Time Discretization in Reinforcement Learning

Armin Ashrafi*
Department of Computing Science
University of Alberta; Amii
Edmonton, Canada
armin.ashrafi@ualberta.ca

Masoud Jafaripour*
Department of Computing Science
University of Alberta; Amii
Edmonton, Canada
jafaripo@ualberta.ca

Sarah Amini*
Department of Computing Science
University of Alberta; Amii
Edmonton, Canada
samini1@ualberta.ca

Abstract

In reinforcement learning (RL), the ongoing advancement of algorithms and agents for improved performance remains highly sensitive to hyperparameter choices, implementation nuances, and minor environmental changes. One such overlooked parameter is the time discretization factor δt, which governs the time step in environment simulation. It is usually implicitly assumed that we don't need to tune this value. However, theoretically, different values of δt introduce a trade-off between computational efficiency and the level of fine control that the agent has over the environment, each offering distinct advantages and disadvantages. This study investigates the importance of δt in RL and examines how its variation affects agent performance across algorithms and environments. Using PPO and A2C algorithms on CartPole and Acrobot environments, we present sensitivity analyses that reveal the influence of δt on performance. Our results suggest that default δt values in standard environments may not yield optimal performance for all algorithms and can introduce bias favoring specific methods. This highlights the need for thoughtful consideration of δt in RL experimentation to ensure fair and robust evaluations.

1 Introduction

Reinforcement learning (RL) research traditionally assumes a discrete-time Markov Decision Process (MDP) when modeling different tasks for learning. However, the transformation from a continuous-time domain to a discrete one is non-trivial and can profoundly influence both the theoretical framing and practical outcomes of RL algorithms. The choice of time discretization—specifically, the time step δt —is often treated as a mere technical detail, but evidence suggests ([Tallec et al., 2019], [Asis and Sutton, 2024], [Karimi et al., 2023]) it is, in fact, a hidden hyperparameter with potentially unintended consequences.

The granularity of δt determines the agent-environment interaction cycles, affecting computational cost, learning efficiency, and performance. A large δt results in sparse interactions, reducing computational overhead but potentially missing critical states. On the other hand, a small δt increases the precision of control and the density of agent interactions but at the cost of significantly higher computation time. Different strategies for approximating time-dependent variables such as reward

*Equal contribution, authors are listed in alphabetical order.

signals, discount factors, and state transitions can further amplify these trade-offs [Tallec et al., 2019].

In many benchmark environments used by the RL community, a default δt is implicitly used as part of the problem specification. Yet, RL algorithms are not inherently time-invariant [Tallec et al., 2019] , [Park et al., 2022]. A fixed δt could, for instance, favor one class of algorithms over another, shaping the results and claims made in state-of-the-art studies. If such choices are not scrutinized, reported improvements may be misleading. If the default choice of δt systematically advantages an algorithm or method over the other, not paying attention to this parameter could undermine any analysis made between algorithms. This is a critical step towards fair, rigorous, and transparent RL research.

This project aims to systematically investigate the role of time discretization in RL. We will examine whether the commonly accepted default values for the δt disproportionately benefit particular learning algorithms, and if so, whether adjusting δt could better reveal underlying algorithmic properties. Ultimately, this inquiry has practical implications for the design of fair benchmarks, the interpretation of published results, and the pursuit of reliable empirical practice in reinforcement learning.

1.1 Related Work

Time discretization in reinforcement learning (RL) significantly influences algorithm performance, as demonstrated by various studies.Tallec et al. [2019] et al. highlighted the failure of Q-learning methods as δt approaches zero, proposing an off-policy algorithm robust across time discretizations.Asis and Sutton [2024] explored inconsistencies in applying discrete-time algorithms to continuous-time environments, emphasizing the importance of aligning return definitions. Karimi et al. [2023] introduced the Continuous-Time Continuous-Options (CTCO) framework, which allows agents to operate at variable decision frequencies, mitigating the effects of fixed discretization. Additionally, temporally extended actions and options, along with action repetition methods, have been proposed to address sensitivity to δt. Collectively, these works underscore the critical role of time discretization as a hidden hyperparameter in RL, influencing both theoretical consistency and practical performance [Tallec et al., 2019], [Asis and Sutton, 2024].

