Bellman's Principle of Optimality in Deep Reinforcement Learning for Time-Varying Tasks

Alessandro Giuseppi[a] and Antonio Pietrabissa[a]

[a]Department of Computer, Control, and Management Engineering Antonio Ruberti at the University of Rome "La Sapeinza", via Ariosto, 25, 00185 Rome, Italy

ARTICLE HISTORY
Compiled May 5, 2021

ABSTRACT
This paper presents the first framework (up to the authors' knowledge) to address time-varying objectives in finite-horizon Deep Reinforcement Learning (DeepRL), based on a switching control solution developed on the ground of Bellman's principle of optimality. By augmenting the state space of the system with information on its visit time, the DeepRL agent is able to solve problems in which its task dynamically changes within the same episode. To address the scalability problems caused by the state space augmentation, we propose a procedure to partition the episode length to define separate sub-problems that are then solved by specialised DeepRL agents. Contrary to standard solutions, with the proposed approach the DeepRL agents correctly estimate the value function at each time-step and are hence able to solve time-varying tasks. Numerical simulations validate the approach in a classic RL environment.

Keywords:Bellman's Principle, Finite-Horizon Optimal Control, Deep Reinforcement learning, Model-Free Control

1. Introduction

Deep Reinforcement Learning (DeepRL) attracted, over the past few years, the focus of the scientific community regarding model-free control systems, thanks to its capability of controlling complex environments without requiring sensory data (e.g., visual information) pre-analysis and for its demonstrated better scalability when compared to traditional Reinforcement Learning (RL). In the DeepRL framework, the controller, or *agent*, is used to maximise a given objective or performance index, without any knowledge on the internal functioning or dynamics of the controlled system. To do that, the agent is said to *learn* how to control the system by *experiencing* its input-output pairs.

One of the most impactful works in the DeepRL literature is the well-known "Deep Q-Network" (DQN) algorithm, proposed in (Mnih et al., 2013) and later evolved into "Double DQN" (DDQN) in (Van Hasselt, Guez, & Silver, 2016) to improve the stability of its training and the quality of its performance estimations. Both DQN and DDQN are designed to control systems with discrete, or quantized, control actions, but DeepRL was tailored also to solve continuous control tasks, notably with the "Deep Deterministic Policy Gradient" (DDPG) algorithm proposed in (Lillicrap et al., 2015),

CONTACT Alessandro Giuseppi. Email: giuseppi@diag.uniroma1.it

which is among the most popular implementations for DeepRL controllers. Current state-of-the-art solutions include the "Twin-Delayed Deep Deterministic policy gradient" (TD3) (Fujimoto, Hoof, & Meger, 2018) algorithm, that improved DDPG by addressing its overestimation bias (like DDQN did for DQN9), and Soft Actor-Critic (SAC) (Haarnoja, Zhou, Abbeel, & Levine, 2018), a sample-efficient solution that demonstrated robust convergence properties. Nowadays, a significant research effort is being spent to find more efficient algorithms with improved control performances. The present paper, instead, aims at extending the class of control problems that DeepRL controllers can solve by proposing a framework in which state-of-the-art solutions, such as TD3, may be employed to deal with problems with time-varying control objectives.

1.1. Related Works and Main Contributions

Standard RL and DeepRL solutions (Barto, Sutton, & Anderson, 1983; Fujimoto et al., 2018; Kaelbling, Littman, & Moore, 1996; Lillicrap et al., 2015; Mnih et al., 2013) are typically concerned with the optimisation of a single objective. Nevertheless, over the years several studies dealt with multi-task and multi-objective problems (C. Liu, Xu, & Hu, 2015), whereas few studies considered scenarios in which the objective of the controller dynamically changes during the episode, as this paper does.

In the literature, several works (Borsa, Graepel, & Shawe-Taylor, 2016; Kelly & Heywood, 2018; Omidshafiei, Pazis, Amato, How, & Vian, 2017; Tanaka & Yamamura, 2003; Teh et al., 2017; Verme, da Silva, & Baldassarre, 2020) considered scenarios in which the trained controller is able to seamlessly cope with different tasks, but not concurrently. The authors of (Borsa et al., 2016) designed a framework in which a learning agent is able to attain optimal control strategies for various tasks that share the same environment. The proposed algorithm relies on two classic Dynamic Programming (DP) algorithms, i.e., Value and Policy Iteration. We mention that, even if DP algorithms are particularly subject to the so-called "curse of dimensionality", that exponentially relates their convergence times with the problem complexity, several approximated solutions have been proposed over the years to speed-up their solving times. Such solutions space from functional approximation (Xu, Lian, Zuo, & He, 2014) to data-driven ones (Chun, Lee, Park, & Choi, 2017; Zhang, Cui, Zhang, & Luo, 2011), finding significant applications in domains such as resource allocation (Forootani, Liuzza, Tipaldi, & Glielmo, 2019; Forootani, Tipaldi, Zarch, Liuzza, & Glielmo, 2019), industrial control (Wang, Chakrabarty, Zhou, & Zhang, 2019) and game theory (Zhang, Wei, & Liu, 2011).

In a similar setting, a solution in which the agent is able to exploit past knowledge to address new objectives was proposed originally in (Tanaka & Yamamura, 2003), where a controller able to continuously adapt to changing tasks was presented. As shown for the multi-agent setting in (Omidshafiei et al., 2017), the explicit knowledge on the on-going task is not in principle required, and a unified control logic that performs well across various different tasks can be developed under mild hypotheses. The authors of (Kelly & Heywood, 2018; Teh et al., 2017; Wilson, Fern, Ray, & Tadepalli, 2007) developed strategies to share knowledge among similar tasks (e.g., playing different Atari games (Kelly & Heywood, 2018)) based respectively on genetic algorithms and the so-called "Distill and Transfer Learning" (Distral) approach, in which a *distilled* control policy that captures common behaviour across tasks is synthesized. Finally, in (Wilson et al., 2007) the authors proposed a hierarchical Bayesian framework that allows to rapidly infer the features of new environments and tasks based on the previously observed ones.

A different approach was followed in the researches related to multi-objective RL (Gábor, Kalmár, & Szepesvári, 1998; Giuseppi & Pietrabissa, 2020; C. Liu et al., 2015; Van Moffaert & Nowé, 2014), where different competing objectives are active at the same time. Pareto optimality was sought in (Van Moffaert & Nowé, 2014), whereas, in (Giuseppi & Pietrabissa, 2020), the authors developed a strategy involving different objective-specific DeepRL agents to optimally control a system under the hypothesis that the prioritisation of the objectives is known to the controller (Gábor et al., 1998).

Differently from the previous solutions, this paper proposes a framework built on a state space augmentation to cope with time-varying objectives.

RL and DeepRL algorithms typically rely on a Markov decision process (MDP) representation of the control problem. MDPs couple a discrete-time dynamic control systems and a reward (or cost) structure to define an optimization problem, whose solution is the optimal control law or *policy*. Under some assumptions on the system (see Section 2), the solution can be computed by DP algorithms, which use backward recursion to efficiently compute the optimal policy. The Bellman's principle of optimality (Bellman, 1966) is the underlying principle of DP. In this paper, focused on finite-horizon MDPs (FH-MDPs), the Bellman's principle of optimality is used to address the scalability problems by dividing the task into sub-tasks. Specialized DeepRL agents are then sequentilally trained for each sub-task and the resulting controller for the original task switches between the various DeepRL agents during the same episode.

We mention that, as will be analyzed in Section 2.2, the DeepRL methods used in the literature for episodic finite-horizon problems actually solve approximated FH-MDPs, in which the the controller is not provided with the information regarding the visiting time of the various states. This lack of information will be shown to be equivalent to aggregating all the visiting times of each state, yielding to an approximation on the value function estimation that DeepRL algorithms use to decide the control actions. In fact, under such aggregation, the value function estimation of each state remains fixed over the episode duration, whereas, in finite-horizon problems, its value is in general time-varying even if the reward is time-invariant since the reward-to-go of a state depends on its visiting time. In general, a correct evaluation of the value function at all times is important for the optimal control of the system (e.g., to correctly evaluate the control effort) and is even more relevant in constrained control problems (as in L-DQN (Giuseppi & Pietrabissa, 2020)). With the framework proposed in this work, finite-horizon DeepRL algorithms solve the actual FH-MDP (and not an approximated one) and, thus, return the value function estimation for all the episode time-steps.

A summary of the main contributions of the paper follows:

- We propose to address FH-MDPs with DeepRL agents considering an augmented state space, in which the state definition includes the time-step of the current episode, to save the time-dependency and to provide better estimations of the value functions. As a result, different values of the value function are now estimated for each state and for each time-step. Moreover, a more accurate estimation of the value function is obtained.
- To solve the scalability issue caused by the state space augmentation, we propose to apply the Bellman's principle of optimality by partitioning the episode over time into sub-tasks, each one governed by a different DeepRL agent. Then, the obtained controllers (the DeepRL agents) are used in a switched control fashion in the original task, without causing any performance/value function estimation degradation with respect to the case in which a unique DeepRL agent is trained

for the whole original task in the augmented state space.

- By considering the time-augmented state space, the proposed framework extends the practicable applications of the DeepRL algorithms to problems in which the reward changes over time either stochastically or deterministically (e.g., with a task consisting of a series of sub-tasks with different reward functions).
- In principle, the framework designed in this work enables the implementation of any state-of-the-art DeepRL agent to solve control tasks with time-varying objectives, under the only hypothesis that the selected agent is able to provide an accurate state-action value function estimation.

