Imperial College London

Time Limits in Reinforcement Learning

Fabio Pardo, Arash Tavakoli, Vitaly Levdik & Petar Kormushev

1. Introduction

In reinforcement learning it is common to let an agent interact with its environment for a **fixed amount of time** before resetting the environment and repeating the process in a series of episodes.

The task can either be to maximize the performance over

- that fixed period → **time-limited task**

$$G^{\gamma}_{t:T} \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots + \gamma^{T-t-1} R_T = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1} \quad (1)$$

- an infinite period (unless environmental termination) while time limits are used during training to diversify experience → **time-unlimited task**

$$G^{\gamma}_{t} \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad (2)$$

2. Standard approach

In the **standard** benchmark domains (e.g. OpenAI Gym) or algorithms (e.g. OpenAI Baselines), this distinction is often overlooked.

- The state-value function is defined as:

$$v_\pi(s) \doteq \mathbb{E}_\pi \left[G^{\gamma}_{t} | S_t = s \right] \quad (3)$$

- The temporal-difference targets to approximate $v_\pi(s)$ are:

$$\begin{cases} r & \text{at termination (including time limits)} \\ r + \gamma \hat{v}_\pi(s) & \text{otherwise} \end{cases} \quad (4)$$

Issues:

- Terminations due to time limits are perceived by the agent as environmental terminations that **dynamically change** with the agent's policy.
- The unperceived remaining time leads to **state aliasing**.

3. Proposed approaches

Time-awareness for time-limited tasks:
Include a notion of the **remaining time in input**.

- The state-value function is defined as:

$$v_\pi(s, T-t) \doteq \mathbb{E}_\pi \left[G^{\gamma}_{t:T} | S_t = s \right] \quad (5)$$

- The temporal-difference targets for $v_\pi(s, T-t)$ are:

$$\begin{cases} r & \text{at termination (including time limits)} \\ r + \gamma \hat{v}_\pi(s', T-t-1) & \text{otherwise} \end{cases} \quad (6)$$

Partial-episode bootstrapping for time-unlimited tasks:
Bootstrap when termination is due to time limits.

- The state-value function is defined as:

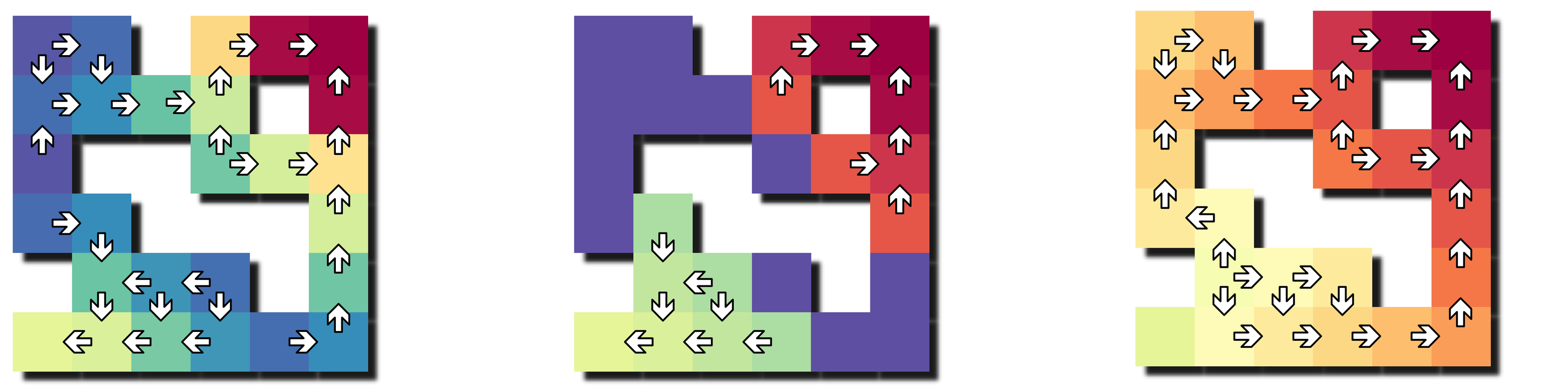
$$v_\pi(s) \doteq \mathbb{E}_\pi \left[G^{\gamma}_{t:T} + \gamma^{T-t} v_\pi(S_T) | S_t = s \right] \quad (7)$$

- The temporal-difference targets for $\hat{v}_\pi(s)$ are:

$$\begin{cases} r & \text{at **environmental** termination} \\ r + \gamma \hat{v}_\pi(s') & \text{otherwise **(including time limits)**} \end{cases} \quad (8)$$

4. A motivational example

The Two-Goal Gridworld problem: Two rewarding terminal states (50 top-right and 20 bottom-left), a penalty of -1 for moving, a time limit $T = 3$.