2 Methodology

The reinforcement learning problem is usually framed as a *Markov Decision Process* (MDP), where the tuple $(\mathcal{S}, \mathcal{A}, T, r, \gamma)$ is used to model the interactions between the agent and the environment. To modify this formulation for our purposes, we need to adapt the formulas to include δt as the time discretization. When defining an MDP, it is essential to account for the influence of time discretization (δt) on the transition function (T), reward (r), and discount factor (γ)[Tallec et al., 2019]. Thus, the MDP tuple can be redefined as:

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, T_{\delta t}, r_{\delta t}, \gamma_{\delta t})$$

We assume that time is discretized into regular intervals of some length δt, and that each time step for the agent occurs at these increments, where it receives the state and the observations from the environment. We also need to reformulate the reward and other time dependencies based on δt for a given trajectory τ:

$$R_{\delta t}(\tau) := \sum_{k=0}^{\infty} (\gamma^{\delta t})^k \, (\delta t \cdot r_t)(s_{k\delta t}, a_{k\delta t}) = \sum_{k=0}^{\infty} (\gamma_{\delta t})^k \, r_{\delta t}(s_{k\delta t}, a_{k\delta t})$$

This demonstrates that the rewards and the discount factor's exponent should be scaled by δt, making comparisons fair by ensuring consistent reward accumulation and discounting. It is important to note that modifying the number of interactions with the environment also changes the number of weight updates. Consequently, the learning rate becomes a critical hyperparameter in this study.

To fairly compare the performance of two agents with different δt values, the total environmental time during training must be kept the same for both. As a result, agents with smaller δt values will

have more interactions and samples than those with larger δt values. The performance of agents is compared by evaluating the Area Under the Curve (AUC) for episodic returns over the environmental time. Finally, we used a relative $\bar{\delta t} = \delta t / \delta t_{default}$ value, instead of an absolute one, to simplify comparison across environments with differing default δt values.

3 Experiments

In this study, we conducted experiments using Proximal Policy Optimization (PPO) [Schulman et al., 2017] and Advantage Actor-Critic (A2C) [Mnih et al., 2016], in Cartpole and Acrobot [Towers et al., 2024] with the following fixed parameters: a discount factor $\gamma = 1$, a batch size of 16, and a Multi-Layer Perceptron (MLP) architecture comprising three fully connected layers with 512, 256, and 64 units respectively. The learning rate was systematically tuned from among the values $[10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}]$. Each experiment was run for 200,000 time steps, divided by $bar\delta t$ to ensure a consistent amount of physical time elapsed across different configurations.

For hyperparameter tuning, we evaluated 10 random seeds for each setting and selected the configuration yielding the best AUC. To assess the performance of this hyperparameter configuration for our final results, we ran this configuration on 30 additional seeds. Since smaller δt values result in more samples than larger ones, we devised a consistent approach for plotting results. Specifically, we calculated the environmental time, aligning data points with the actual progression of time. To ensure legibility of the plots, values between samples were imputed using the previous sample value.

4 Results and Discussion

We analyzed the sensitivity of the AUC of the episodic return for each configuration, along with their confidence intervals, using 10000 bootstrap samples. The resulting sensitivity curves reveal several key patterns.

First, Figure 1 demonstrates a general downward trend in performance as δt increases. This outcome is expected, as a larger δt reduces the agent's ability to control its environment by limiting the number of decision-making opportunities.

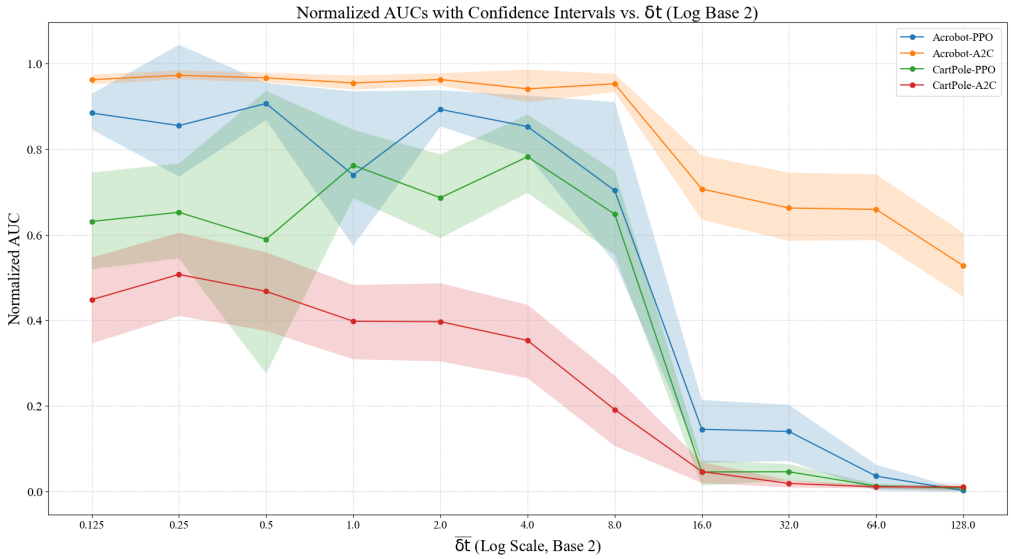

Figure 1: Sensitivity Curve of different algorithms with 30 seeds, with bootstrap confidence intervals.