The contributions highlighted above extend to the time-varying objective case the classes of control tasks that DeepRL agent can correctly solve, without requiring any modification in their specific training algorithms.

The remainder of the paper is organized as follows: Section 2 provides the definition of FH-MDPs and of the considered state augmentation approach, and also analyses the approximations performed by current DeepRL implementations; Section 3 shows how the Bellman's principle of optimality can be used to address the scalability problem introduced by the state space augmentation; Section 4 discusses some numerical simulations; finally, Section 5 draws the conclusions and highlights future research directions.

2. Finite-horizon MDPs, current DeepRL solutions and Proposed Finite-Horizon DeepRL

In the RL literature, introductions to MDPs range from simple ones, aimed at providing the reader with sufficient elements for the understanding of RL and DeepRL approaches (Sutton & Barto, 1998), to more specialized and theoretically sound ones (Bertsekas, 2005), (Puterman, 1994). In this paper we opted for an intermediate choice, similar to the one of (Shimkin, 2011), in order to give the necessary insight to understand the proposed DeepRL solutions without diverting the readers' attention from the scope of the paper.

Section 2.1 defines the FH-MDPs. In Section 2.2, two approaches for DeepRL implementations are analysed, while in Section 2.3 the proposed approach is described.

2.1. Finite-Horizon MDPs with Non-Stationary Rewards

The FH-MDP will be defined under the assumptions that the system model is stationary, i.e., the system dynamics does not depend on time, and that the Markov property holds, i.e., the system evolution from a given state does not depend on past visited states or past control actions.

Let us consider the dynamic system

$$x_{t+1} = h(x_t, u_t), x \in \mathcal{X}, u \in \mathcal{U}, t \in \mathcal{T}, \tag{1}$$

with x_t and u_t denoting the state and the control action at time t, respectively, and where $h : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ is a stochastic function, $\mathcal{T} := \{0, 1, ..., T-1\}$ is the set of T time-instants of the finite-horizon, $\mathcal{X}$ is the discrete or continuous finite state space and $\mathcal{U}$ is the discrete or continuous finite action space (in general, the available control actions might depend on the current state, but we neglect this dependency for the sake of the presentation clarity). The state equation (1) defines a controlled Markov chain over the state space. Let $p(x, u, x')$ denote the transition probability from state x to state x' when the control action u is chosen.

The system dynamics is driven by the control policy. We consider stationary and non-stationary non-randomized control policies. A non-stationary non-randomized control policy, denoted by π, defines a time-dependent mapping from the state space to the action space:

$$\pi : \mathcal{X} \times \mathcal{T} \to \mathcal{U}. \tag{2}$$

If $\pi(x, t) = u$, the system chooses action $u \in \mathcal{U}$ when it is in state $x \in \mathcal{X}$ at time $t \in \mathcal{T}$. Let Π be the set of all the possible non-stationary control policies. A stationary non-randomized control policy, denoted by $\bar{\pi}$, defines a mapping between the state space and the action space and verifies the condition

$$\pi(x, t) = \bar{\pi}(x), \text{ for all } x \in \mathcal{X} \text{ and } t \in \mathcal{T}. \tag{3}$$

The set of stationary policies $\bar{\Pi}$ is a subset of Π.

FH-MDPs also require the definition of a reward structure. A non-stationary reward function is defined by $\rho : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \times \mathcal{T} \to \mathbb{R}$, with $\rho(x, u, x', t)$ being the reward gained by the system when the controller chooses action u in state x at time t and the next state is x'. The terminal reward is defined as $\rho^f : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$, with $\rho^f(x, u)$ being the reward associated to the terminal state x_T when $x_T = x$ and $u_T = u$. Considering a discrete state space, the one-step expected reward incurred by the system at time t under policy π is then computed as

$$\begin{aligned} r(x_t, u_t) = \mathbb{E}^{\pi}\{\rho(x, u, x', t) \big| x = x_t, u = u_t, x' \in \mathcal{X}\} = \\ = \sum_{x' \in \mathcal{X}} p^{\pi}(x_t, u_t, x')\rho(x_t, u_t, x', t), \end{aligned} \tag{4}$$

where $\mathbb{E}^{\pi}$ and p^{π} denote the expected value and the transition probability, respectively, under policy π. Note that the one-step reward (4) is finite since $\mathcal{X}$, ρ and ρ^f are so.

Finally, we need to define a reward criterion to evaluate the system performance. A common reward criterion in FH-MDPs is the expected discounted total reward, defined as

$$J^{\pi,\varsigma} = \mathbb{E}^{\pi,\varsigma}\left\{\sum_{t=0}^{T-1} \gamma^t \rho(x_t, u_t, x_{t+1}) + \gamma^T \rho^f(x_T, u_T)\right\} \tag{5}$$

where $\gamma \in (0, 1)$ is the discount factor which weights immediate versus delayed rewards, $\varsigma \in \Sigma$ is the initial state distribution such that $\varsigma(x) > 0$ for all $x \in \mathcal{X}$, with Σ denoting the set of possible initial distributions over the state set $\mathcal{X}$, and $\mathbb{E}^{\pi,\varsigma}$ denotes the expected value under policy π and initial state distribution ς.

In conclusion, a FH-MDP is represented by the tuple

$$\{\mathcal{X},\mathcal{T},\mathcal{U},h,\rho,\rho^f,\gamma,\varsigma\} \tag{6}$$

and, together with the reward (5), defines the finite-horizon optimization problem

$$\max_{\pi\in\Pi} J^{\pi,\varsigma} \tag{7}$$

aimed at finding the optimal policy π^* which maximizes $J^{\pi,\varsigma}$.

The problem (7) can be written in terms of non-stationary (state,action)-value functions as

$$\max_{\pi\in\Pi} J^{\pi,\varsigma} = \max_{\pi\in\Pi}\Big\{\sum_{x\in\mathcal{X}}\varsigma(x)Q^\pi\big(x,0,\pi(x,0)\big)\Big\}, \tag{8}$$

where $\varsigma(x)$ is the probability that x is the initial state and $Q^\pi(x,t,u)$ is the expected discounted total reward of the system visiting state x at time t, choosing action u and following the policy π thereafter, for the last $T-t$ time-steps:

$$\begin{aligned} Q^\pi(x,t,u) = \mathbb{E}^\pi\Bigg\{&\sum_{k=t}^{T-1}\gamma^{k-t}\rho(x_k,u_k,x_{k+1})+ \\ &+\gamma^{T-t}\rho^f(x_T,u_T)|x_t=x,u_t=u\Bigg\}, \end{aligned} \tag{9}$$

with $\mathbb{E}^\pi$ denoting the expected value under policy π.

2.2. DeepRL Solutions

RL and DeepRL algorithms solve the problem (7) by finding the optimal non-stationary value functions, denoted by Q^*, which are then used to find the optimal non-stationary policy, for all $x\in\mathcal{X}$ and $t\in\mathcal{T}$, as

$$\pi^*(x,t) = \arg\max_{u\in\mathcal{U}} Q^*(x,t,u). \tag{10}$$

Two approaches are presented in this section: the former relying on an augmentation of the state space (Boyan & Littman, 2001; L. Liu & Sukhatme, 2018; Rachelson, Fabiani, & Garcia, 2009), the latter based on Approximate Dynamic Programming (ADP) strategies.

2.2.1. Augmented state space approach

In FH-MDPs, to handle non-stationary policies and rewards, it is convenient to define the augmented state space

$$\mathcal{Z} := \mathcal{X} \times \mathcal{T} = \big\{[x,t] \big| x \in \mathcal{X}, t \in \mathcal{T}\big\}, \tag{11}$$

where each augmented state $[x,t]$ represents a state of the original state space and its visit time.

The equivalent augmented FH-MDP is defined as

$$\big\{\mathcal{Z}, \mathcal{T}, \mathcal{U}, g, \rho, \rho^f, \gamma, \varsigma\big\}, \tag{12}$$

where $g : \mathcal{Z} \times \mathcal{U} \to \mathcal{Z}$ is such that the dynamics (1) is written in the augmented space as

$$[x_{t+1}, t+1] = \big(h(x_t, u_t), t+1\big) = g([x_t, t], u_t) \tag{13}$$

and where, with little abuse of notation, we assume that $\rho\big([x,t], u, g([x,t], u)\big) = \rho\big(x, u, h(x,u), t\big)$ and $\rho^f([x,t], u) = \rho^f(x, u)$. Equation (13) shows that in each episode the state $[x,t]$ is either visited once or not visited.

Remark 1. The augmented FH-MDP is stationary since non-stationary rewards $\rho(x, u, x', t)$ and policies $\pi(x,t)$ in the original state space are mapped onto stationary rewards $\rho([x,t], u, [x', t+1])$ and policies $\pi([x,t])$ in the augmented state space.

The non-stationary value functions $Q^\pi(x,t,u)$ are computed as stationary value functions in the augmented state space as

$$\begin{aligned} Q^\pi([x,t], u) = \mathbb{E}^\pi \Bigg\{ & \sum_{k=t}^{T-1} \gamma^{k-t} \rho([x_k, k], u_k, [x_{k+1}, k+1]) + \\ & + \gamma^{T-t} \rho^f([x_T, T], u_T) | [x_t, t] = [x,t], u_t = u \Bigg\}, \end{aligned} \tag{14}$$

In the augmented state space, standard DP and RL algorithms can be used to evaluate or estimate $Q^\pi([x,t],u)$ in an iterative fashion. We note that, from the Bellman's principle, the optimal value functions $Q^*\big([x,0], u)$ and the optimal policy π^* are independent of the initial state distribution ς, which, however, is still needed to compute the optimal cost $J^{\pi^*,\varsigma}$.

