning and planning in complex action spaces. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4476–4486. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/hubert21a.html.

Jinwook Huh and Daniel D. Lee. Efficient sampling with q-learning to guide rapidly exploring random trees. *IEEE Robotics and Automation Letters*, 3(4):3868–3875, 2018. DOI: 10.1109/LRA.2018.2856927.

Brian Ichter, James Harrison, and Marco Pavone. Learning sampling distributions for robot motion planning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7087–7094. IEEE, 2018.

L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996. DOI: 10.1109/70.508439.

Beomjoon Kim, Leslie Pack Kaelbling, and Tomas Lozano-Perez. Guiding the search in continuous state-action spaces by learning an action sampling distribution from off-target samples. *arXiv preprint arXiv:1711.01391*, 2017.

Dexter Kozen and Chee-Kang Yap. Algebraic cell decomposition in nc. In *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*, pp. 515–521. IEEE, 1985.

Voemir Kunchev, Lakhmi Jain, Vladimir Ivancevic, and Anthony Finn. Path Planning and Obstacle Avoidance for Autonomous Mobile Robots: A Review. In Bogdan Gabrys, Robert J. Howlett, and Lakhmi C. Jain (eds.), *Knowledge-Based Intelligent Information and Engineering Systems*, pp. 537–544, Berlin, Heidelberg, 2006. Springer. ISBN 978-3-540-46539-3. DOI: 10.1007/11893004_70.

Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Minghuan Liu, Menghui Zhu, and Weinan Zhang. Goal-conditioned reinforcement learning: Problems and solutions. *arXiv preprint arXiv:2201.08299*, 2022.

Tao Liu, Ruida Zhou, Dileep Kalathil, P. R. Kumar, and Chao Tian. Learning Policies with Zero or Bounded Constraint Violation for Constrained MDPs, January 2023. URL http://arxiv.org/abs/2106.02684. arXiv:2106.02684 [cs].

Kendall Lowrey, Aravind Rajeswaran, Sham Kakade, Emanuel Todorov, and Igor Mordatch. Plan Online, Learn Offline: Efficient Learning and Exploration via Model-Based Control, January 2019. URL http://arxiv.org/abs/1811.01848. arXiv:1811.01848 [cs].

Tomás Lozano-Pérez and Michael A Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, 1979.

Zhuang Ma and Michael Collins. Noise Contrastive Estimation and Negative Sampling for Conditional Models: Consistency and Statistical Efficiency, September 2018. URL http://arxiv.org/abs/1809.01812. arXiv:1809.01812 [cs].

Marlos C. Machado, Andre Barreto, Doina Precup, and Michael Bowling. Temporal Abstraction in Reinforcement Learning with the Successor Representation, April 2023. URL http://arxiv.org/abs/2110.05740. arXiv:2110.05740 [cs].

Chris J Maddison, Daniel Tarlow, and Tom Minka. A* sampling. *Advances in neural information processing systems*, 27, 2014.

Pierre-François Massiani, Steve Heim, Friedrich Solowjow, and Sebastian Trimpe. Safe Value Functions. *IEEE Transactions on Automatic Control*, 68(5):2743–2757, May 2023. ISSN 0018-9286, 1558-2523, 2334-3303. DOI: 10.1109/TAC.2022.3200948. URL http://arxiv.org/abs/2105.12204. arXiv:2105.12204 [eess].

Vivek Myers, Catherine Ji, and Benjamin Eysenbach. Horizon Generalization in Reinforcement Learning, January 2025. URL http://arxiv.org/abs/2501.02709. arXiv:2501.02709 [cs].

Soroush Nasiriany, Vitchyr H. Pong, Steven Lin, and Sergey Levine. Planning with Goal-Conditioned Policies, November 2019. URL http://arxiv.org/abs/1911.08453. arXiv:1911.08453 [cs].

Junhyuk Oh, Satinder Singh, and Honglak Lee. Value prediction network. *Advances in neural information processing systems*, 30, 2017.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation Learning with Contrastive Predictive Coding, January 2019. URL http://arxiv.org/abs/1807.03748. arXiv:1807.03748 [cs].

Andreas Orthey, Constantinos Chamzas, and Lydia E Kavraki. Sampling-based motion planning: A comparative review. *Annual Review of Control, Robotics, and Autonomous Systems*, 7, 2023.

Seohong Park, Kevin Frans, Deepinder Mann, Benjamin Eysenbach, Aviral Kumar, and Sergey Levine. Horizon Reduction Makes RL Scalable, June 2025. URL http://arxiv.org/abs/2506.04168. arXiv:2506.04168 [cs].

Santiago Paternain, Miguel Calvo-Fullana, Luiz F. O. Chamon, and Alejandro Ribeiro. Safe Policies for Reinforcement Learning via Primal-Dual Methods, January 2022. URL http://arxiv.org/abs/1911.09101. arXiv:1911.09101 [eess].