Interestingly, PPO exhibits a notable performance collapse between δt values of 8 and 16. In Acrobot, however, A2C maintains stable performance even at much higher δt values, such as 128. The inclusion of a random baseline would be necessary to contextualize the significance of this behavior.

The sensitivity curve also shows a sharp drop at 16 times the default δt, where performance drops dramatically. In contrast, for δt values between 1 and 8, as supported by the learning curves, the performance declines are not statistically significant.

This suggests that, at least in the environments tested, some algorithms could operate with lower-frequency interactions without substantial performance degradation, potentially saving computational resources. These results underscore the importance of understanding algorithm sensitivity to δt and highlight promising avenues for balancing computational efficiency and performance in reinforcement learning. For more detailed analysis, see Appendix A.

5 Limitations and Future Work

Our results were limited by the range of δts tested, constrained by computational feasibility, which excluded extreme values. Currently, we use relative δt values to compare environments, but this approach complicates generalization due to varying default values. Future work could investigate using absolute δt values across a standardized range for all environments, enabling consistent evaluation of algorithms. Addressing these limitations would make our results more reliable and provide a broader perspective on algorithm performance.

It is also worth exploring whether computational gains can be achieved by training with larger δt values and deploying the resulting policies at smaller δt values. While our current environments have clear maximum and minimum performance thresholds that inform our conclusions, running random baselines could provide additional insights by establishing how a random agent would perform. This would help quantify the extent of performance degradation. Additionally, incorporating tolerance intervals around the sensitivity curves could offer a measure of each algorithm's stability across different δt values.

In future studies, we could explore alternative framings of the problem, such as disentangling δt for the environment and the agent to investigate how differing time scales affect interactions. Adaptive heuristics like linear scaling of the learning rate could reduce the reliance on manual hyperparameter tuning. Additionally, testing algorithm robustness in environments with dynamically varying δt, inspired by [Karimi et al., 2023], could provide insights into their flexibility.

6 Conclusion

In this project, we explored the effects of different time discretizations on the performance of various RL algorithms across multiple environments, while linearly scaling time-dependent variables in the underlying MDP. Through a sensitivity analysis, we uncovered several key insights: algorithms can be advantaged or disadvantaged in a non-trivial manner by the default value of δt in different environments. The analysis further revealed that performance drops are influenced not only by the environment but also by the learning algorithm, emphasizing the importance of carefully examining this hyperparameter when conducting RL experiments. Additionally, we identified an opportunity to reduce computational costs by training with lower-frequency δt values. However, our approach and time constraints introduced limitations that warrant further investigation to fully understand its impact on performance and learning dynamics.

References

Kris De Asis and Richard S. Sutton. An idiosyncrasy of time-discretization in reinforcement learning, 2024. URL `https://arxiv.org/abs/2406.14951`.

Amirmohammad Karimi, Jun Jin, Jun Luo, A Rupam Mahmood, Martin Jagersand, and Samuele Tosatto. Dynamic decision frequency with continuous options. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7545–7552. IEEE, 2023.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning, 2016. URL `https://arxiv.org/abs/1602.01783`.

Seohong Park, Jaekyeom Kim, and Gunhee Kim. Time discretization-invariant safe action repetition for policy gradient methods, 2022. URL `https://arxiv.org/abs/2111.03941`.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Corentin Tallec, Léonard Blier, and Yann Ollivier. Making deep q-learning methods robust to time discretization. In *International Conference on Machine Learning*, pages 6096–6104. PMLR, 2019.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A standard interface for reinforcement learning environments, 2024. URL https://arxiv.org/abs/2407.17032.

A Learning Curves

In this section, we present the learning curves for four configurations: PPO on CartPole, PPO on Acrobot, A2C on CartPole, and A2C on Acrobot. Each curve illustrates the average performance computed over 30 random seeds, using a sliding average window of 1000. The results are reported for 11 relative values of δt: $[0.125, 0.25, 0.5, 1, 2, 4, 8, 16, 32, 64, 128]$.

The y-axis represents the episodic returns, while the environmental time is kept consistent across all experiments by scaling the number of timesteps proportionally to δt. To improve interpretability, the x-axis displays integer multiples of the default δt for each environment, ensuring that comparisons remain fair and environmental time remains uniform across the experiments. Below are the results for the different settings.