In DeepRL, the value function estimates are computed using Deep Neural Networks (DNNs), referred to as *critic* DNNs. In case of continuous action set, a DNN, named *actor* DNN (Grondman, Busoniu, Lopes, & Babuska, 2012), is also used to estimate $\pi([x,t])$. Such DNNs will be denoted as $\mathcal{Q}([x,t], u|\varphi)$ and $\mathcal{A}([x,t]|\vartheta)$, respectively, where φ and ϑ are the DNNs' parameters. The critic DNN $\mathcal{Q}$ maps $\mathcal{Z} \times \mathcal{U}$ onto $\mathbb{R}$ and is aimed at estimating the optimal value function $Q^*([x,t], u)$. The actor DNN $\mathcal{A}$ maps the augmented state set $\mathcal{Z}$ onto the action set $\mathcal{U}$ and is aimed at estimating the optimal policy $\pi^*(x,t)$.

Standard DeepRL algorithms, such as DQN (Mnih et al., 2015), DDQN (Van Hasselt et al., 2016), DDPG (Lillicrap et al., 2015) and TD3 (Fujimoto et al., 2018), can be used to find the critic DNN $\mathcal{Q}$ and, in the DDPG and TD3 cases, the the actor DNN $\mathcal{A}$. By assuming perfect estimates, i.e., that $\mathcal{Q}([x,t],u|\varphi) = Q^*([x,t],u)$ and $\mathcal{A}([x,t]|\vartheta) = \pi^*(x,t)$ for all $[x,t] \in \mathcal{Z}$ and $u \in \mathcal{U}$, the non-stationary optimal policy for the problem (7) is computed as

$$\pi^*(x,t) = \arg\max_{u\in\mathcal{U}} \mathcal{Q}([x,t],u|\varphi) \tag{15}$$

for discrete action spaces, or

$$\pi^*(x,t) = \mathcal{A}([x,t]|\vartheta), \tag{16}$$

for continuous action spaces.

2.2.2. Aggregated state space approach

With the state space augmentation, the problem becomes rapidly unfeasible as the time-horizon length increases. Alternatively, problems in which a task or episode has a finite duration can be treated as problems with a continuous task by using discounting (i.e., at time t of each episode, the reward is discounted by γ^t) and by restarting the time at the end of each episode (Barto et al., 1983). Two approximations are however implied:

(1) *Policy aggregation.* The policies which are considered are stationary in the original state space $\mathcal{X}$, as in (3). This first approximation is commonly used in ADP to restrict the policy space.
(2) *State aggregation.* The one-step expected reward $r(x_t, u_t)$ is estimated as it was independent of the visit time - which is not true when considering time-varying policies and rewards. This second approximation is a state aggregation with respect to the augmented state space $\mathcal{Z}$: each state $x \in \mathcal{X}$ represents all the states $[x,t] \in \mathcal{Z}$, for all $t \in \mathcal{T}$. This approximation might be acceptable if, during the episode, the system rarely visits the same state, whereas it might be problematic if the states are repeatedly visited.

Remark 2. Up to the authors' knowledge, all the DeepRL algorithms, including the ones mentioned in Section (1.1), have been used under the described approximations and with stationary reward functions.

Hereafter, the aggregated FH-MDP will be denoted by $\overline{\text{FH-MDP}}$ and its aggregate policies $\bar{\pi}$ coincide with the stationary policies of equation (3).

Under the state aggregation, the reward-to-go of the system in a given state x is associated to a single estimated value, regardless of the visit time. Strictly speaking, the resulting process is not Markovian anymore, since the expected reward in a state x depends on the time of the visit, i.e., on the history of the system. The system can be also regarded as a Partially Observables MDP (POMDP) (S. P. Choi, Yeung, & Zhang, 2000; S. P. M. Choi, Yeung, & Zhang, 2000; Hallak, Castro, & Mannor, 2015; Padakandla, J, & Bhatnagar, 2019): when the system is in state x, the controller operates without knowing in which augmented state the system lies among $[x,0], [x,1], ..., [x,T-1]$. This means that the estimated expected reward depends on the probability distribution of the visit times of the state x over the states $[x,t]$, for all $t \in \mathcal{T}$, i.e., on the probability distribution of the visit times of the state x during the time interval $\mathcal{T}$. This probability distribution depends, in turn, on the initial state distribution (i.e., the probability that the system is in a given state at time $t=0$) and on the control policy. For example, if the system almost always starts from a state x and then follows a policy under which the state x is rarely visited, the probability that the state x is visited at time $t=0$ is much larger than the probabilities that the state is visited at time $t=1,...,T-1$.

Let $\xi^{\bar{\pi},\varsigma}([x,t]|x)$ denote the probability that the augmented state is $[x,t]$ given that the system is in state x, when the system is under aggregated policy $\bar{\pi}$ and the initial state distribution is ς. Considering the aggregated $\overline{\text{FH-MDP}}$, the critic DNN is then aimed at estimating the stationary value function $\bar{Q}^{\bar{\pi}}\big(x,u\big|\varsigma\big)$, which is the sum over the episode time, weighted by $\xi^{\bar{\pi},\varsigma}$, of value functions $Q^{\bar{\pi}}([x,t],u)$:

$$\bar{Q}^{\bar{\pi}}\big(x,u\big|\varsigma\big) = \sum_{t=0}^{T-1} \xi^{\bar{\pi},\varsigma}([x,t]|x) Q^{\bar{\pi}}([x,t],u). \tag{17}$$

The aggregated $\overline{\text{FH-MDP}}$ is defined by the tuple

$$\big\{\mathcal{X},\mathcal{T},\mathcal{U},h,\rho,\rho^f,\gamma,\varsigma\big\}, \tag{18}$$

and the optimal control problem solved by DeepRL algorithms is an approximation of the original one (7) and is stated as

$$\max_{\bar{\pi}\in\bar{\Pi}} \Big\{ \sum_{x\in\mathcal{X}} \varsigma(x) \bar{Q}^{\bar{\pi}}\big(x,\bar{\pi}(x)\big|\varsigma\big) \Big\}. \tag{19}$$

Remark 3. The value function (17) may be interpreted as the average of the values of $Q^{\bar{\pi}}([x,t],u)$, for all $t \in \mathcal{T}$, weighted by the probability distribution of the visit times t over the state x induced by $\bar{\pi}$ and ς. Equation (17) shows that the optimal stationary value function $\bar{Q}^*(x,u)$ under the aggregated optimal policy $\bar{\pi}^*$ differs from the non-stationary optimal value function $Q^*(x,0,u)$ under the optimal policy π^* - unless $\bar{\pi}^* \approx \pi^*$ and $\xi^{\bar{\pi},\varsigma}([x,t]|x) \approx 1$.

Remark 4. Equation (17) gives another insight of DeepRL algorithms applied to approximated $\overline{\text{FH-MDPs}}$: in addition to implicitly estimating the transition probabilities under the optimal policy (as done by standard RL algorithms applied to finite- or infinite-horizon MDPs), the critic DNN $\bar{\mathcal{Q}}$ of current implementations of DeepRL algorithms has the further task of implicitly estimating the probability distribution of the visiting times of each state. If the reward is time-varying, the task is even harder since a unique estimated value $\bar{\mathcal{Q}}(x,u|\varphi)$ cannot express the fact that the reward gained in a state x when action u is chosen changes with the visit time.

In DeepRL approaches, the critic DNN, denoted by $\bar{\mathcal{Q}}(x,u|\varphi)$, maps $\mathcal{X}\times\mathcal{U}$ onto $\mathbb{R}$ and estimates the optimal value function $\bar{Q}^*(x,u|\varsigma)$. The actor DNN $\bar{\mathcal{A}}$ maps the state set $\mathcal{X}$ onto the action set $\mathcal{U}$ and is aimed at estimating the optimal aggregated policy $\bar{\pi}(x,t)$. By assuming perfect estimates, the optimal aggregated policy for the problem (19) is computed as

$$\bar{\pi}^*(x) = \arg\min_{u \in \mathcal{U}} \bar{\mathcal{Q}}(x, u|\varphi), \tag{20}$$

in case of discrete action space, or

$$\bar{\pi}^*(x) = \bar{\mathcal{A}}(x|\vartheta), \tag{21}$$

in case of continuous action space.

A simple example of FHMDP is shown in the Annex to clarify the approximation yielded by the aggregate modeling.

2.3. Proposed Finite-Horizon DeepRL

In this paper, the proposed approach is to apply DeepRL algorithms with a twofold objective: i) to solve the original problem (7) (and not the approximated one (19)) in the augmented state space $\mathcal{Z}$ to overcome the approximation errors caused by the policy and state aggregations used by current approaches; ii) to cope with time-varying rewards. The scalability disadvantage of the augmented state modeling is not negligible, given that the state-action value functions are estimated over $\mathcal{X} \times \mathcal{T} \times \mathcal{U}$ and not over $\mathcal{X} \times \mathcal{U}$. In Section 3.1, we will show that it is possible to overcome the scalability problem by applying the Bellman's principle of optimality.

3. Bellman's principle of optimality for DeepRL

Section (3.1) shows how the Bellman's principle of optimality can be used to address the scalability problem of the augmented state space by dividing the overall control task into a number of sub-tasks. It is shown that the optimal solution of the task can be obtained by using, in switched-control fashion, the optimal controllers for the sub-tasks. The result can be also applied, with some approximations, to the case of aggregated state space, as shown in Section 3.2, and also in case part of the sub-tasks are defined over an augmented state space and part over an approximated one, as described in Section 3.3.

