# CONTINUOUS CONTROL ON TIME

**Tianwei Ni**[*]
twni2016@gmail.com

**Eric Jang**[*]
ericjang2004@gmail.com

## ABSTRACT

The physical world evolves continuously in time. Most prior works on reinforcement learning cast continuous-time environments into a discrete-time Markov Decision Process (MDP), by discretizing time into constant-width decision intervals. In this work, we propose Continuous-Time-Controlled MDPs (CTC-MDP), a continuous-time decision process that permits the agent to decide how long each action will last in the physical time of the environment. However, reinforcement learning in vanilla CTC-MDP may result in agents learning to take infinitesimally small time scales for each action. To prevent such degeneration and allow users to control the computation budget, we further propose CTC-MDPs with a constraint on the average time scale over a given threshold. We hypothesize that constrained CTC-MDPs will allow agents to "budget" fine-grained time scales to states where it may need to adjust actions quickly, and coarse-grained time scales to states where it can get away with a single decision. We evaluate our new CTC-MDP framework (with and without constraint) on the standard MuJoCo benchmark.

## 1 INTRODUCTION

Table 1: **Glossary of concepts related to *time*, in existing MDP framework for continuous control.**

| Quantity | Notation | Definition |
|---|---|---|
| Frame skip | $n \in \mathbb{N}_+$ | number of frames between two consecutive decisions |
| Time scale | $\delta \in \mathbb{R}_+$ | physical elapsed time between two consecutive frames |
| Decision time scale | $n * \delta$ | physical elapsed time between two consecutive decisions |
| Episode duration | $H \in \mathbb{R}_+$ | total physical duration (*e.g.* in secs) for an episode |
| Episode decision steps | $\lceil H/(n*\delta) \rceil$ | total number of decisions made by a policy |
| Episode interaction steps | $\lceil H/\delta \rceil$ | total number of interactions (*i.e.* frames) for an episode |

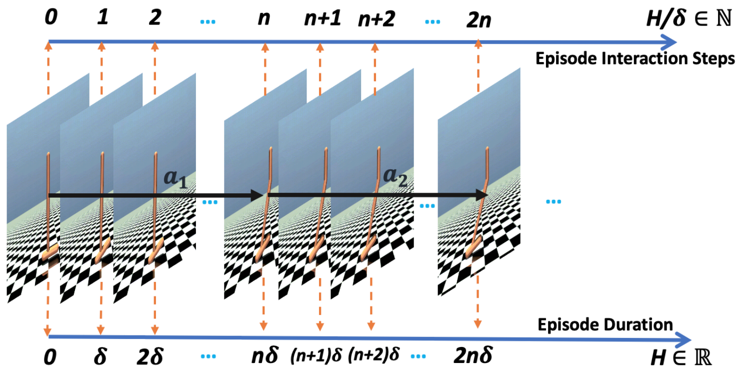

Figure 1: **Illustration of the concepts in Table 1.** We show the time evolution in the Hopper environment. The top time axis is the episode interaction steps, and the bottom time axis is the episode duration. Each decision $a_i$ $(i = 1, 2, \dots)$ takes effective for $n\delta$ secs, *i.e.* $n$ frames.

[*]Work was done in their spare time.

Reinforcement learning (RL) mostly focuses on discrete-time Markov Decision Processes (MDP) where the environment advances from one state to another, with time indexed by integers. However, many real-world control problems are inherently continuous-time. For example, the state of a physical robot is modeled as a continuous-time differential equation and is simulated via numerical integration (Todorov et al., 2012). To simplify the design of the sequential decision making framework, most RL environments for continuous control (Brockman et al., 2016; Tassa et al., 2018) assume the physical elapsed time between two consecutive decisions is fixed as a constant time interval.

In most simulated environments, there are two basic quantities related to time, namely *frame skip* (also known as *action repetition*) $n \in \mathbb{N}_+$ and *time scale* $\delta \in \mathbb{R}_+$. At each step, a policy makes a decision, *i.e*. selects an action that is emitted into a simulator. The simulator simulates the dynamics with $\delta$ physical time for $n$ consecutive internal steps (frames) with that same action. Then it returns the last frame to the policy for the next decision. Thus, the elapsed physical time between two decisions, *i.e*. decision time scale, is $n * \delta$. We summarize the related concepts in Table 1 and Fig. 1. Normally, the frame skip $n$ and time scale $\delta$ are set with heuristics by practitioners. Empirically, the values of frame skip (Braylan et al., 2015; Yarats et al., 2019; Reda et al., 2020; Andrychowicz et al., 2020) and time scale (Tallec et al., 2019) have drastic effects on agent performance, and their optimal values vary from task to task, requiring much human effort on tuning. In principle, small frame skips and time scales have the potential to learn a better policy, as actions can be adjusted more frequently. However, the tradeoff is that the MDP becomes harder to optimize due to vanishing reward signals through a longer episode (Tallec et al., 2019). Conversely, large frame skips and time scales can reduce the episode decision steps (defined in Table 1), accelerating exploration and learning (McGovern et al., 1997; Kalyanakrishnan et al., 2021), but incurs more simulation error and implausible physical dynamics, especially in discontinuous dynamics such as contact modeling (Moreau & Panagiotopoulos, 2014).