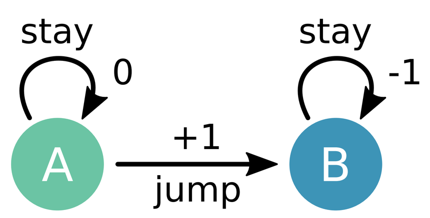


Standard — Time-aware — Partial-ep. bootstrap

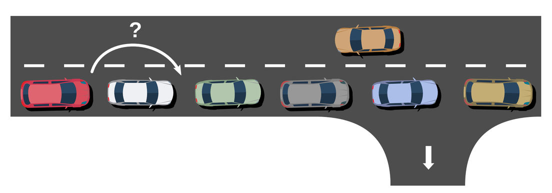
Policies and color-coded state-values. The **standard** agent optimizes for none of the two tasks. The agent with **time-awareness**, that learns to stay in place when there is not enough time to reach a goal, optimizes for the time-limited task. The agent with **partial-episode bootstrapping**, that aims for the highest reward, optimizes for the time-unlimited task.

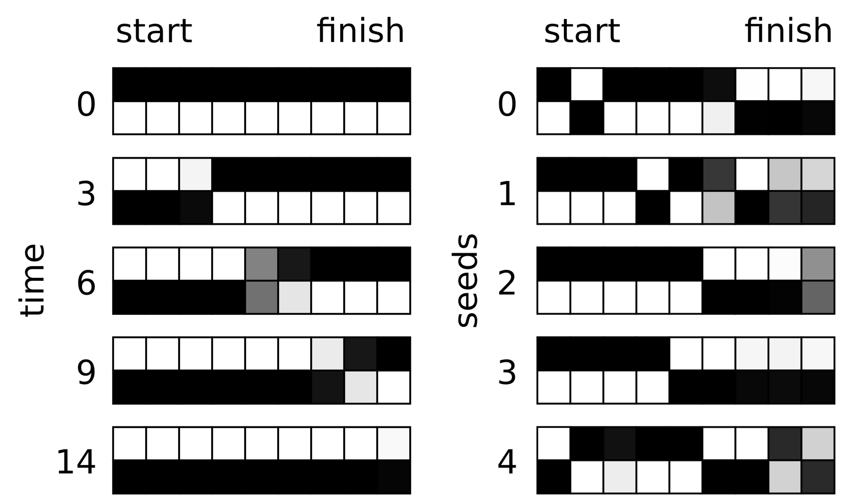
5. Time-awareness

The Last Moment problem: The agent has to jump just before the time limit to maximize its score. Only a **time-aware** agent can learn the optimal policy for $T > 1$.



The Queue of Cars problem: The agent has to reach the exit before the time limit. The "safe" action moves the car with 50% probability, the "dangerous" action does it with 80% probability but with 10% chance of collision. A reward of 1 is provided at destination.