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In *International conference on machine learning*, pp. 5062–5071. PMLR, 2019.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., USA, 1st edition, 1994. ISBN 0471619779.

Lun Quan, Luxin Han, Boyu Zhou, Shaojie Shen, and Fei Gao. Survey of uav motion planning. *IET Cyber-systems and Robotics*, 2(1):14–21, 2020.

Ahmed H Qureshi, Anthony Simeonov, Mayur J Bency, and Michael C Yip. Motion planning networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 2118–2124. IEEE, 2019.

John H. Reif. Complexity of the mover's problem and generalizations. In *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*, pp. 421–427, 1979. DOI: 10.1109/SFCS.1979.10.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pp. 1312–1320. PMLR, 2015.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588 (7839):604–609, December 2020. ISSN 1476-4687. DOI: 10.1038/s41586-020-03051-4. URL http://dx.doi.org/10.1038/s41586-020-03051-4.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018. DOI: 10.1126/science.aar6404. URL https://www.science.org/doi/abs/10.1126/science.aar6404.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value iteration networks. *Advances in neural information processing systems*, 29, 2016.

Siyu Teng, Xuemin Hu, Peng Deng, Bai Li, Yuchen Li, Yunfeng Ai, Dongsheng Yang, Lingxi Li, Zhe Xuanyuan, Fenghua Zhu, et al. Motion planning for autonomous driving: The state of the art and future perspectives. *IEEE Transactions on Intelligent Vehicles*, 8(6):3692–3711, 2023.

Garrett Thomas, Yuping Luo, and Tengyu Ma. Safe Reinforcement Learning by Imagining the Near Future.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Edan Toledo. Stoix: Distributed single-agent reinforcement learning end-to-end in jax, April 2024. URL https://github.com/EdanToledo/Stoix.

Ahmed Touati and Yann Ollivier. Learning One Representation to Optimize All Rewards, October 2021. URL http://arxiv.org/abs/2103.07945. arXiv:2103.07945 [cs, math].

Ahmed Touati, Jérémy Rapin, and Yann Ollivier. Does Zero-Shot Reinforcement Learning Exist?, March 2023. URL http://arxiv.org/abs/2209.14935. arXiv:2209.14935 [cs].

Nelson Vithayathil Varghese and Qusay H Mahmoud. A survey of multi-task deep reinforcement learning. *Electronics*, 9(9):1363, 2020.

Nolan C. Wagener, Byron Boots, and Ching-An Cheng. Safe Reinforcement Learning Using Advantage-Based Intervention. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 10630–10640. PMLR, July 2021. URL https://proceedings.mlr.press/v139/wagener21a.html. ISSN: 2640-3498.

Kevin Wang, Ishaan Javali, Michał Bortkiewicz, Tomasz Trzciński, and Benjamin Eysenbach. 1000 Layer Networks for Self-Supervised RL: Scaling Depth Can Enable New Goal-Reaching Capabilities, March 2025. URL http://arxiv.org/abs/2503.14858. arXiv:2503.14858 [cs].

Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1433–1440, 2016. DOI: 10.1109/ICRA.2016.7487277.

Andrew Zhao, Matthieu Lin, Yangguang Li, Yong-jin Liu, and Gao Huang. A mixture of surprises for unsupervised reinforcement learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 26078–26090. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/a7667ee5d545a43d2f0fda98863c260e-Paper-Conference.pdf.

Chongyi Zheng, Ruslan Salakhutdinov, and Benjamin Eysenbach. Contrastive Difference Predictive Coding, February 2024. URL http://arxiv.org/abs/2310.20141. arXiv:2310.20141 [cs].

Matt Zucker, James Kuffner, and J. Andrew Bagnell. Adaptive workspace biasing for sampling-based planners. In *2008 IEEE International Conference on Robotics and Automation*, pp. 3757–3762, 2008. DOI: 10.1109/ROBOT.2008.4543787.

Supplementary Materials

The following content was not necessarily subject to peer review.

A Experiment Details

For learning the goal conditioned policy and value function, we used the default hyperparameters provided in Bortkiewicz et al. We used a deeper architecture as recommended by Wang et al. (2025). Specifically,

Hyperparameter	**Value**	**Description**
Activation function	`swish`	Activation function.
Number of layers	8	Number of layers in the neural network.
Hidden units per layer	1024	Number of neurons in each hidden layer.
Skip connections	2	Number of skip connections

Table 1: Hyperparameter settings used in the experiments.

We provide the search hyperparameters below

Hyperparameter	**Value**	**Description**
Num simulations	64	Number of simulations used in the search.
Num samples	8	Number of actions available at each node in search.
Obstacle penalty	-50	Reward penalty for hitting an obstacle

Table 2: Hyperparameter settings used in the experiments.

B Failed Experiments

- *Exploratory Bonus:* Instead of Gumbel we tried to add a UCB bonus to encourage visits at the root node of unvisited actions.
- *Deterministic Actions:* We tried the deterministic action selection around the unit circle around the agent. We did not observe a substantial difference.