A natural question follows: what if we enable RL agents to automate the selection of frame skip and time scale per task? Can we relax the assumption of a fixed frame skip and simulation time scale throughout an episode? Intuitively, intelligent agents like humans are capable of selecting different time scales for their actions depending on the contexts (Klapp, 1995; 2003; Schneider et al., 2011). A number of recent works in deep RL (Durugkar et al., 2016; Vezhnevets et al., 2016; Lakshminarayanan et al., 2017; Sharma et al., 2017) propose automating the learning of the *frame skip* $n$, but not the *time scale* $\delta$, which we study in this work. In this paper, we aim to build agents that can learn the **continuous time scale** $\delta$ for continuous control problems. Our eventual goal for the agents is to solve the tasks within the fewest *episode interaction steps* (defined in Table 1), which are the actual number of interactions between actions and environments. Such a goal cannot be realized by learning the frame skip (Sharma et al., 2017), because the episode interaction steps $\lceil H/\delta \rceil$ is independent of it.

To formulate the continuous control on time scale, we first propose a generic MDP framework named **continuous-time-controlled MDP** (**CTC-MDP** in short, Def. 3.2), a time discretization of continuous-time optimal control problem. CTC-MDP allows a policy to select continuous time scale depending on the state in each step, like an extra degree of freedom, within a predefined interval. However, vanilla RL algorithms for solving CTC-MDPs may converge to always choosing a minimal value. This is also found in previous work on action repetition (Sharma et al., 2017) and hierarchical RL (Bacon et al., 2017). Inspired by the regularization in options framework (Harb et al., 2017), we put a constraint on the maximal number of episode interaction steps upon CTC-MDP, named **constrained CTC-MDP**. This constraint will not only mitigate the issue of degeneration, but also give the RL practitioners an interface to control the agent behavior. For example, for applications with accurate simulation or real robots, users can set a small value for the episode interaction steps for cheap computation, and vice versa.

We conduct experiments on several continuous control environments to show the effectiveness of the proposed constrained CTC-MDP and corresponding RL algorithms. Given roughly the same number of episode interaction steps, policies with constrained continuous time control can outperform the counterpart with fixed time scale or unconstrained setting. Behavior analysis on continuous time control verifies our intuition that the policy will tend to exploit the dynamics by taking a suitable time scale conditioned on the physical states. Finally, we also discuss the limitations of our new framework for future work.

## 2 Related Work

In this section, we introduce the two lines of our related work. The first is learning to repeat actions, *i.e.* automating the selection of frame skips. The other is continuous-time RL and semi-MDPs, which assumes that the continuous time scale can change but is controlled by the environment. Our work differs from them in that we aim to learn to control the continuous time scale. We also provide an algorithm to learn to solve tasks within episode interaction steps of any given value for regularization.

**Learning to repeat actions.** Action repetition, also known as *macro-actions* and *action persistence*, appears as an integer hyperparameter named *frame skip* in many environment simulators. For example, DQN (Mnih et al., 2013) uses frame skip number of 4 in Atari games (Bellemare et al., 2013), *i.e.* repeating the same actions for 4 consecutive frames. Normally, this hyperparameter is pre-defined and fixed by the simulator. It is critical to achieve good agent performance; optimal frame skip empirically varies from task to task almost everywhere, including Atari games (Braylan et al., 2015), coordinated-state-based locomotion (Reda et al., 2020) and pixel-based robots (Yarats et al., 2019). Large frame skips can reduce the episode decision steps to accelerate exploration and learning, with the assumption that consecutive states are similar (McGovern et al., 1997; Kalyanakrishnan et al., 2021; Metelli et al., 2020). Motivated by this, a number of works (Durugkar et al., 2016; Vezhnevets et al., 2016; Lakshminarayanan et al., 2017; Sharma et al., 2017; Hessel et al., 2019; Lee et al., 2020; Grigsby et al., 2021; Chen et al., 2021; Yu et al., 2021; Biedenkapp et al., 2021) automate the selection of frame skip per step, and show superior performance over the fixed number setting. Learning to repeat actions can be thought as a special case of hierarchical RL (Dayan & Hinton, 1993; Sutton et al., 1999) where the learned options or high-level policies select current primitive action and its repeated times.

**Continuous-time RL and semi-MDPs.** While most deep RL algorithms focus on discrete-time MDPs, there are also a rich history of continuous-time RL (Baird, 1994; Bradtke & Duff, 1995; Munos & Bourgine, 1997; Doya, 2000; Ramstedt & Pal, 2019; Xiao et al., 2020) where the state, action, and reward are indexed over continuous time instead of discrete integers. The time scales are often set as an *infinitesimal* number to approximate the underlying continuous-time optimal control problem (Bryson & Ho, 2018) (Def. 3.1). Semi-MDPs (Ross, 2013; Howard, 2012; Du et al., 2020) generalize the continuous-time RL problem to allow the time scales to vary through one episode, but the time scale transition probabilities are preset by the environment. Similar to the frame skip, a small time scale adds difficulty to policy optimization (Tallec et al., 2019).

## 3 Continuous-Time-Controlled MDP

We derive CTC-MDPs by beginning with the classic formulation of continuous-time optimal control (Def. 3.1, then introduce an approximation through time discretization, *i.e.* continuous-time-controlled MDPs (CTC-MDPs, Def. 3.2). Finally, we provide a deep RL algorithm for the CTC-MDPs.

### 3.1 Formulation: Discretizing the Continuous Time

As a background, we first introduce the raw formulation of continuous control.

