Information Sciences 591 (2022) 365–380

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Actor-critic continuous state reinforcement learning for wind-turbine control robust optimization

Borja Fernandez-Gauna [a,b,*], Manuel Graña [a], Juan-Luis Osa-Amilibia [b], Xabier Larrucea [b]

[a] Computational Intelligence Group, University of the Basque Country, San Sebastian, Spain
[b] Alava Polytechnic Engineering School, University of the Basque Country (UPV/EHU), Vitoria, Spain

ARTICLE INFO

Article history:
Received 31 December 2019
Received in revised form 17 January 2022
Accepted 22 January 2022
Available online 29 January 2022

Keywords:
Actor critic reinforcement leraning
Varying speed wind turbine control

ABSTRACT

The control of Variable-Speed Wind-Turbines (VSWT) extracting electrical power from the wind kinetic energy are composed of subsystems that need to be controlled jointly, namely the blade pitch and the generator torque controllers. Previous state of the art approaches decompose the joint control problem into independent control subproblems, each with its own control subgoal, carrying out separately the design and tuning of a parameterized controller for each subproblem. Such approaches neglect interactions among subsystems which can introduce significant effects. This paper applies Actor-Critic Reinforcement Learning (ACRL) for the joint control problem as a whole, carrying out the simultaneous control parameter optimization of both subsystems without neglecting their interactions, aiming for a globally optimal control of the whole system. The innovative control architecture uses an augmented input space so that the parameters can be fine-tuned for each working condition. Validation results conducted on simulation experiments using the state-of-the-art OpenFAST simulator show a significant efficiency improvement relative to the best state of the art controllers used as benchmarks, up to a 22% improvement in the average power error performance after ACRL training.

1. Introduction

Onshore wind energy has become one of the most competitive renewable alternatives to traditional fossil-based energy sources [1] because it has the lowest operational costs and environmental footprints among the renewable energy sources. Hence, the growing number of Variable-Speed Wind-Turbines (VSWT) deployed worldwide. Designing an efficient VSWT controller is challenging because of the complex dynamics induced by the wind variability, and the large number of variables to be controlled simultaneously. Most of the research work on VSWT control [2–7] has addressed these challenges in two ways:

a) Using simplified dynamic models of VSWT operation that allow easy derivation of analytical expressions instead of detailed, complex, and non-linear dynamical models that
b) Decomposing the control problem into subtasks where each subsystem local control goal contributes toward the global control goal.

Departing from these conventional approaches, this paper provides two alternative approaches:

* Corresponding author.

https://doi.org/10.1016/j.ins.2022.01.047
0020-0255/

a) Validating the control models over highly accurate simulations using OpenFAST [8], which is the state-of-the-art open source simulation suite for VSWT. To the best of our knowledge, this is the first time a RL-based approach has been validated on such numerically accurate VSWT simulator despite being extremely demanding computationally.
b) Optimizing the VSWT control parameters as a whole according to the global control objective formulated as "maximize the power extraction efficiency".

Reinforcement Learning (RL) allows to learn the parameters of a decision policy with minimal input from the designer and without domain-specific knowledge in the form of labelled samples, which makes RL a promising research tool to develop VSWT control systems. This paper presents a novel approach that optimizes the parameters of VSWT baseline controllers using Actor-Critic RL (ACRL) as a function of additional system variables not used by the baseline controller. This approach augments the input space of the learned controller decoupling it from the simplified model baseline control. Thus, it is possible to carry out fine-tuning of the parameters for different working conditions while preserving the modular structure of the controller.

This paper reports results of *proof-of-concept* experiments on two baseline controllers that will be denoted accross the paper by the name of their respective authors, i.e. *Vidal* [9], and *Boukhezzar* [10]. Accordingly, we will denote the optimized controllers obtained after ACRL training as *Vidal** and *Boukhezzar**. The experiments have two phases: The first phase is a grid-search on the baseline controller parameter space in order to select the best performing parameter values for each baseline controller. The second phase carries out the ACRL training using the results of the grid-search as the initial condition. Experimental results show a significant performance improvement over both baseline controllers after ACRL training.

Summarizing, the contributions of this paper are as follows:

- We carry out the overall system control design, without decomposing the problem into subproblems. The optimization problem is tackled as a whole, taking into account interactions that are neglected in other approaches.
- We define an augmented input space for the ACRL training of the controller, which is different from the input space of the baseline controller, allowing more flexibility in the definition of operational conditions and experiments.
- We propose a global measure of the VSWT efficiency, namely the power extraction efficiency, which is the global reward function in the ACRL process.
- Up to the best of our knowledge, this is the first time a RL-based approach has been validated over a high quality dynamically accurate (and computationally expensive) VSWT simulator like OpenFAST.
- The experimental design provides a comprehensive exploration of the ACRL metaparameters space.