3.1. Case of Augmented FH-MDPs

Let us consider the FH-MDP (12) with augmented state space $\mathcal{Z} = \mathcal{X} \times \mathcal{T}$. The results of this Section will stated be under the following assumption:

Assumption 1. The critic DNN $\mathcal{Q}([x,t], u|\varphi)$ of the FH-MDP $\left\{\mathcal{Z}, \mathcal{U}, g, \rho, \rho^f, \gamma, \varsigma\right\}$ perfectly estimates the optimal value function $Q^*([x,t], u)$. In case $\mathcal{U}$ is a continuous action space, the actor DNN $\mathcal{A}([x,t]|\vartheta)$ perfectly estimates the optimal policy $\pi^*(x,t)$.

Let us define n sub-sets of time-instants, denoted by $\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_n$, such that the set $\mathcal{T}_i$ is defined as

$$\mathcal{T}_i := \{t_1^i, t_2^i, ..., t_{T_i}^i\}, \tag{22}$$

for $i = 1, ..., n-1$, where T_i is the cardinality of the set $\mathcal{T}_i$, $t_1^1 = 0$, $t_1^{i+1} = t_{T_i}^i + 1$ and $t_{T_n}^n = T - 1$. The defined sets $\mathcal{T}_i$'s form a partition of the original set of time-instants $\mathcal{T}$. By defining the augmented state spaces $\mathcal{Z}_i = \mathcal{X} \times \mathcal{T}_i$, n MDPs can be defined as

$$\{\mathcal{Z}_i, \mathcal{U}, g, \rho, \rho_i^f, \gamma, \varsigma^i\}, \text{ for } i = 1, ..., n, \tag{23}$$

where ς_i and ρ_i^f are the initial state distribution and the final reward of the i-th FH-MDP, respectively, and where $\varsigma_1 = \varsigma$. The following Property 1 states that the problem (7) can be solved by solving the n sub-problems (18) and is a consequence of the Bellman's principle of optimality.

Property 1. The optimal non-stationary policy and value functions and the optimal expected discounted reward of the finite-horizon problem (7) can be computed by sequentially solving the n sub-problems (23), starting from the n-th one to the first one, with final rewards defined as

$$\begin{aligned} \rho_i^f([x, t_{T_i}^i], u) = r([x, t_{T_i}^i], u) + \\ + \gamma \max_{u' \in \mathcal{U}} \mathcal{Q}_{i+1}([x_{t_1^{i+1}}, t_1^{i+1}], u' | \varphi_{i+1}), \end{aligned} \tag{24}$$

for $i = 1, ..., n-1$, and $\rho_n^f = \rho^f$.

Proof. Firstly, let us consider the n-th (last) FH-MDP

$$\{\mathcal{Z}_n, \mathcal{U}, g, \rho, \rho^f, \gamma, \varsigma_n\}, \tag{25}$$

with augmented state space $\mathcal{Z}_n = \mathcal{X} \times \mathcal{T}_n$. By solving this problem, the critic DNN $\mathcal{Q}_n([x, t], u | \varphi_n)$ is obtained, which estimates the optimal value functions $Q_n^*([x, t], u)$ of the MDP (25) for all $x \in \mathcal{X}$ and $t \in \mathcal{T}_n$. According to the Bellman's principle of optimality, the optimal value function $Q_n^*([x, t], u)$ of the n-th FH-MDP (25) coincides with the optimal value function $Q^*([x, t], u)$ of the original FH-MDP (12):

$$Q^*([x, t], u) = Q_n^*([x, t], u), \forall x \in \mathcal{X}, \forall t \in \mathcal{T}_n. \tag{26}$$

Consequently, under Assumption 1, it holds that, for all $x \in \mathcal{X}$ and for all $t \in \mathcal{T}_n$,

$$\mathcal{Q}([x, t], u | \varphi) = \mathcal{Q}_n([x, t], u | \varphi_n), \tag{27}$$

where $\mathcal{Q}$ is the critic DNN perfectly estimating the optimal value functions Q^* of the original problem (12), and that the optimal policy π^* is obtained in $\mathcal{T}_n$ as

$$\pi^*([x,t]) = \pi_n^*([x,t]) = \arg\max_{u \in \mathcal{U}} \mathcal{Q}_n([x,t],u|\vartheta_n), \tag{28}$$

in case of discrete action set, or as

$$\pi^*([x,t]) = \pi_n^*([x,t]) = \mathcal{A}_n([x,t]|\vartheta_n), \tag{29}$$

in case of continuous action space.

Then, let us define the $(n-1)$-th FH-MDP as

$$\{\mathcal{Z}_{n-1}, \mathcal{U}, g, \rho, \rho_{n-1}^f, \gamma, \varsigma_{n-1}\}, \tag{30}$$

with augmented state space $\mathcal{Z}_{n-1} = \mathcal{X} \times \mathcal{T}_{n-1}$.

In the original FH-MDP (12) under the optimal policy, the one-step expected reward at time t plus the discounted expected reward from time t on is computed as

$$r([x_t,t],u_t) + \gamma Q^*\big([x_{t+1},t+1], \pi^*([x_{t+1},t+1])\big). \tag{31}$$

Given equations (26) and (27), the quantity in equation (31) at time $t = t^{n-1}_{T_{n-1}}$ is computed as the final reward (24), with $i = n-1$, using the estimated values of the n-th critic DNN $\mathcal{Q}_n$ at time $t = t^{n-1}_{T_{n-1}} + 1 = t_1^n$ under the optimal poicy π_n^*.

As with the last problem (25), by solving the second-to-last problem (30), the critic DNN $\mathcal{Q}_{n-1}$ and the optimal policy π_{n-1}^* for the MDP (30) are obtained, which coincide with the critic DNN $\mathcal{Q}$ and with the optimal policy π^* of the original FH-MDP (12), for all $t \in \mathcal{T}_{n-1}$.

The procedure is iterated for the remaining FH-MDPs

$$\{\mathcal{Z}_i, \mathcal{T}_i, \mathcal{U}, g, \rho, \rho_i^f, \gamma, \varsigma_i\}, \tag{32}$$

for $i = n-2, n-3, ..., 1$. The FH-MDPs (32), solved starting from $i = n-2$ to $i = 1$, return the remaining critic DNNs $\mathcal{Q}_i$ and optimal policies π_i^*.

In conclusion, recalling that the $\mathcal{T}_i$'s form a partition of $\mathcal{T}$, the Belmann's principle of optimality guarantees that the non-stationary critic DNN for the original problem (6) can be obtained from the critic DNNs of the n sub-problems (23) as

$$\mathcal{Q}(x,t,u|\varphi) := \begin{cases} \mathcal{Q}_1([x,t],u|\varphi_1) & \text{if } t \in \mathcal{T}_1 \\ \mathcal{Q}_2([x,t],u|\varphi_2) & \text{if } t \in \mathcal{T}_2 \\ ... & \\ \mathcal{Q}_n([x,t],u|\varphi_n) & \text{if } t \in \mathcal{T}_n \end{cases}, \tag{33}$$

where $\varphi := [\varphi_1, \varphi_2, ..., \varphi_n]^T$ is the vector collecting the critic DNNs' parameters. Consequently, the optimal non-stationary policy for the original problem (12) is computed as

$$\pi^*(x,t) = \arg\max_{u\in\mathcal{U}} \mathcal{Q}([x,t],u|\varphi) = \begin{cases} \pi_1^*([x,t]) & \text{if } t\in\mathcal{T}_1 \\ \pi_2^*([x,t]) & \text{if } t\in\mathcal{T}_2 \\ \dots \\ \pi_n^*([x,t]) & \text{if } t\in\mathcal{T}_n \end{cases}, \tag{34}$$

for discrete action spaces, or

$$\pi^*(x,t) = \mathcal{A}([x,t]|\vartheta) := \begin{cases} \mathcal{A}_1([x,t]|\vartheta_1) & \text{if } t\in\mathcal{T}_1 \\ \mathcal{A}_2([x,t]|\vartheta_2) & \text{if } t\in\mathcal{T}_2 \\ \dots \\ \mathcal{A}_n([x,t]|\vartheta_n) & \text{if } t\in\mathcal{T}_n \end{cases}, \tag{35}$$

for continuous action spaces, where $\vartheta := [\vartheta_1, \vartheta_2, ..., \vartheta_n]^T$ is the vector collecting the actor DNNs' parameters.

The knowledge of ς_i is not required for computing the optimal value function and the optimal policy for the sub-problems $i = 2, ..., n$, whereas the initial state distribution ς is necessary to compute the optimal expected reward as

$$J^{\varsigma,*} = \sum_{x\in\mathcal{X}} \varsigma(x)\mathcal{Q}_1\big([x,0], \pi_1^*([x,0])\big) \tag{36}$$

■

3.2. Case of Aggregated FH-MDPs

A similar *divide et impera* approach can be adopted also for the approximated $\overline{\text{FH-MDP}}$ (18). Starting from the time set partition $\{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_n\}$, n approximated $\overline{\text{FH-MDPs}}$ are obtained, for $i = 1, ..., n$, as

$$\{\mathcal{X}, \mathcal{U}, h, \rho, \rho_i^f, \gamma, \varsigma_i\}, \tag{37}$$

where ρ_i^f is the i-th final reward and ς_i is the state distribution at time t_1^i. Analogously with the augmented FH-MDP case, the final rewards of the sub-problems are computed as $\rho_n^f = \rho^f$ and, for $i = 1, ..., n-1$, as

$$\rho_i^f(x_{t_{T_i}^i}, u_{t_{T_i}^i}) = r(x_{t_{T_i}^i}, u_{t_{T_i}^i}) + \gamma \max_{u'\in\mathcal{U}} \bar{\mathcal{Q}}_{i+1}(x_{t_1^{i+1}}, u'|\varphi_{i+1}), \tag{38}$$

where $\bar{\mathcal{Q}}_i$ is the critic DNN solving the i-th $\overline{\text{FH-MDP}}$.