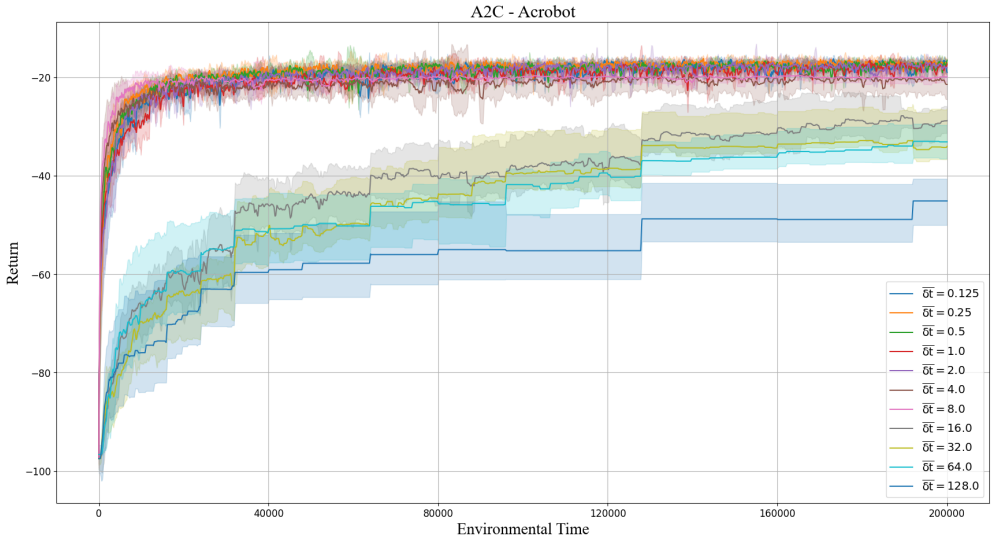

Figure 2: A2C Acrobot

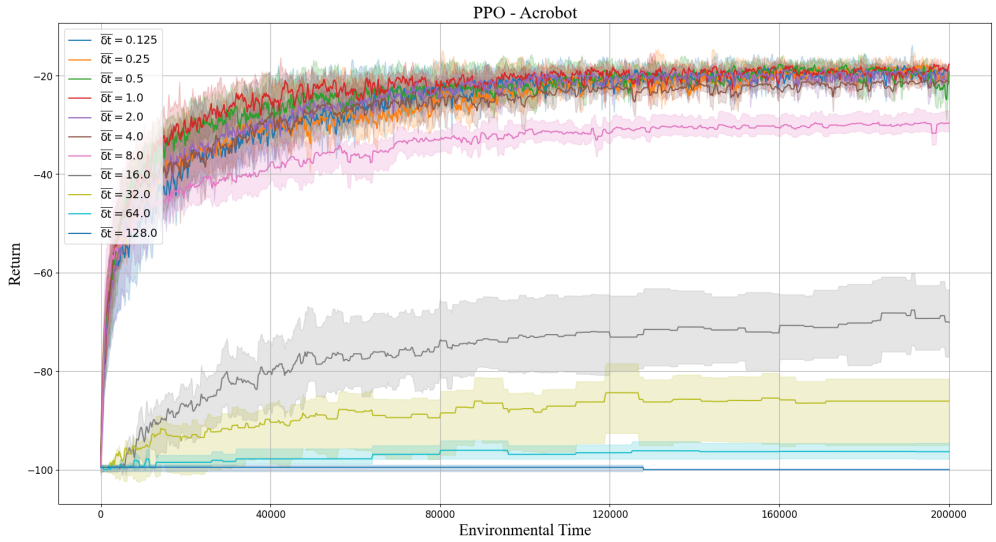

Figure 3: PPO Acrobot

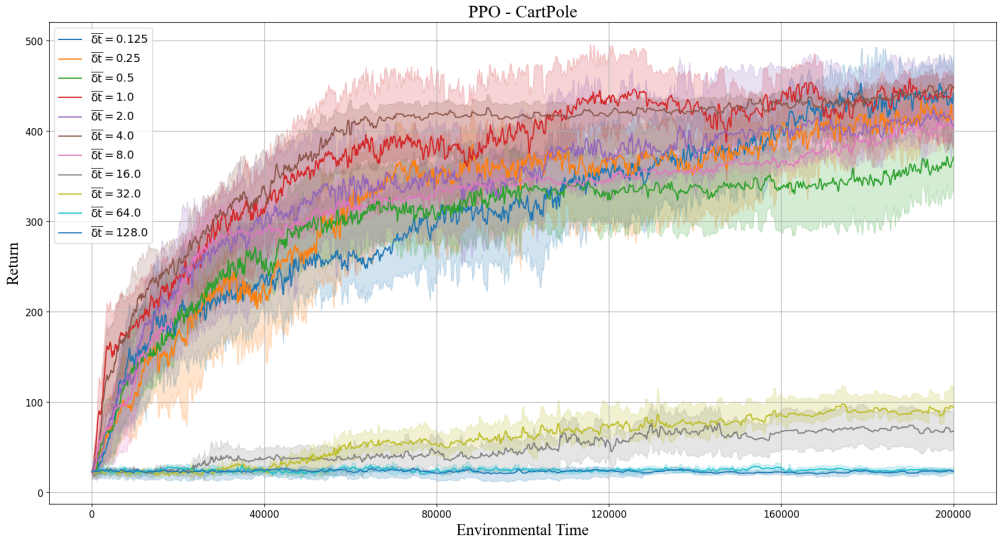

Figure 4: PPO Cartpole

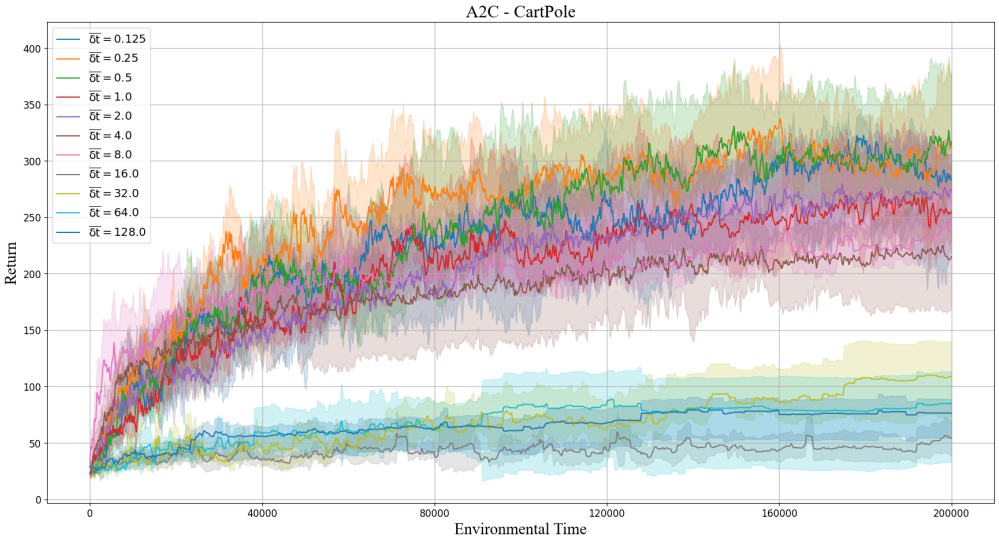

Figure 5: A2C CartPole

B Best Learning Rate

As detailed in the report, we tuned the learning rate by evaluating the AUC for six different learning rate values: $[0.125, 0.25, 0.5, 1, 2, 4, 8, 16, 32, 64, 128]$, averaged over 10 random seeds. The table below summarizes the best-performing learning rate for each setting.

Table 1: Best learning rate in each experimental setting

Algorithm	**Environment**	**Relative δt**	**Best Learning Rate**