The outline of the paper is as follows. Section 2 reviews the state of the art of VSWT control approaches. Section 3 provides relevant RL background. Section 4 describes the proposed approach. Section 5 reports experimental settings and results. Section 6 provides conclusions and further work ideas.

2. Variable-Speed Wind-Turbines control state of the art

In the last years, a great scientific effort has been made toward finding alternatives to fossil fuels with a lower environmental impact. One of the most promising renewable energy sources is wind energy. Wind-Turbines (WT) transform the wind kinetic energy into electrical energy with the smallest known environmental impact [11]. Furthermore, wind energy is available in such quantities that it could potentially replace all the other sources still providing many times the energy consumed by the whole world population [12]. However, WT are not able to harvest 100% of the energy passing through the disk area drawn by the WT blades.Only a fraction according to the following expression is converted into electrical power:

$$P_a = \frac{1}{2}\rho \cdot \pi \cdot R^2 \cdot v^3 \cdot C_p, \tag{1}$$

where ρ is the air density, R is the radius of the disk area drawn by the blades, v is the wind-speed, and C_p is the power coefficient of the WT. This coefficient is a unimodal function of the tip speed ratio λ, with $\lambda = \frac{\omega_r \cdot R}{v}$, where ω_r is the rotor's angular speed. A gearbox transmits the speed of the rotor to the generator $n_g = \frac{\omega_g}{\omega_r}$, where n_g is gearbox ratio, and ω_g is the generator's angular speed.

There are two types of WT: Fixed-Speed WT (FSWT) and Variable-Speed WT (VSWT). FSWT operate at a fixed rotor speed, hence, since ω is constant, the power coefficient C_p only changes with v. Thus, FSWT can only reach maximum efficiency (maximum C_p) for a specific wind speed. VSWT on the other hand, can change the rotor speed to operate at maximum C_p for a broad range of wind speeds. For this reason, VSWT is more widespread than FSWT. The downside is that the control of VSWT is more complicated because the controller must set both the blade pitch β and the generator torque T_g with a single control objective: maximizing the power extraction efficiency.

2.1. Baseline controllers

State of the art aproaches [13,10,9] tackle VSWT control by decomposing the problem into two separate control subtasks:

- control the blade pitch β to track a reference value for the generator speed ω_{ref} that depends on v, i.e. the blade pitch controller aims at minimizing the absolute generator speed error $e_{\omega_g} = |\omega_g - \omega_{ref}|$, and
- control the torque T_g to reduce the absolute power error $e_p = |P_e - P_{ref}|$, where P_e is the electrical power generated and P_{ref} is the reference value of electrical power to be achieved (which also depends on v).

These two subtasks are usually tackled by a proportional-integral-derivative (PID) feedback controller that aims to minimize the error variable e, whose parameters are denoted by k_i, $i = 1 \dots n$. In the conventional approaches, these parameters are usually set by manual tuning or using a heuristic method [14], and they are kept constant during the entire life-cycle of the system. The baseline controllers *Vidal* [9] and *Boukhezzar* [10] are both analytically derived from a simplified one-mass dynamic model of a VSWT [10,4,3,5,6].

The *Vidal* model is specified by the following formulae:

$$\dot{T_g} = \frac{1}{\omega_g}\left[-T_g\left(A \cdot \omega_g + \dot{\omega_g}\right) + A \cdot P_{ref} + K_\alpha \cdot sgn(e_p)\right], \tag{2}$$

$$\beta = \frac{1}{2} K_p e_{\omega_g} \cdot \left(1 + sgn(e_{\omega_g})\right) + K_{it} e_{\omega_g}, \tag{3}$$

where A, K_α, K_p and K_i are the controller parameters.

The *Boukhezzar* model is defined by the following two expressions:

$$\dot{T_g} = \frac{1}{\omega_g}\left[C_o e_p - \frac{1}{J_t}\left(T_a T_g - K_t \omega_g T_g - T_g^2\right)\right], \tag{4}$$

$$\beta = K_p e_{\omega_g} + K_{it} e_{\omega_g}, \tag{5}$$

where C_0, K_p and K_i are the controller parameters, J_t, K_t are parameters of the VSWT one-mass model and T_a is the aerodynamic torque on the rotor blades.

2.2. Alternative control design approaches

In the literature, the modeling of the VSWT dynamics is simplified in order to achieve an analytical derivation of the controllers [15], or to carry out simulations in affordable time. Examples of simplifications are the use of the two mass system as a surrogate of the VSWT dynamics in [16], or the linearization of the VSWT model using extended Kalman filter [17] in order to select the operation regime that must be feed to the PID controller.