**Definition 3.1** (**Continuous-Time Optimal Control (CTOC)**). CTOC (Bryson & Ho, 2018) can be defined as a tuple $(\mathcal{S}, \mathcal{A}, f, \rho_0, r, \gamma, H)$: $\mathcal{S}$ and $\mathcal{A}$ are state and action space, $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the dynamics in the form of differential equations, $\rho_0$ is the initial state distribution, and $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\gamma \in (0, 1]$ is the discount factor, and $H \in (0, \infty]$ is the maximal time horizon.[1] The RL objective is to find a policy $\pi : \mathcal{S} \to \mathrm{Distr}(\mathcal{A})$ maximizing the discounted cumulative rewards:

$$J^{\mathrm{CTOC}}(\pi) := \int_0^H \gamma^t r(s(t), a(t))\, dt \tag{1}$$

$$\text{s.t.} \quad s(0) \sim \rho_0(s), \quad \dot{s}(t) = f(s(t), a(t)), \quad a(t) \sim \pi(s(t)), \quad \forall t \in [0, H] \tag{2}$$

□

[1]We allow early stopping in one episode that has episode horizon less than $H$. Although it is equivalent to have zero rewards after the stop in CTOC, it will make a difference in constrained MDP (Sec. 4.1).

In optimal control literature, the dynamics $f$ are usually assumed to be known or have some simple form (*e.g.* affine w.r.t. control), and the reward $r$ is also known or quadratic, so that the problem could be analytically solved via Hamilton–Jacobi–Bellman equation.

In deep reinforcement learning, both dynamics and reward are unknown to the policy (Sutton & Barto, 2018). The optimal policy should be learned from the collected trajectories through *interaction* with environment. To computationally interact with a CTOC environment, we replace the integral with a summation over a finite number of time scales. Here we propose a generic and flexible time discretization formulation as follows.

**Definition 3.2** (**Continuous-Time-Controlled MDP (CTC-MDP)**). We consider a generic formulation on time discretization: given a CTOC problem $(\mathcal{S}, \mathcal{A}, f, \rho_0, r, \gamma, H)$, CTC-MDP can be represented as $(\mathcal{S}, \mathcal{A}_{\text{CTC}}, f, \rho_0, r, \gamma, H)$ where the new action space $\mathcal{A}_{\text{CTC}} = \mathcal{A} \cup \mathcal{A}_\delta$ is augmented with an extra continuous 1-dimensional action space on **time scale** $\mathcal{A}_\delta$. For simplicity, we consider $\mathcal{A}_\delta = [\delta_{\min}, \delta_{\max}]$ where $0 < \delta_{\min} \leq \delta_{\max}$ are in physical unit of seconds. CTC-MDP can be constructed for any frame skip, but for simplicity we also assume the frame skip $n = 1$ so that the time scale is equal to decision time scale (Table 1).

The optimal control objective of CTC-MDP can be viewed as an approximation of that of CTOC: to find an optimal policy $\pi : \mathcal{S} \to \text{Distr}(\mathcal{A}_{\text{CTC}})$

$$\begin{aligned} J^{\text{CTC-MDP}}(\pi) &:= \sum_{t=1}^{T} \gamma^{\sum_{k=1}^{t} \delta_k} \underbrace{r(s_t, a_t)\delta_t}_{R(s_t,a_t,\delta_t)} \\ &\approx \sum_{t=1}^{T} \gamma^{\sum_{k=1}^{t} \delta_k} \int_{\sum_{k=1}^{t} \delta_k}^{\sum_{k=1}^{t+1} \delta_k} r(s(t), a(t))\, dt \approx J^{\text{CTOC}}(\pi) \end{aligned} \tag{3}$$

$$\text{s.t.} \quad s_0 \sim \rho_0(s), \quad s_{t+1} = F(f, s_t, a_t, \delta_t)^{2}, \quad (a_t, \delta_t) \sim \pi(s_t), \quad \sum_{t=1}^{T} \delta_t \leq H \tag{4}$$

where the dynamics is approximated by some numerical integrator $F$ (Bulirsch et al., 1991), in the simple case that it could be forward Euler method, *i.e.* $F(f, s, a, \delta) := s + f(s, a)\delta$ . The reward function is approximated by linearization, *i.e.* $R(s, a, \delta) = r(s, a)\delta$. The time scale $\delta_t$ and episode interaction steps $T$ follow the constraint that $\sum_{t=1}^{T} \delta_t \leq H$. □

Intuitively, the policy in CTC-MDP can control the time scale $\delta_t \in \mathcal{A}_\delta$ in each time step $t$. Between two consecutive time steps $t, t+1$, the action at the previous time step $a_t$ would hold in the dynamics during that period, *i.e.* repeat the action for $\delta_t$ physical seconds.

**Connection with previous formulations.** There are several alternatives to discretizing the time in continuous-time optimal control problem. Here we list the main previous approaches in Table 2. (Discrete-time) MDPs are a standard approach that assumes the time scales (and frame skips) are constant. Action repetition frameworks (Sharma et al., 2017) extends MDPs, where the policy can control the frame skip, but not the time scale. Moreover, when $n * \delta_{\min} \leq 1 < n \leq n * \delta_{\max}$, it is a strict subset of CTC-MDP. Semi-MDPs (Bradtke & Duff, 1995; Ross, 2013) support adaptive continuous time scales, but it is determined by the environment, not the policy. Therefore, CTC-MDP can be viewed as a generalization of the MDPs above, which indicates the performance of an optimal policy in CTC-MDP should be at least as good as that in those MDPs given the same time scale space.

## 3.2 Reinforcement Learning on CTC-MDP

To solve CTC-MDPs with RL, we can define a policy evaluation operator $\mathcal{T}^\pi$ and Bellman optimality operator $\mathcal{T}^\star$ similar to those in MDPs. Let Q value function $Q : \mathcal{S} \times (\mathcal{A} \times \mathcal{A}_\delta) \to \mathbb{R}$:

$$\mathcal{T}^\pi Q^\pi(s, a, \delta) := R(s, a, \delta) + \gamma^{\delta_t} \mathbb{E}_{s' \sim F(f,s,a,\delta),(a',\delta') \sim \pi(s')}[Q^\pi(s', a', \delta')] \tag{5}$$

$$\mathcal{T}^\star Q^\star(s, a, \delta) := R(s, a, \delta) + \gamma^{\delta_t} \mathbb{E}_{s' \sim F(f,s,a,\delta)} \left[ \max_{a',\delta'} Q^\star(s', a', \delta') \right] \tag{6}$$

[2]We refer to $s(t)$ as the state at physical time $t \in \mathbb{R}$, and $s_t$ as the $t$-th state where $t \in \mathbb{N}$, same applied to action $a$.