Application examples might be tasks constituted by multiple temporally consecutive sub-tasks with different rewards: while standard DeepRL approaches cannot cope with this kind of problems, the proposed approach can be used by defining an aggregated $\overline{\text{FH-MDP}}_i$ for each sub-task i, with the sub-tasks linked by equation (38).

Differently from the former case, the estimation of the value functions (17) requires that the episodes initial states of all the sub-problems are chosen according to the state distributions ς_i at time t_1^i for all $i = 1, ..., n$. Note that this is true for the initial state distribution ς also in the common DeepRL approaches (Fujimoto et al., 2018; Lillicrap et al., 2015; Mnih et al., 2015). The distributions should then be estimated based on the knowledge of the system and the tasks. A trivial approximation is for example to assume that the initial distribution is uniform over the state space, as often assumed for ς (Brockman et al., 2016). Note that poor estimations of the ς_i's introduce a further approximation in the evaluation of the optimal policy and of the optimal value functions, which shall be evaluated on a case-by-case basis.

In the original FH-MDP, the overall non-stationary critic DNN is computed as

$$\mathcal{Q}(x, t, u|\varphi) := \begin{cases} \bar{\mathcal{Q}}_1(x, u|\varphi_1) & \text{if } t \in \mathcal{T}_1 \\ ... & \\ \bar{\mathcal{Q}}_n(x, u|\varphi_n) & \text{if } t \in \mathcal{T}_n \end{cases}, \tag{39}$$

and the sub-optimal stationary policy as

$$\pi(x) = \arg\max_{u \in \mathcal{U}} \bar{\mathcal{Q}}(x, u|\varphi) = \begin{cases} \bar{\pi}_1^*(x) & \text{if } t \in \mathcal{T}_1 \\ ... & \\ \bar{\pi}_n^*(x) & \text{if } t \in \mathcal{T}_n \end{cases}, \tag{40}$$

for discrete action spaces, or as

$$\pi(x) = \bar{\mathcal{A}}(x|\vartheta) := \begin{cases} \bar{\mathcal{A}}_1(x|\vartheta_1) & \text{if } t \in \mathcal{T}_1 \\ ... & \\ \bar{\mathcal{A}}_n(x|\vartheta_n) & \text{if } t \in \mathcal{T}_n \end{cases}, \tag{41}$$

for continuous action spaces.

3.3. Case of Partially Aggregated FH-MDPs

It is conceivable - and, in some cases, convenient in practice - to use the aggregated state space representation during some $\mathcal{T}_i$'s (e.g., the initial ones, when, due to the discount factor, the optimal policy might behave as if the problem was an infinite-horizon one) and to use the augmented state space representation in other $\mathcal{T}_i$'s (e.g., the final ones, when the effect of the final reward cannot be neglected). The DNNs and policies of each time set are straightforwardly derived from the ones in equations (33)-(35) and (39)-(41).

For instance, let $n = 2$ and let one aggregated $\overline{\text{FH-MDP}}$ be defined in the first time set $\mathcal{T}_1$ and one augmented FH-MDP in the second time set $\mathcal{T}_2$:

$$\begin{cases} \overline{\text{FH-MDP}}_1 : \{\mathcal{X}, \mathcal{U}, h, \rho, \rho_1^f, \gamma, \varsigma\} \\ \text{FH-MDP}_2 : \{\mathcal{Z}_2, \mathcal{U}, g, \rho, \rho^f, \gamma, \varsigma_2\} \end{cases}, \tag{42}$$

with ρ_1^f defined as in equation (38). The critic DNN is then

$$\mathcal{Q}(x,t,u|\varphi) := \begin{cases} \bar{\mathcal{Q}}_1(x,u|\varphi_1) & \text{if } t \in \mathcal{T}_1 \\ \mathcal{Q}_2([x,t],u|\varphi_2) & \text{if } t \in \mathcal{T}_2 \end{cases}, \tag{43}$$

and the non-stationary policy for the original FH-MDP is

$$\pi(x,t) = \begin{cases} \arg\max_{u\in\mathcal{U}} \bar{\mathcal{Q}}_1(x,u|\varphi_1) & \text{if } t \in \mathcal{T}_1 \\ \arg\max_{u\in\mathcal{U}} \mathcal{Q}_2([x,t],u|\varphi_2) & \text{if } t \in \mathcal{T}_2 \end{cases}, \tag{44}$$

for discrete action spaces, or

$$\pi(x,t) = \begin{cases} \bar{\mathcal{A}}_1(x|\vartheta_1) & \text{if } t \in \mathcal{T}_1 \\ \mathcal{A}_2([x,t]\big|\vartheta_2) & \text{if } t \in \mathcal{T}_2 \end{cases}, \tag{45}$$

for continuous action spaces.

Note that, under Assumption 1, the found value functions and the policy are optimal for the original FH-MDP for all $t \in \mathcal{T}_2$ since a perfect estimation of ς_2 is not necessary in the augmented state space problem. For all $t \in \mathcal{T}_1$, the value functions and the policy are optimal for the approximated $\overline{\text{FH-MDP}}_1$, considering the optimization problem (19), since the final reward $\rho_1^f(x)$ is correctly estimated by $\mathcal{Q}_2([x,t_1^2],\pi(x,t_1^2)|\varphi_2)$, but they are sub-optimal for the original FH-MDP (7).

This kind of *partially aggregated* MDP is shown to perform well in the second simulation example of Section 4,

3.4. Pseudo-code of the proposed algorithm

We report here the pseudo-code for the proposed training and control procedure, tailored to the case of section 3.1.

4. Simulations

To demonstrate the characteristics of the proposed framework, we developed a simulation environment starting from the classic benchmark problem for RL solutions of the cart-pole balancing problem, originally proposed in (Barto et al., 1983). The considered system, depicted in Figure 1, is constituted by a 1kg cart with the task of balancing a $0.5\text{m}, 0.1\text{kg}$ pole. The state of the system is formed by the horizontal position of the cart p, the angular position of the pole θ, and their first derivatives. The control action consists of the application of a force F to the cart that can range in the continuous set $[-30,+30]$N. The sampling time was set to 0.02s.

The symbolic dynamical model of the cart-pole is the following (Florian, 2007):

$$\begin{cases} \ddot{p} = \frac{F+ml\left(\dot{\theta}^2\sin\theta - \ddot{\theta}\cos\theta\right)}{(M+m)} \\ \ddot{\theta} = \frac{g\sin\theta + \cos\theta\left[\frac{-F-ml\dot{\theta}^2\sin\theta}{m+M}\right]}{l\left(\frac{4}{3}-\frac{m\cos^2\theta}{m+M}\right)} \end{cases}, \tag{46}$$

Algorithm 1: Design process of the proposed DeepRL agent: Training and Control

```
1  Training{
2      Divide the control horizon into the sets $\mathcal{T}_1, \mathcal{T}_2, ...\mathcal{T}_n$ defined in (22);
3      for $i$=$n$ to 1 do
4          augment the state space of the system as $\mathcal{Z}_i = \mathcal{X} \times \mathcal{T}_i$ ;
5          define the final cost $\rho_i^f$ of agent $i$ as in (24) (i.e., using the original final
             reward $\rho^f$ for agent $n$ and the critic of agent $i+1$, $\mathcal{Q}_{i+1}$, for the other
             agents);
6          train a DeepRL agent (e.g., utilising the actor-critic TD3 algorithm) to
             solve the control task with the augmented state over the time interval
             $\mathcal{T}_i$, with final cost $\rho_i^f$;
7      end
8      return the trained actors $\mathcal{A}_1, ..., \mathcal{A}_n$;
9  }
10 Control{
11     for $t = 0$ to $T - 1$ do
12         observe the system state $x_t$ at time $t \in \mathcal{T}_i$ (we recall from (22) that each
             time-step $t$ belongs to exactly one of the sets $\mathcal{T}_1, \mathcal{T}_2, ...\mathcal{T}_n$);
13         define the augmented state of the system as $z_t = [x_t, t]$;
14         use the trained actor $\mathcal{A}_i$ to determine $u_t$ on the basis of the state
             feedback $z_t$;
15         actuate the control $u_t$;
16     end
17 }
```

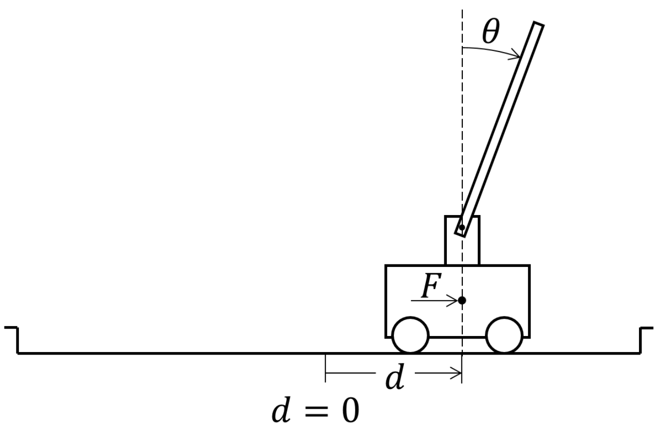

Figure 1. Cart-pole system schematics adapted from (Barto et al., 1983).

in which m, M are respectively the masses of the pole and the cart, l is half the length of the pole and g is the gravity.