Alternatives to PID controller design have been approached from diverse points of view, like the insertion of active tendons inside the blades to achive reduction of vibrations due to wind turbulence [18]. The use of fuzzy modelling of control command generation for blade pitch control [19] requires an *a priori definition of the response strategy and the identification of operating regimes that should be detected by a discrete event supervision subsystem. Fuzzy Mamdani 's inference method has been demonstrated for improved pitch control, requiring manual construction of the fuzzy rules [20]. A sensorless method based on the readings of voltage and current at the output of the generator was developed to avoid WT overloading [21]. The modeling of nonlinear slide mode controller allows to cope with high frequency oscillations [22], and for large scale WT power optimization without wind speed sensors [23], while Artificial Neural Networks have been also reported to assist in the speed control of the switched reluctance generator driven by an VSWT [24]. Previous works reporting results of the application of RL to VSWT control design [25–27] have been limited to learn one of the control subtasks according to its local control goal, testing the controller on simplified VSWT models to reduce the computational costs. Moreover, these approaches have used the same input variables for the controller and the learner. The approach proposed in this paperdeparts from such limitations, learning the control of the entire VSWT system and carrying out the validation on high quality dynamic simulations.*

3. Reinforcement learning

The *trial-and-error* interaction between an agent and its environment is modeled in the RL framework as a Markov Decision Processes $\langle S, A, T, R\rangle$, where S is the set of observable variables defining the system state space, A is the set of actions that the agent can take, $T : S \times A \times S \rightarrow [0, 1]$ is a stochastic transition function that gives the probability of observing state $s\prime$ after executing action a in state s, and R is a reward function assessing the value of the outcome of a transition $\langle s, a, s\prime\rangle$has been. In control applications, the agent's goal is to learn a deterministic policy $\pi(s)$ that maximizes the *value function* $V(s)$, that is defined as the accumulated discounted rewards to be expected if the agent follows deterministic policy $\pi(s)$:

$$V^{\pi}(s) = E\left\{\sum_{k=0}^{\infty} r_{t+k+1} \cdot \gamma^t | s_t = s\right\}, \tag{6}$$

where γ is the discount factor weighting immediate and future rewards.

In Actor-Critic RL (ACRL) methods the agent has two components: the Actor whose task is to learn an optimal deterministic policy $\pi^*(s)$ from the interaction with the environment, and the Critic whose task is to estimate the value $V^{\pi_t}(s)$ of the Actor current deterministic policy $\pi_t(s)$ at time t. Because the deterministic function that implements the policy and the function that computesits value are modelled independently, the ACRL architecture allows to formulate and learn continuous valued policies. This is a key feature for most real-world control problems, thus ACRL methods dealing with continuous control problems have become widespread [28,29]. The main ACRL iteration loop follows a simple procedure: a) the Actor observes the state s and selects an action a applying its deterministic policy $a = \pi(s)$. b) The Critic, after observing the resulting state $s\prime$ and received reward r_t, proceeds to update its estimate of $V^{\pi_t}(s)$, and to give the Actor a critique δ_t. Finally, c) the Actor updates its deterministic policy π_{t+1} according to the critique. In this paper, the critique δ_t is computed following the *Temporal-Difference error* formulation

$$\delta_t = r_t + \gamma \cdot V^{\pi_t}(s\prime) - V^{\pi_t}(s), \tag{7}$$

assessing whether the last selected action improved over the expected value of the deterministic policy.

3.1. Function approximation

Real-world control problems [30] often have continuous state and action spaces, which are best modelled using Value Function Approximation (VFA) to represent the action value maps as parameterized functions instead of lookup tables [31]. There are two main types of VFAs: linear and non-linear. Non-linear VFAs and Deep Reinforcement Learning (Deep RL) [32] have been key to recent RL advances and thus, have received a lot of attention from the scientific community. Although they can approximate more complex high-dimensional functions, non-linear VFAs are very expensive computationally and not suitable for real-time adaptation. On the other hand, linear VFAs require much less computation and have better understood theoretical properties than non-linear VFAs [33], so that the effect of parameter variations are easier to understand. Additionally, faster function evaluation and parameter update make linear VFA *a priori better suited to real-time control problems, such as VSWT control.*

In this paper, the approximating function is as follows:

$$f(x) = \vec{\theta}^T \vec{\phi}(x), \tag{8}$$

where $\vec{\theta}$ is the vector of parameters learned by the agent, and $\phi : X \rightarrow \Phi$ is a feature extraction function mapping the input space X (i.e, the state space S) to a feature vector $\vec{\phi}(x) = [\phi_1 \dots \phi_m]$ in feature space Φ, following a kernel transformation approach [34]. The experiments in this paper have used Gaussian Radial Basis Functions (GRBF) as feature extraction kernels:

$$\phi_i(x) = \exp^{-\frac{\|x - c_i\|^2}{2\sigma^2}}, \tag{9}$$

where i is the feature index, c_i and σ (which is the same for all GRBFs) are the parameters of the Gaussian kernel. Kernel parameters are initialized by a grid search. For simplicity, this expression assumes that x is a real-valued scalar variable, but it can be extended to vector states using a separate grid search for each state variable.