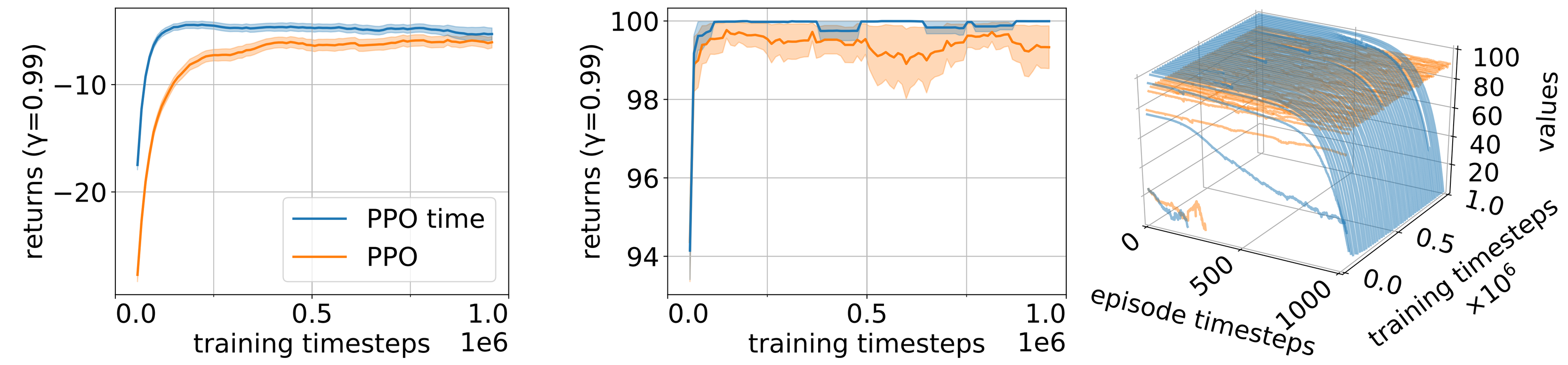




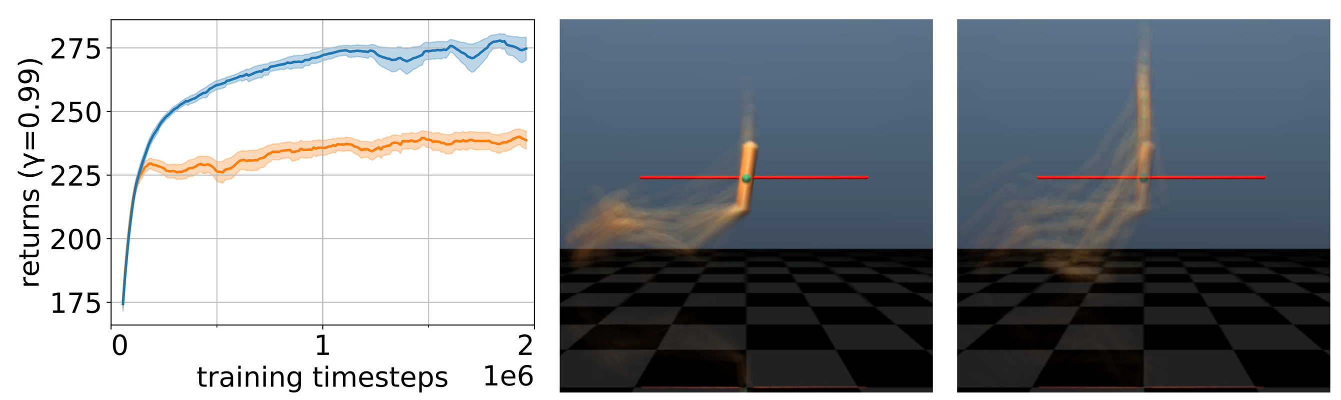
Time-aware PPO — Standard PPO

Decay ($\gamma = 0.99$)