The environment was implemented in OpenAI Gym (Brockman et al., 2016) starting from the official “CartPole-v0” environment. All the simulations were developed in Python 3.7 with PyTorch 1.2 and were run on a PC equipped with an i7 processor with 16GB of RAM. The code was developed starting from the implementation of the TD3 algorithm (Fujimoto et al., 2018) provided by the authors and available on (Fujimoto, Hoof, & Meger, 2020). Details on the simulation hyper-parameters are provided in section 4.1.2. In line with (Fujimoto et al., 2018), in Figures 2-7 a solid line represents the mean values of the depicted quantities and a shaded region represents half of their standard deviation.

4.1. First Simulation: Deterministic Time-Varying Reward

4.1.1. Task Description

The original task proposed in (Barto et al., 1983) consisted in controlling the system so that its state remains in the region defined by $-2.4\text{m} \le p \le 2.4\text{m}$ and -12 rad $\le \theta \le 12$ rad. The rationale of the control objective was that the pole should not not fall down while the cart should remain in its operative region.

In this paper, to better demonstrate the advantages of the proposed approach, the control objective is modified so that the controller shall try to maintain the system in the region where $p < 0\text{m}$ for 100 time-steps, after which the system should be driven to the region $p > 0\text{m}$ for an additional 50 time-steps. This time-varying objective is captured by the following reward function:

$$\rho(p(t), t) = \begin{cases} +1.5 & \text{if } t \le 100 \text{ and } p(t) < 0 \\ +4 & \text{if } 100 < t \le 150 \text{ and } p(t) > 0 \\ -0.5 & \text{otherwise} \end{cases} . \tag{47}$$

To speed up training, the controller receives a terminal reward $\rho^f = -10$ whenever the pole falls or the cart leaves its operative boundaries. The discount factor γ is set

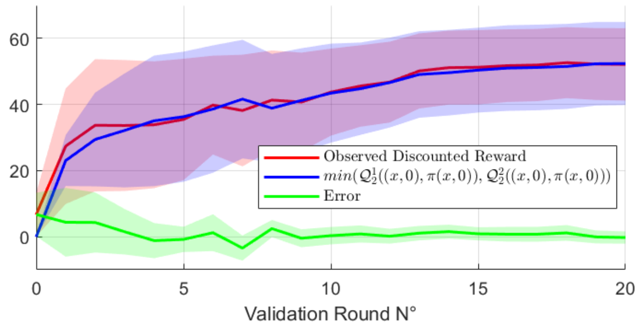

Figure 2. Simulation 1: Comparison between the estimated value function and the observed discounted reward when controlled by $\mathcal{A}_2$.

to 0.95.

4.1.2. Controller Design

Being the objective deterministically time-variant, a natural choice to develop the controller is to divide the 150 time-steps of the control horizon into two time sets $\mathcal{T}_1$,$\mathcal{T}_2$ corresponding to the intervals over which the reward function (47) is stationary: $\mathcal{T}_1 = [0, 100), \mathcal{T}_2 = [100, 150]$.

Dealing with a problem with continuous control actions, the DeepRL algorithm selected to train the neural networks is the “Twin-Delayed Deep Deterministic policy gradient” (TD3) (Fujimoto et al., 2018), as it represents the state-of-the-art solution regarding the precision of value function estimation, which is a crucial element for the functioning of the proposed approach (we recall that Property 1 was proven under Assumption 1).

All the neural networks were trained for 10^5 time-steps. The training hyper-parameters were kept the same as the ones proposed by the authors of (Fujimoto et al., 2018) in (Fujimoto et al., 2020), save for the policy noise ϵ and its clip c that were set respectively to 0.3 and 0.6, for the agent controlling the system over $\mathcal{T}_2$, and 0.4 and 0.8, for the one trained for $\mathcal{T}_1$. We mention that both actor and critic networks were characterized by two hidden layers of 256 neurons with ReLu activation functions, followed by an output layer with a hyperbolic tangent activation function for the actors and a linear output layer for the critics. All the network weights were initialized as default in Pytorch 1.2.

4.1.3. First Simulation Results

Compliant with the procedure detailed in Section 3, the first agent to be trained was the one related to $\mathcal{T}_2$, for which training we sampled the initial conditions of the system uniformly over $[-0.1, 0.1]$ for the cart position and over $[-0.05, 0.05]$ for the other variables. For TD3 agents, two critic networks $\mathcal{Q}^1$, $\mathcal{Q}^2$ are trained, and between the two only the one providing the lower estimation is used at each time-step.

Figure 2 compares the estimation provided by $min(\mathcal{Q}_2^1, \mathcal{Q}_2^2)$ with the actual discounted reward observed during the evaluation (i.e., with no training) episodes. Each validation round in Figure 2 details, as in (Fujimoto et al., 2018), the mean and half of the standard deviation observed over 100 episodes, each with different initial conditions. It is clear from the figure that the estimations converge to the actual values, as the two curves overlay and the error between the two quantities converges in mean to

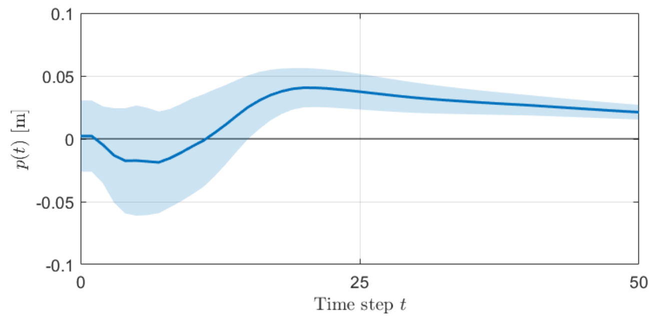

Figure 3. Simulation 1: Evolution of the cart position when controlled by $\mathcal{A}_2$.

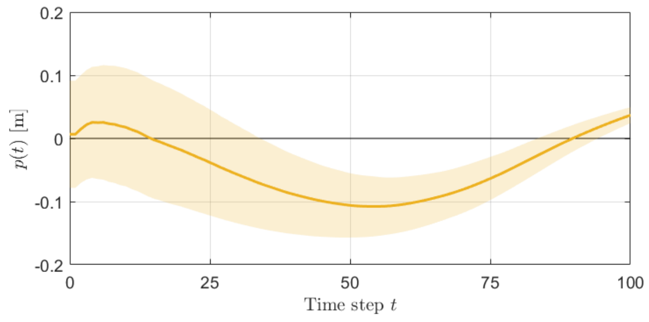

Figure 4. Simulation 1: Evolution of the cart position when controlled by $\mathcal{A}_1$.

zero with a very small deviation. The validation rounds were done every 5000 training time-steps and a similar behaviour was observed for all the other critics used in this section.

Figure 3 shows that the trained agent correctly steers the system towards the region in which $p(t) > 0$, with a small initial fluctuation to account for unfavorable initial conditions (e.g., negative starting position or particularly inclined pole). Note that the behaviour of this agent is in-line with the standard DeepRL solutions, as, in light of the proposed partition of the time-steps, this agent is subject to a stationary (non time-varying) reward.

Once this first agent was trained, it was possible to utilise its initial estimation of the discounted reward as a final reward function ρ_1^f for the other agent, according to equation (24). The distribution of $p(0)$ was set uniform over $[-0,3,0.3]$m, while the initial values of all the other state variables were drawn from a uniform distribution between -0.05 and 0.05, as in the original simulation environment. The cart behaviour is reported in Figure 4: we note that the system is correctly steered to the region $p(t) < 0$ and is then driven towards $p(t) > 0$, with a crossing time that occurs, in mean, at $t = 91$. This behaviour is due to the fact that the agent is able to foresee that the objective is going to change, thanks to the final reward ρ_1^f provided at time $t = 100$ during the training, which was learned by the other agent as the estimation of the future discounted reward after $t = 100$. Note that without the information from the first agent regarding ρ_1^f, this agent would have shown a behaviour equivalent (in fact symmetric) to the one of figure 3, without the additional crossing of the $p(t) = 0$ line. In fact, what we observe is that the agent prefers to sacrifice some immediate rewards (obtained for $p(t) < 0$) in favour of a greater reward in the subsequent task,

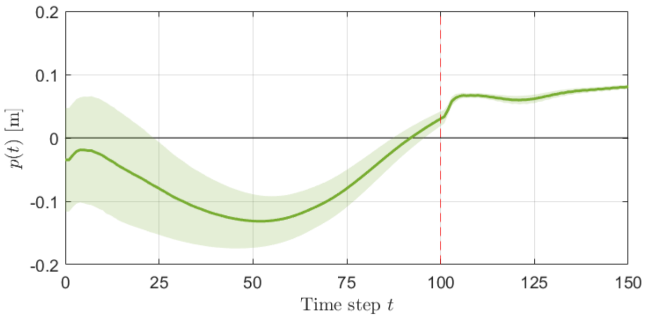

Figure 5. Simulation 1: Evolution of the cart position when controlled by the complete FH-MDP agent. Red dashed vertical line: switching instant between the two agents.