PPO	CartPole	2^{-3}	0.0001
		2^{-2}	0.0001
		2^{-1}	0.00001
		2^{0}	0.0001
		2^{1}	0.0001
		2^{2}	0.0001
		2^{3}	0.0001
		2^{4}	0.001
		2^{5}	0.001
		2^{6}	0.001
		2^{7}	0.00001
	Acrobot	2^{-3}	0.0001
		2^{-2}	0.00001
		2^{-1}	0.0001
		2^{0}	0.00001
		2^{1}	0.0001
		2^{2}	0.00001
		2^{3}	0.00001
		2^{4}	0.00001
		2^{5}	0.0001
		2^{6}	0.0001
		2^{7}	0.00001
A2C	CartPole	2^{-3}	0.0001
		2^{-2}	0.0001
		2^{-1}	0.0001
		2^{0}	0.0001
		2^{1}	0.0001
		2^{2}	0.0001
		2^{3}	0.001
		2^{4}	0.001
		2^{5}	0.001
		2^{6}	0.001
		2^{7}	0.001
	Acrobot	2^{-3}	0.0001
		2^{-2}	0.0001
		2^{-1}	0.0001
		2^{0}	0.0001
		2^{1}	0.0001
		2^{2}	0.0001
		2^{3}	0.0001
		2^{4}	0.0001
		2^{5}	0.0001
		2^{6}	0.0001
		2^{7}	0.0001