Reacher-v1 $T = 50$ — **InvertedPendulum-v1 $T = 1000$**



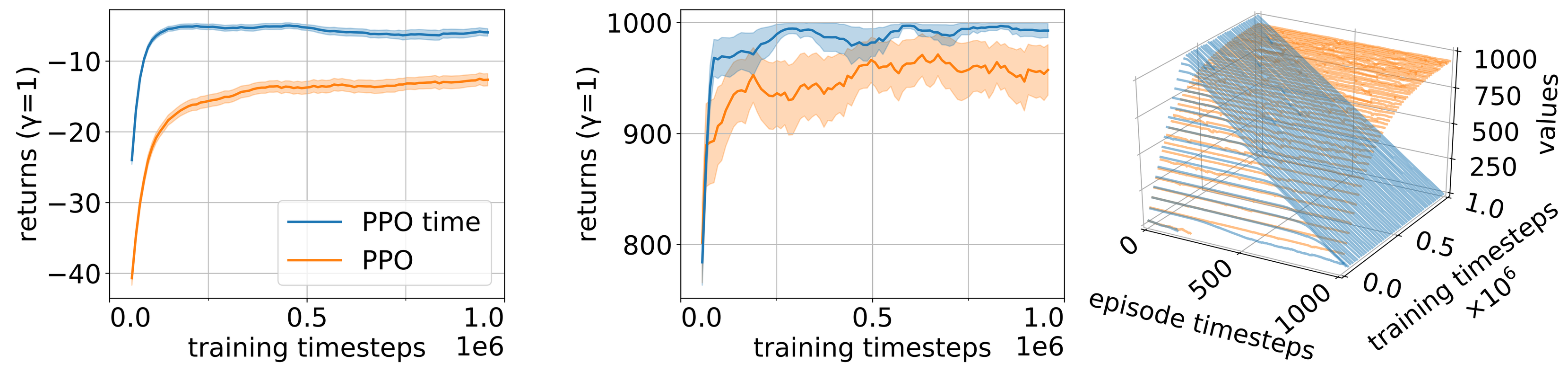
Hopper-v1 $T = 300$



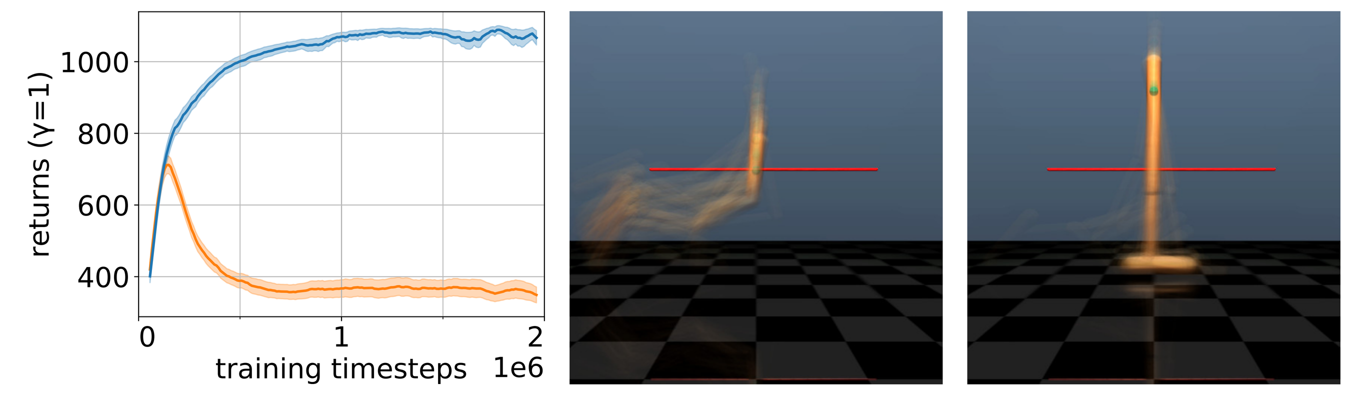
Time-aware PPO — Standard PPO

No decay ($\gamma = 1$)

Reacher-v1 $T = 50$ — **InvertedPendulum-v1 $T = 1000$**



Hopper-v1 $T = 300$



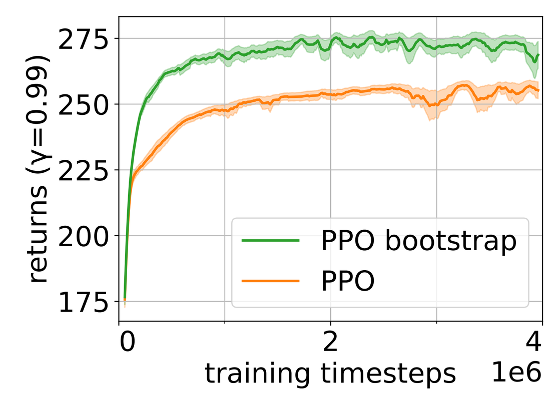
Time-aware PPO — Standard PPO

Implementation details for the proposed agents:

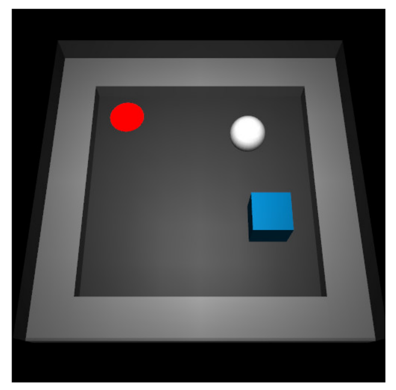
- Two-Goal Gridworld problem: three Q-learning tables were individually learned, one for each time step.
- Other experiments: a scalar representing the normalized remaining time was concatenated to the observations.

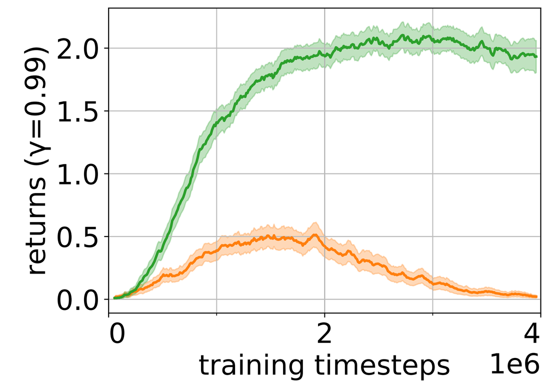
6. Partial-episode bootstrapping

Hopper-v1: Trained with a small time limit ($T = 200$, i.e. 2–3 hops) and tested with a very large time limit ($T = 10^6$, i.e. more than 2 hours of rendered hopping). With **partial-episode bootstrapping**, PPO manages to reach the evaluation time limit multiple times.



The Infinite Cube Pusher task: The agent has to push the cube onto the target with the ball. The target moves to a new position and a reward of 1 is given every time. Trained with a small time limit ($T = 50$, i.e. 1 push) and tested with a larger time limit ($T = 1000$). With **partial-episode bootstrapping**, PPO manages to reach 36 targets.





Implementation details for the proposed agent:

- The GAE(λ) of PPO was modified to keep bootstrapping when termination was only due to time limits.

Information

Article and videos are available at: sites.google.com/view/time-limits-in-rl
Contact: f.pardo@imperial.ac.uk
Research presented in this poster has been supported by Dyson Technology Ltd.