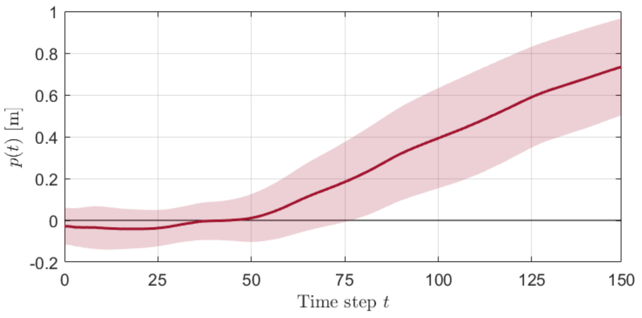

Figure 6. Simulation 1: Evolution of the cart position when controlled by one $\overline{\text{FH-MDP}}$ agent (as in standard DeepRL solutions).

that can be attained by placing the cart in the region $p(t) > 0$ at $t = 100$ with a favorable pole angle.

Figure 5 reports the evolution of the cart position evaluated when controlled by both agents in a switched control fashion. The figure clearly shows that the overall controller manages to handle the time-varying reward, a task that would have been impossible for a standard DeepRL agent trained on a state representation that did not include t, as will be discussed in the following subsection. We note how the system evolves in a way that is very similar to the composition of the figures 3 and 4, save for the fact that the agent is able to keep the cart in the region $p(t) > 0$ for all times $100 < t \leq 150$. The agent prevents the loss of some high rewards in $\mathcal{T}_2$ (+4 according to (47)) by renouncing to a few positive, but low, rewards towards the end of $\mathcal{T}_1$, anticipating the crossing of $p(t) = 0$, in average, to $t = 93$.

The following subsection compares the results of the proposed agent with a standard DeepRL agent and shows the advantages of the proposed framework to address time-varying tasks.

4.1.4. Approximated $\overline{\text{FH-MDP}}$ Benchmark

For the sake of comparison, Figure 6 reports the behaviour of the cart when controlled by a single TD3 agent trained to solve the problem formulated as a $\overline{\text{FH-MDP}}$ (i.e., without including the time-step t in the system state) over the entire time horizon. This comparison was designed to show how a state-of-the-art agent solves the given control task under the state aggregation discussed in section 2.2.2.

In this case the system is quickly driven to $p(t) > 0$, as, according to (47), the expected reward in that region is larger than the one associated with $p(t) < 0$. Overall, this agent is correctly trained, in the sense that it seeks the maximum reward that it can obtain with the observations it is provided with, but the lack of the explicit knowledge on the time instant at which the various cart-pole states are visited leads to an overall less performing control strategy compared to the previous one.

Overall, the mean trajectory of the $\overline{\text{FH-MDP}}$ agent, reported with a solid line in Figure 6, obtained a cumulative reward of 240, whereas the proposed agent reported in Figure 5 obtained an average cumulative reward of 336. The proposed agent obtains on average 96% of the highest possible reward equal to 350 - computed starting from favourable initial conditions and with a perfect crossing of the line $p(t) = 0$ at $t = 100$). The result of the proposed agent is even stronger considering that the reward 350 can not be achieved from arbitrary initial conditions (e.g., when the cart starts in an undesired position or the pole is significantly inclined).

The reason behind the performance difference between the two solutions is to be found in the different approximation of the value function that is available to the two agents: due to the state space aggregation, the $\overline{\text{FH-MDP}}$ performs only a single estimation of the value function over the entire episode length. On the contrary, the FH-MDP agent estimates the value function over all the time-steps. This is the reason why the proposed agent is able to correctly anticipate changes in the episode objective and in the future rewards.

To summarise, Figure 6 shows that standard DeepRL implementations are not able to cope with time-varying rewards unless, as proposed in section 2.2.1, the state space is augmented with the time-step.

4.2. Second Simulation: Stochastic Time-Varying Reward

4.2.1. Task Description

For this simulation, we considered the case in which the time-varying reward function has a stochastic behaviour, i.e.:

$$\rho(p(t), t) = \begin{cases} +1.5 & \text{if } t \leq 150 + \tau \text{ and } p(t) < 0 \\ +4 & \text{if } 150 + \tau < t \leq 200 \text{ and } p(t) > 0 \\ -0.5 & \text{otherwise} \end{cases}, \tag{48}$$

with τ distributed uniformly over $[0, 10]$ time-steps. This scenario represents a situation in which it is not possible to know *a priori* the switching instant of the control objective and hence an immediate partition as the one proposed in the first simulation is no longer possible. However, for the sake of presentation clarity, we assume that the controller knows that the objective does not change in the first 100s.

4.2.2. Controller Design

In this simulation, the control horizon is split in the two sets $\mathcal{T}_1 = [0, 100), \mathcal{T}_2 = [100, 200]$ and the problem is modeled as a partially aggregated FH-MDP (see Section 3.3), with an aggregated $\overline{\text{FH-MDP}}_1$ over the first period $\mathcal{T}_1$ (during which the reward function is invariant and depends on the visited state only) and an augmented FH-MDP$_2$ over the time period $\mathcal{T}_2$ (during which the reward function is time-varying).

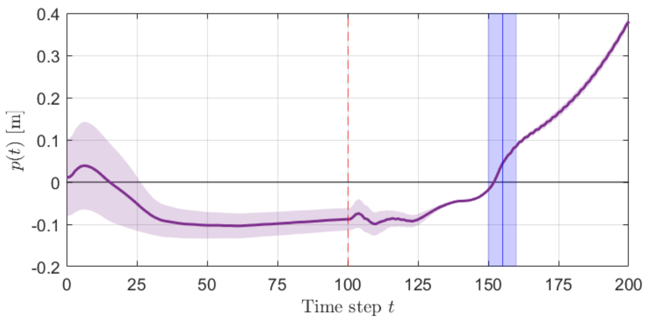

Figure 7. Simulation 2: Evolution of the cart position when controlled by the complete partially aggregated FH-MDP agent. Red dashed vertical line: switching instant between the two agents; blue vertical line and shaded band: expected task/objective switching instant and the region over which it is uniformly distributed.

4.2.3. Second Simulation results

Figure 7 reports the behaviour of the system when controlled by the two trained agents. In the figure, we can observe that the overall control system steers the cart-pole in the region $p(t) < 0$ for the first 150s, experiencing only minor fluctuations when the controlling agent switches from the $\overline{\text{FH-MDP}}_1$ one to the FH-MDP_2 one at time $t = 100$. Then, as shown in Figure 7, the controller drives the system to $p(t) > 0$ starting from $t = 152$ (with small standard deviation), quite close to the expected time-switching instant $t = 155$ of equation (48), losing about 1% of the cumulative reward compared to the perfect crossing case. This simulation demonstrated that it is not necessary to have exact information regarding the structure of the time-varying rewards to design a DeepRL agent according to the proposed framework. In fact, even an arbitrary partition, chosen for computational reasons as the presented one, allows the combined agent to correctly solve the stochastic control problem.

5. Conclusions and Future Works

This work presents a framework to address time-varying objectives in finite-horizon DeepRL, following a strategy inspired by switching control systems.

To enable the solution of problems in which the objective function changes with time, the proposed solution relies on the augmentation of the state space by including the visit time into the state definition. To overcome the scalability problem caused by the state augmentation, the paper shows that the Bellman's principle of optimality can be applied to divide the original task into a set of sub-tasks, each one defined over a time interval that is a subset of the episode duration and solved by a specific DeepRL agent. The overall task is then solved by a controller that uses in sequence the trained DeepRL agents, each one operating during its associated control interval.

The proposed solution does not affect the quality of the value function estimation and, on the contrary, it was shown to improve it in the cases in which the visiting time of the system states contains information useful to anticipate upcoming objective changes.

The paper reported two proof-of-concept simulations that demonstrated the capabilities of the proposed framework to solve time-varying tasks on a typical benchmarking control task for RL problems, comparing it to a standard DeepRL agent that does not include the current time-step in the system state.

Future research directions involve the inclusion of constrained problems in the formulation and the continuous learning setting.

6. declaration of interest statement

This work received no funding and there is no conflict of interest to declare.

References

Barto, A. G., Sutton, R. S., & Anderson, C. W. (1983, September). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-13*(5), 834–846.

Bellman, R. (1966, July). Dynamic programming. *Science*, *153*(3731), 34–37.

Bertsekas, D. P. (2005). *Dynamic programming and optimal control* (3rd ed., Vol. I). Belmont, MA, USA: Athena Scientific.

Borsa, D., Graepel, T., & Shawe-Taylor, J. (2016). Learning shared representations in multi-task reinforcement learning. Retrieved from `http://arxiv.org/abs/1603.02041`

Boyan, J. A., & Littman, M. L. (2001). Exact solutions to time-dependent mdps. In *Advances in neural information processing systems* (pp. 1026–1032).

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., & Tang, J. (2016). *Openai gym.* Retrieved from `https://arxiv.org/abs/1606.01540`

Choi, S. P., Yeung, D.-Y., & Zhang, N. L. (2000). An environment model for nonstationary reinforcement learning. In *Advances in neural information processing systems* (pp. 987–993).

Choi, S. P. M., Yeung, D.-Y., & Zhang, N. L. (2000). Hidden-mode markov decision processes for nonstationary sequential decision making. In *Sequence learning* (pp. 264–287). Springer Berlin Heidelberg.

Chun, T. Y., Lee, J. Y., Park, J. B., & Choi, Y. H. (2017, April). Adaptive dynamic programming for discrete-time linear quadratic regulation based on multirate generalised policy iteration. *International Journal of Control*, *91*(6), 1223–1240.

Florian, R. V. (2007). Correct equations for the dynamics of the cart-pole system. *Report for the Center for Cognitive and Neural Studies (Coneural).*

Forootani, A., Liuzza, D., Tipaldi, M., & Glielmo, L. (2019, November). Allocating resources via price management systems: a dynamic programming-based approach. *International Journal of Control*, 1–21.