Table 2: **Comparison among different MDPs on the spaces of frame skip and time scale.** The math symbols follow $\delta \in \mathbb{R}_+$, $n \in \mathbb{N}_+$ and $\delta_{\min}, \delta_{\max} \in \mathbb{R}_+$.

| **Formulation** | Space of **Frame Skip** | Space of **Time Scale** | **Can Control Time Scale?** |
|---|---|---|---|
| (Discrete-time) MDP (Bellman, 1957) | $\{n\}$ | $\{\delta\}$ | ✗ |
| Action Repetition Learning (Sharma et al., 2017) | $\{1, \ldots, n\}$ | $\{\delta\}$ | ✗ |
| Semi-MDP (Bradtke & Duff, 1995; Ross, 2013) | $\{n\}$ | $[\delta_{\min}, \delta_{\max}]$ | ✗ |
| Continuous-Time-Controlled MDP (Ours) | $\{n\}$ | $[\delta_{\min}, \delta_{\max}]$ | ✓ |

In the tabular case, the operators are contractive since $\gamma^\delta < \gamma^0 = 1$. In general cases, we can train deep RL policies for continuous control such as DDPG (Lillicrap et al., 2015) based on the operators.

However, there exists a potential issue: the optimal policy is free to make the time scales as small as it likes, without penalty. Although this may be effective in realizing higher overall returns, this will excessively slow down the simulation cost of each episode, and result in more environment interaction steps. This is typically followed by lower performance because the agent does not experience as many episode resets as needed for good exploration. We demonstrate this phenomenon in Sec. 5.1. In fact, this is not limited to CTC-MDPs; action repetition frameworks without penalty (Sharma et al., 2017, Table 6) also suffer from expensive data collection due to small frame skips. In the next section, we provide a solution to it, by introducing a constraint on the maximum of episode interaction steps to encourage efficient task solving.

## 4 Constrained CTC-MDP

### 4.1 Formulation: Constraint on Episode Interaction Steps

We want the agent to *intelligently* choose an appropriate time scale for each state, instead of learning to always predict the minimal time scale. We implement this by imposing a constraint on the number of episode interaction steps $T$. This is similar to the idea of regularization on the horizon of options (Harb et al., 2017). We could directly constrain $T$ to be smaller than a target value, but it will have the risk of encouraging the policy to stop early as a local optima, given that we allow the episode to stop early in the original CTOC problem (Def 3.1).

Instead, we constrain the average time scale (involving the episode interaction steps $T$) posed on the CTC-MDP objective (Eq. 3):

$$\frac{\sum_{t=1}^{T} \delta_t}{T} \geq \Delta \tag{7}$$

where $\Delta$ is the **target average timescale**, a constant. Note that the episode horizon $\sum_{t=1}^{T} \delta_t$ is not a constant, but less than maximum value $H$, *i.e.* $\sum_{t=1}^{T} \delta_t \leq H$. Thus, the constraint above controls the *upper bound* of average episode steps, with $T \leq \frac{H}{\Delta}$.

**Definition 4.1** (**Constrained CTC-MDP**). By transforming CTC-MDP (Def. 3.2) with the constraint (Eq. 7 involving the constant $\Delta$) into an unconstrained optimization by the method of Lagrange multipliers, we can define the objective of constrained CTC-MDP $J^{\text{CTC-MDP}}(\pi, \lambda; \Delta)$:

$$\min_{\lambda \geq 0} \max_{\pi} J^{\text{CTC-MDP}}(\pi, \lambda; \Delta) := \sum_{t=1}^{T} \gamma^{\sum_{k=1}^{t} \delta_k} R(s_t, a_t, \delta_t) + \lambda \left( \sum_{t=1}^{T} \delta_t - T\Delta \right) \tag{8}$$

Constrained CTC-MDP objective can be alternatively trained by the policy $\pi$ and Lagrange multiplier $\lambda$ that balances the return and episode interaction steps:

$$\max_{\pi} J^{\text{CTC-MDP}}(\pi; \lambda, \Delta) \approx \sum_{t=1}^{T} \gamma^{\sum_{k=1}^{t} \delta_k} \underbrace{\left(R(s_t, a_t, \delta_t) + \overbrace{\lambda(\delta_t - \Delta)}^{\text{penalty term}}\right)}_{R'(s_t, a_t, \delta_t, \lambda)} \tag{9}$$

$$\min_{\lambda \geq 0} J^{\text{CTC-MDP}}(\lambda; \pi, \Delta) := \lambda \left( \frac{1}{T} \sum_{t=1}^{T} \delta_t - \Delta \right) \tag{10}$$

where we assume $\gamma \to 1$ to incorporate the penalty term into the new rewards $R'$. If we let $\lambda \equiv 0$, then the constrained CTC-MDP reduces to CTC-MDP. □

### 4.2 Reinforcement Learning on Constrained CTC-MDP

Now we consider practical RL algorithms to solve the constrained CTC-MDP (Def. 4.1), given a target time scale $\Delta$. We follow the idea of soft actor-critic (SAC) with a constraint on policy entropy (Haarnoja et al., 2018) to derive the actor and critic objectives for off-policy RL. The spirit is to let the actor explicitly optimize the penalty term $\lambda(\delta_t - \Delta)$ through backprop, instead of treating the penalty term as a scalar non-differentiable signal. This is possible because the time scale, like the rest of the actions, is *differentiable* w.r.t. the policy parameters, *i.e.* $(a_t, \delta_t) \sim \pi(\cdot, \cdot \mid s_t)$.

The critic's objective is based on the following policy evaluation operator $\mathcal{L}^{\pi,\lambda}$ (similar to Eq. 5, but adding the penalty term into the Q value on next state):

$$\mathcal{L}^{\pi,\lambda} Q(s,a,\delta) := R(s,a,\delta) + \gamma^{\delta} \mathbb{E}_{s' \sim F, (a',\delta') \sim \pi}[Q(s',a',\delta') + \lambda(\delta' - \Delta)] \tag{11}$$

With the critic parameterized as $Q_\phi$ and the actor as $\pi_\theta$, we can define the **critic** objective:

$$\min_{\phi} \mathbb{E}_{(s,a,\delta)\sim\mathcal{D}} \left[ (Q_\phi(s,a,\delta) - \mathcal{L}^{\pi_\theta,\lambda} \widetilde{Q_\phi}(s,a,\delta))^2 \right] \tag{12}$$

where $\mathcal{D}$ is off-policy collection and $\widetilde{Q_\phi}$ is the target Q value function in TD.

The **actor** objective is defined as follows:

$$\max_{\theta} \mathbb{E}_{s\sim\mathcal{D},(a,\delta)\sim\pi_\theta(s)} \left[ Q_\phi(s,a,\delta) + \underbrace{\lambda(\delta - \Delta)}_{\text{depend on } \theta} \right] = \mathbb{E}[Q_\phi(s,a,\delta) + \lambda\delta] \tag{13}$$

where we use pathwise derivative to optimize $\theta$. We emphasize that the penalty term (the time scale $\delta$) depends on the actor parameter $\theta$.

Finally, the **Lagrange multiplier** $\lambda$'s objective is defined as

$$\min_{\lambda \geq 0} \mathbb{E}_{s\sim\mathcal{D},(\cdot,\delta)\sim\pi_\theta(s)}[\lambda(\delta - \Delta)] \tag{14}$$

where we use the truncated version of Eq. 10 similar to the policy entropy constraint in SAC.

## 5 Experiments

In this section, we aim to compare the effectiveness of **constrained CTC-MDP** (Def. 4.1), with two baselines: **unconstrained CTC-MDP** (Def. 3.2) and the standard **MDP**, through empirical results with deep RL algorithms trained on each environment.

We aim to answer two important questions:

1. **(Sec. 5.1) Performance of RL on constrained CTC-MDP over baseline MDPs.** How does the performance of a constrained CTC-MDP compare to a standard MDP, given the same (target) average time scale? How does RL on CTC-MDP compare to RL on an unconstrained CTC-MDP?
2. **(Sec. 5.2) Policy interpretability.** Can we learn a ***meaningful*** policy in constrained CTC-MDP? Intuitively, regions of state space where the dynamics are smooth are less sensitive to choice of time scale, while states with non-smooth dynamics (*e.g.* a leg making contact with the ground) are more sensitive to choice of time scale. We hypothesize that in such tasks, a learned policy can select a suitable time scale corresponding to current state.

We use the standard MuJoCo benchmark (Todorov et al., 2012) as our testbed. To enable continuous time control in MuJoCo, we augment the arguments in the `step(action)` function in the environment into `step(action, dt)`. In the uncontrolled setting (*i.e.* MDPs), `dt` is a constant; while in controlled setting (*i.e.* CTC-MDPs), `dt` comes from the policy output, and the policy can only modify the continuous dt, but leave the number of frame skips fixed. We add policy entropy bonus (Haarnoja et al., 2018) into rewards to encourage exploration. Please see the appendix for environment and training details.

### 5.1 Constrained CTC-MDP over Unconstrained One and MDP

In Fig. 2, we show the learning curves in three continuous control environments, namely, Ant, Hopper, and Walker in MuJoCo simulator. We set the constraint on episode interaction steps as the original value of the corresponding MDP. We find that policies trained within constrained CTC-MDP perform the best episode returns in Ant and Hopper, and perform comparably to MDP in Walker. Moreover, it has similar episode interaction steps with MDP, meeting the average time scale constraint. On the contrary, (unconstrained) CTC-MDP degrades to learn to take a very small decision time scale and cause very large episode steps in Ant, resulting in lower performance.

This shows that constrained CTC-MDPs can induce policies with both high episode returns and similar episode decision steps to those in MDPs. We also show the learning curves in Fig. 4 with x-axis being the total physical duration, with similar findings.