Forootani, A., Tipaldi, M., Zarch, M. G., Liuzza, D., & Glielmo, L. (2019, September). Modelling and solving resource allocation problems via a dynamic programming approach. *International Journal of Control*, 1–12.

Fujimoto, S., Hoof, H., & Meger, D. (2018). Addressing function approximation error in actor-critic methods. In *International conference on machine learning ICML* (pp. 1582–1591).

Fujimoto, S., Hoof, H., & Meger, D. (2020). *Author's PyTorch implementation of TD3 for OpenAI gym tasks.* Retrieved from `https://github.com/sfujim/TD3`

Gábor, Z., Kalmár, Z., & Szepesvári, C. (1998). Multi-criteria reinforcement learning. In *Proceedings of the international conference on machine learning ICML* (Vol. 98, pp. 197–205).

Giuseppi, A., & Pietrabissa, A. (2020, July). Chance-constrained control with lexicographic deep reinforcement learning. *IEEE Control Systems Letters*, *4*(3), 755–760.

Grondman, I., Busoniu, L., Lopes, G. A. D., & Babuska, R. (2012, November). A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *42*(6), 1291–1307.

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). *Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor.* Retrieved from `https://arxiv.org/abs/1801.01290`

Hallak, A., Castro, D. D., & Mannor, S. (2015). *Contextual markov decision processes.* Retrieved from `https://arxiv.org/abs/1502.02259`

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996, May). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, *4*, 237–285.

Kelly, S., & Heywood, M. I. (2018, September). Emergent solutions to high-dimensional multitask reinforcement learning. *Evolutionary Computation*, *26*(3), 347–380.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., & Silver, D. (2015). *Continuous control with deep reinforcement learning.* Retrieved from `https://arxiv.org/abs/1509.02971`

Liu, C., Xu, X., & Hu, D. (2015, March). Multiobjective reinforcement learning: A comprehensive overview. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *45*(3), 385–398.

Liu, L., & Sukhatme, G. S. (2018, July). A solution to time-varying markov decision processes. *IEEE Robotics and Automation Letters*, *3*(3), 1631–1638.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., & Wierstra, D. (2013). *Playing atari with deep reinforcement learning.* Retrieved from `https://arxiv.org/abs/1312.5602`

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., . . . Hassabis, D. (2015, February). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.

Omidshafiei, S., Pazis, J., Amato, C., How, J. P., & Vian, J. (2017). Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *Proceedings of the 34th international conference on machine learning ICML* (Vol. 70, pp. 2681–2690).

Padakandla, S., J, P. K., & Bhatnagar, S. (2019). Reinforcement learning in non-stationary environments. Applied Intelligence 2020. Retrieved from `arXiv:1905.03970`

Puterman, M. L. (1994). *Markov decision processes: Discrete stochastic dynamic programming.* Wiley.

Rachelson, E., Fabiani, P., & Garcia, F. (2009, November). TiMDPpoly: An improved method for solving time-dependent MDPs. In *2009 21st IEEE international conference on tools with artificial intelligence.* IEEE.

Shimkin, N. (2011). Learning in complex systems, lecture notes. Retrieved from `https://shimkin.net.technion.ac.il/courses/learning-in-complex-systems-2011/`

Sutton, R. S., & Barto, A. G. (1998). Reinforcement learning: An introduction.

Tanaka, F., & Yamamura, M. (2003). Multitask reinforcement learning on the distribution of MDPs. In *Proceedings 2003 IEEE international symposium on computational intelligence in robotics and automation. computational intelligence in* *robotics and automation for the new millennium (cat. no.03ex694).* IEEE.

Teh, Y., Bapst, V., Czarnecki, W. M., Quan, J., Kirkpatrick, J., Hadsell, R., ... Pascanu, R. (2017). Distral: Robust multitask reinforcement learning. In *Advances in neural information processing systems 30* (pp. 4496–4506). Curran Associates, Inc.

Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double Q-learning. In *Thirtieth aaai conference on artificial intelligence.*

Van Moffaert, K., & Nowé, A. (2014). Multi-objective reinforcement learning using sets of Pareto dominating policies. *The Journal of Machine Learning Research*, *15*(1), 3483–3512.

Verme, M. D., da Silva, B. C., & Baldassarre, G. (2020). *Optimal options for multi-task reinforcement learning under time constraints.* Retrieved from `https://arxiv.org/abs/2001.01620`

Wang, Y., Chakrabarty, A., Zhou, M., & Zhang, J. (2019). Near-optimal control of motor drives via approximate dynamic programming. In *2019 ieee international conference on systems, man and cybernetics (smc)* (p. 3679-3686).

Wilson, A., Fern, A., Ray, S., & Tadepalli, P. (2007). Multi-task reinforcement learning: A hierarchical bayesian approach. In *Proceedings of the 24th international conference on machine learning ICML* (p. 1015–1022). Association for Computing Machinery.

Xu, X., Lian, C., Zuo, L., & He, H. (2014). Kernel-based approximate dynamic programming for real-time online learning control: An experimental study. *IEEE Transactions on Control Systems Technology*, *22*(1), 146-156.

Zhang, H., Cui, L., Zhang, X., & Luo, Y. (2011). Data-driven robust approximate optimal tracking control for unknown general nonlinear systems using adaptive dynamic programming method. *IEEE Transactions on Neural Networks*, *22*(12), 2226-2236.

Zhang, H., Wei, Q., & Liu, D. (2011). An iterative adaptive dynamic programming method for solving a class of nonlinear zero-sum differential games. *Automatica*, *47*(1), 207 - 214.

ANNEX

Figures 8 and 9 shows a simple example of FH-MDP modeled by an augmented FH-MDP and by an aggregate $\overline{\text{FH-MDP}}$, respectively.

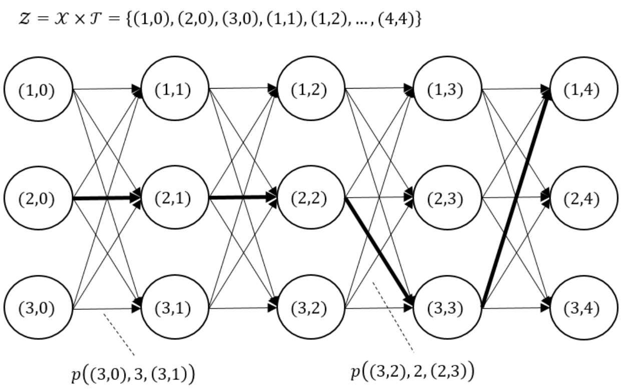

Figure 8. Example of augmented discrete state space $\mathcal{Z}$, with $\mathcal{X} = \{1,2,3\}, \mathcal{U} = \{1,2,3\}$, $T = 4$, $g([x,t],u) = [u,t+1]$ and with transitions probabilities $p([x_t,t],u_t,[x_{t+1},t+1])$. Bold transitions characterize a possible episode path starting from the initial state $z_0 = [2,0]$, passing via the states $z_1 = [2,1]$, $z_2 = [2,2]$ and $z_3 = [3,3]$ and with final state $z_T = z_4 = [1,4]$. For simplicity, let the reward function be function of the next state only, i.e., $\rho([x,t],u_t,[x_{t+1},t]) = \rho(x_{t+1})$, and the terminal reward function of the terminal state only, i.e., $\rho^f([x,T],u_T) = \rho^f(x)$. The reward-to-go of a state x depends on the visit time but the one of an augmented state z is stationary: for example, the reward-to-go of $x = 2$ at time $t = 0$, i.e., of the augmented state $[2,0]$, is equal to the reward gained by the system along the path: $\rho(2) + \gamma\rho(2) + \gamma^2\rho(3) + \gamma^3\rho^f(1)$; the reward-to-go of $x = 2$ at times $t = 1$ and $t = 2$, i.e., of the augmented states $[2,1]$ and $[2,2]$, are equal to $\rho(2) + \gamma\rho(3) + \gamma^2\rho^f(1)$ and $\rho(1) + \gamma\rho^f(1)$, respectively.

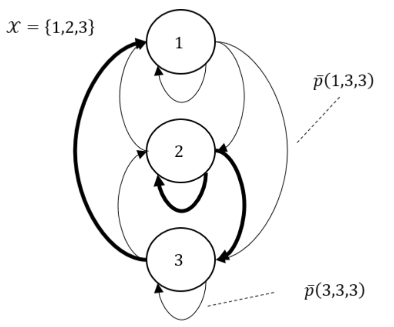

Figure 9. Same problem of Fig. 8 with aggregate discrete state space $\mathcal{X} = \{1,2,3\}$, with $h(x,u) = u$ and with transition probabilities $\bar{p}(x_t,u_t,x_{t+1})$. Bold transitions characterize the same episode path of Fig. 8, which, in $T = 4$ time-steps, starts from the initial state $x_0 = 2$, passes via the states $x_1 = 2$, $x_2 = 2$ and $x_3 = 3$ and ends in the final state $x_4 = 1$. Under the same reward assumption of Fig. 8, the reward-to-go of a state x depends on the visit time. For example, the reward gained by the system along the path starting from state 2 at time $t = 0$ is $\rho(2) + \gamma\rho(2) + \gamma^2\rho(3) + \gamma^3\rho^f(1)$; the rewards-to-go in the same state $x = 2$ visited at time $t = 1$ and $t = 2$ are $\rho(2) + \gamma\rho(3) + \gamma^2\rho^f(1)$ and $\rho(3) + \gamma\rho^f(1)$, respectively.