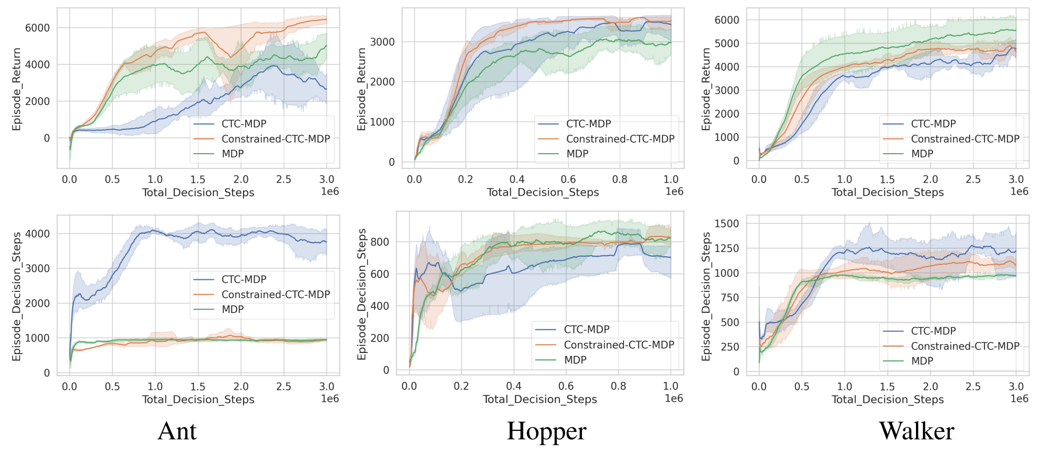

Figure 2: **Learning curves on constrained CTC-MDP, CTC-MDP, and MDP setting.** We show the results in three continuous control tasks Ant, Hopper, Walker, and report the **episode return** (top row), and **episode decision steps** (bottom row), which is proportional to episode interaction steps due to fixed frame skip.

### 5.2 Intepretability of Continuous Control on Time

To analyze how the policy controls the time, we first plot the time series through the episode decision steps of the learned time scale and physical states, in Hopper environment. Fig. 3 shows the time series plots of two physical quantities: the height of Hopper (`torso_z`; in meter) and binary signal of contacting the floor (`contact_floor`). We also show the scatter plots between height and corresponding learned time scales in Fig. 5.

From time series plots, when Hopper contacts the floor (with lower height), it will take a very large time scale; when it hops in the air (with higher height), it will take a very small time scale. Similarly, the scatter plots show that Hopper takes a large time scale when it accelerates to hop above the floor. We hypothesize that the policy achieves higher returns by learning to exploit the dynamics; as the contact dynamics are highly non-smooth, a large time scale may lead to implausible simulation which allows the agent to skip past some bad intermediate states.

## 6 Discussion

**Limitations.** Introducing an extra DoF on time scale $\mathcal{A}_\delta = [\delta_{\min}, \delta_{\max}]$ in the CTC-MDP framework affords the agent flexibility to achieve higher returns, but also has some caveats, mostly around hyperparameter selection of $\delta_{\min}$ and $\delta_{\max}$. (1) Too small $\delta_{\min}$ may cause difficulty in policy optimization (Tallec et al., 2019) and induce long episodes, if policy learns to adjust actions frequently. (2) Too large $\delta_{\max}$ are prone to large simulation errors, especially notorious for non-smooth dynamics such as contact modeling. The simulation becomes an inaccurate approximation of real world

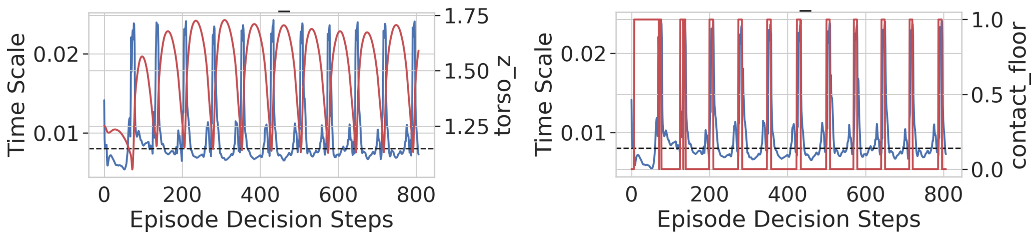

Figure 3: **Time series** between learned time scales (in blue) and the height of Hopper (in red; left figure) and the binary signal of contacting the floor (in red; right figure). The black horizontal line is the target time scale.

systems if the policy learns to exploit the dynamics (Sec. 5.2). That being said, in real world environments such as robotics, we do not need to worry about large time scales as there would not be simulation error. (3) A good choice of $[\delta_{\min}, \delta_{\max}]$ varies from dynamics to dynamics and from simulator to simulator, which requires some human effort to tune. (4) We also found it hard to learn the **discrete frame skip** together with the continuous time in our framework, due to the mixture of continuous and discrete actions introducing additional complexity to actor-critic algorithms. (5) Finally, we tried to generalize the constrained CTC-MDP $J^{\text{CTC-MDP}}(\pi, \lambda; \Delta)$ to a **contextual** one where the context is the target average time scale $\Delta$, but found some mixed results when comparing to a standard contextual MDP without time scale control. We suspect that solving a contextual CTC-MDP requires better algorithms for adaptive sampling context variables during training, which generally remains an open problem in multi-task RL.

**Future work.** (1) On the algorithmic side, we could investigate non-Gaussian distributions of the learned time scale actions, to see if they are easier to learn or more expressive. (2) On the framework side, we could extend constrained CTC-MDPs to contextual ones where an agent can condition on the target average timescale $\Delta$ to learn a multi-task policy that generalizes to unseen timescales. As mentioned before, this may depend on general innovations in online RL algorithms for contextual MDPs. (3) On the experimental side, we could conduct real robot experiments to evaluate our framework, given that real robots are exempt from simulation error. In a real robot system, the agent would select the physical time interval until its next action computation.

## Acknowledgement

TN thanks Haoxue Fan for actionable suggestions on human decision-making and correlation analysis.

## References

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphael Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, et al. What matters in on-policy reinforcement learning? a large-scale empirical study. *arXiv preprint arXiv:2006.05990*, 2020.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Leemon C Baird. Reinforcement learning in continuous time: Advantage updating. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 4, pp. 2448–2453. IEEE, 1994.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47: 253–279, 2013.

Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, 6(5): 679–684, 1957.

André Biedenkapp, Raghu Rajan, Frank Hutter, and Marius Lindauer. Temporl: Learning when to act. *arXiv preprint arXiv:2106.05262*, 2021.

Steven J Bradtke and Michael O Duff. Reinforcement learning methods for continuous-time markov decision problems. In *Advances in neural information processing systems*, pp. 393–400, 1995.

Alex Braylan, Mark Hollenbeck, Elliot Meyerson, and Risto Miikkulainen. Frame skip is a powerful parameter for learning to play atari. In *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Arthur E Bryson and Yu-Chi Ho. *Applied optimal control: optimization, estimation, and control*. Routledge, 2018.

Roland Bulirsch, Josef Stoer, and J Stoer. *Introduction to numerical analysis*. Springer, 1991.

Chen Chen, Hongyao Tang, Jianye Hao, Wulong Liu, and Zhaopeng Meng. Addressing action oscillations through learning policy inertia. *arXiv preprint arXiv:2103.02287*, 2021.

Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. In *Advances in neural information processing systems*, pp. 271–278, 1993.

Kenji Doya. Reinforcement learning in continuous time and space. *Neural computation*, 12(1): 219–245, 2000.

Jianzhun Du, Joseph Futoma, and Finale Doshi-Velez. Model-based reinforcement learning for semi-markov decision processes with neural odes. *arXiv preprint arXiv:2006.16210*, 2020.

Ishan P Durugkar, Clemens Rosenbaum, Stefan Dernbach, and Sridhar Mahadevan. Deep reinforcement learning with macro-actions. *arXiv preprint arXiv:1606.04615*, 2016.

Jake Grigsby, Jin Yong Yoo, and Yanjun Qi. Towards automatic actor-critic solutions to continuous control. *arXiv preprint arXiv:2106.08918*, 2021.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Jean Harb, Pierre-Luc Bacon, Martin Klissarov, and Doina Precup. When waiting is not an option: Learning options with a deliberation cost. *arXiv preprint arXiv:1709.04571*, 2017.

Matteo Hessel, Hado van Hasselt, Joseph Modayil, and David Silver. On inductive biases in deep reinforcement learning. *arXiv preprint arXiv:1907.02908*, 2019.

Ronald A Howard. *Dynamic probabilistic systems: Markov models*, volume 1. Courier Corporation, 2012.

Shivaram Kalyanakrishnan, Siddharth Aravindan, Vishwajeet Bagdawat, Varun Bhatt, Harshith Goka, Archit Gupta, Kalpesh Krishna, and Vihari Piratla. An analysis of frame-skipping in reinforcement learning. *arXiv preprint arXiv:2102.03718*, 2021.

Stuart T Klapp. Motor response programming during simple choice reaction time: The role of practice. *Journal of Experimental Psychology: Human perception and performance*, 21(5):1015, 1995.

Stuart T Klapp. Reaction time analysis of two types of motor preparation for speech articulation: Action as a sequence of chunks. *Journal of motor behavior*, 35(2):135–150, 2003.

Aravind S Lakshminarayanan, Sahil Sharma, and Balaraman Ravindran. Dynamic action repetition for deep reinforcement learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pp. 2133–2139, 2017.

Jongmin Lee, Byung-Jun Lee, and Kee-Eung Kim. Reinforcement learning for control with multiple frequencies. In *Thirty-fourth Conference on Neural Information Processing Systems (NeurIPS 2020)*. Neural information processing systems foundation, 2020.

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.

Amy McGovern, Richard S Sutton, and Andrew H Fagg. Roles of macro-actions in accelerating reinforcement learning. In *Grace Hopper celebration of women in computing*, volume 1317, pp. 15, 1997.

Alberto Maria Metelli, Flavio Mazzolini, Lorenzo Bisi, Luca Sabbioni, and Marcello Restelli. Control frequency adaptation via action persistence in batch reinforcement learning. *arXiv preprint arXiv:2002.06836*, 2020.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Jean Jacques Moreau and Panagiotis D Panagiotopoulos. *Nonsmooth mechanics and applications*, volume 302. Springer, 2014.

Rémi Munos and Paul Bourgine. Reinforcement learning for continuous stochastic control problems. In *NIPS*, pp. 1029–1035, 1997.

Simon Ramstedt and Christopher Pal. Real-time reinforcement learning. *arXiv preprint arXiv:1911.04448*, 2019.

Daniele Reda, Tianxin Tao, and Michiel van de Panne. Learning to locomote: Understanding how environment design matters for deep reinforcement learning. *arXiv preprint arXiv:2010.04304*, 2020.

Sheldon M Ross. *Applied probability models with optimization applications*. Courier Corporation, 2013.

Katrin Schneider, Grzegorz Dogil, and Bernd Möbius. Reaction time and decision difficulty in the perception of intonation. In *Twelfth Annual Conference of the International Speech Communication Association*. Citeseer, 2011.

Sahil Sharma, Aravind S Lakshminarayanan, and Balaraman Ravindran. Learning to repeat: Fine grained action repetition for deep reinforcement learning. *arXiv preprint arXiv:1702.06054*, 2017.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. 2018.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Corentin Tallec, Léonard Blier, and Yann Ollivier. Making deep q-learning methods robust to time discretization. *arXiv preprint arXiv:1901.09732*, 2019.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

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.

Alexander Vezhnevets, Volodymyr Mnih, Simon Osindero, Alex Graves, Oriol Vinyals, John Agapiou, et al. Strategic attentive writer for learning macro-actions. In *Advances in neural information processing systems*, pp. 3486–3494, 2016.

Ted Xiao, Eric Jang, Dmitry Kalashnikov, Sergey Levine, Julian Ibarz, Karol Hausman, and Alexander Herzog. Thinking while moving: Deep reinforcement learning with concurrent control. *arXiv preprint arXiv:2004.06089*, 2020.

Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. *arXiv preprint arXiv:1910.01741*, 2019.

Haonan Yu, Wei Xu, and Haichao Zhang. Taac: Temporally abstract actor-critic for continuous control. *Advances in Neural Information Processing Systems*, 34, 2021.

## A Environment Details

We adopt the Hopper, Walker, and Ant from standard MuJoCo benchmark[3], use their default parameters such as frame skips and episode duration, and introduce continuous time scale $\mathcal{A}_\delta$. The bounds of $\mathcal{A}_\delta$ are roughly tuned per environment to avoid numerical issues in simulation. Table 3 shows the details of environment parameters.

| | Hopper | Walker | Ant |
|---|---|---|---|
| Episode duration $H$ (secs) | 8 | 8 | 50 |
| Original fixed time scale $\delta_{\text{ori}}$ (secs) | 0.002 | 0.002 | 0.01 |
| Minimal time scale $\delta_{\text{min}}$ (secs) | 0.00064 | 0.00064 | 0.002 |
| Maximal time scale $\delta_{\text{max}}$ (secs) | 0.0064 | 0.0064 | 0.02 |
| Frame skip $n$ | 4 | 4 | 5 |
| Target average time scale $\Delta$ (secs) | 0.008 | 0.008 | 0.1 |

Table 3: **Environment setting.** We keep the default frame skip $n$, and set the target average time scale as $\Delta = \delta_{\text{ori}} * n$ to match the default setting.

To match the default MDP's reward function $r_{\text{ori}}$ when $\delta = \delta_{\text{ori}}$, we define the CTC-MDP's reward function $R$ as:

$$R(s, a, \delta) := r_{\text{ori}}(s, a)\frac{\delta}{\delta_{\text{ori}}}$$

so that we can directly compare the returns between MDPs and CTC-MDPs.

## B Training Details

We use SAC with policy entropy constraint (Haarnoja et al., 2018) to train RL for continuous control. For MDPs, we use the default hyperparameters of SAC. For (constrained) CTC-MDPs, we find SAC is very sensitive to the value of target policy entropy, so we tune this hyperparameter for each environment within $\{-3, -4, -5, -6, -7, -8\}$ and find $-5$ works well for all the environments, which is finally taken in the plots in the main paper.

## C Additional Figures

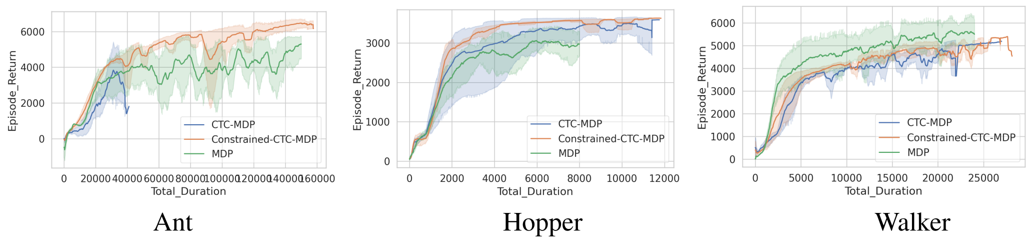

Figure 4: Learning curves on constrained CTC-MDP, CTC-MDP, and MDP setting. We change the x-axis from total decision steps (in Fig. 2) to **total physical duration** (in secs). We find similar pattern: constrained CTC-MDP can achieve comparable performance as MDP with same physical duration, and is faster and more stable than unconstrained CTC-MDP.

[3]https://www.gymlibrary.ml/pages/environments/mujoco/

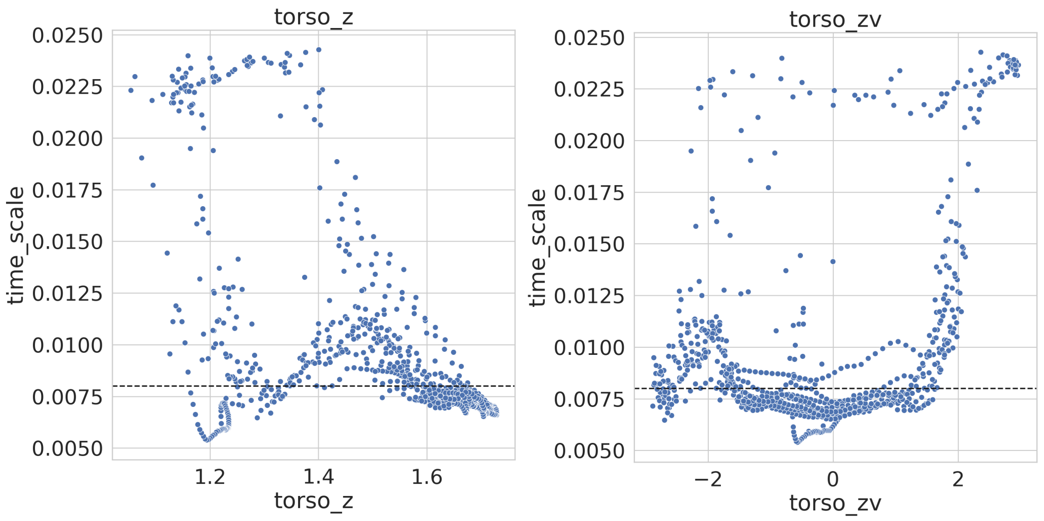

Figure 5: **Scatter plots** between the height of Hopper (`torso_z`; left figure) or the height velocity (`torso_zv`; right figure) in x-axis, and corresponding learned time scales in y-axis. The black horizontal line is the target time scale.