3.2. Exploration

The Actor deterministic policy can only be improved if the agent is allowed to explore alternative policies. When learning a continuous valued policy function, exploration can be implemented in two ways: either by modeling the Actor policy during the learning process as a stochastic process $\pi : S \times A \rightarrow [0, 1]$ [35], or by adding a perturbation signal to the output of the deterministic policy computation. In the former case, the actual action is generated by a stochastic sampling process where the exploration takes place. From a systems control perspective, the latter approach makes more sense because control commands should be generated univocally for each system state, hence it has been adopted in the computational experiments. To avoid biasing the exploration, the perturbation signal must follow a stochastic process whose distribution has zero mean and is symmetric around the mean. Specifically, in this paper it follows an *Ornstein-Ühlenbeck* process $O(t)$ [36], which is a symmetric mean-reverting process widely used to implement the exploration in control applications of RL [32] because it produces time correlated perturbation samples that result in a more natural exploratory behavior. The *Ornstein-Ühlenbeck* stochastic process is defined by the following stochastic differential equation:

$$d\eta_t = -\rho\eta_t dt + \sigma dW_t, \tag{10}$$

where parameters ρ and σ control how far the perturbation signal departs from its mean value and how fast it returns to it, respectively. W_t is a Wiener process [37]. The following formula provides the discrete realization of the process:

$$\eta_{t+1} = \eta_t - \rho\eta_t\Delta t + \sigma\sqrt{\Delta t}N(0,1). \tag{11}$$

Specifically, the perturbation signal used in the experiments is defined as

$$O(t) = \eta_t \cdot \psi, \tag{12}$$

where ψ is the scaling factor modulating the amplitude of the perturbation signal without changing its statistical properties. This perturbation signal is added to the Actor policy to generate the action executed at each time-step t:

$$a_t = \pi_t + O(t) \tag{13}$$

3.3. Learning rules

The ACRL approach to VSWT control design is implemented by a Continuous Actor Critic Learning Automaton *(CACLA)* [38]. The Critic is implemented as a *Temporal-Difference* (TD) learning rule [28]defined as follows:

$$\vec{w}_{t+1} \leftarrow \vec{w}_t + \alpha^c \cdot \delta_t \cdot \nabla V_t(s_t), \tag{14}$$

where $\overrightarrow{w_t}^c$ is the VFA weight learned by the Critic at time t, and α^c is the learning rate that can be annealed for improved convergence. In the case of linear VFA, the rule reduces to

$$\vec{w}_{t+1} \leftarrow \vec{w}_t + \alpha^c \cdot \delta_t \cdot \phi^c(s_t), \tag{15}$$

where $\phi^c(s_t)$ denotes the Critic kernel. The Actor parameter learning rule is a gradient descent algorithm that updates the policy function parameters only when a value improvement is observed:

$$\vec{\theta}_{t+1} \leftarrow \vec{\theta}_t + \alpha^a \cdot \left(a_t - \pi_t\left(s_t;\vec{\theta}_t\right)\right) \cdot \nabla\pi_t\left(s_t;\vec{\theta}_t\right) \quad \textit{if } \delta_t > 0, \tag{16}$$

where θ^a is the weight vector of the policy linear approximation function learned by the Actor, a_t is the action taken by the agent at time-step t, and α^a is the Actor's learning rate. The linear approximation rule used in the experiments is as follows:

$$\vec{\theta}_{t+1} \leftarrow \vec{\theta}_t + \alpha^a \cdot \left(a_t - \pi_t\left(s_t;\vec{\theta}_t\right)\right) \cdot \phi^a(s_t) \quad \textit{if } \delta_t > 0, \tag{17}$$

where $\phi^a(s_t)$ is the Actor kernel. Moreover, if there is no exploration (i.e. $a_t = \pi(s_t)$) then the learning rule stops to work, because the Actor will become purely deterministic. This is contrary to conventional policy gradient methods that would update the policy in the opposite direction $(2\pi_t(s_t) - a_t)$ when the evaluation of the action by the Critic results in a negative TD δ_t value. Therefore, CACLA doesn't assume that $2\pi_t(s_t) - a_t$ may have a value greater than a_t, because when $\delta_t < 0$ there is no evidence that gradient descent will improve the value of the policy. CACLA learning strategy avoids fluctuations of the policy [38]. The argument is that adaptation when $\delta_t > 0$ follows an evidence of improvement, while adaptation when $\delta_t < 0$ does not ensure improving future values of the policy, thus inviting dynamic oscillations.

In order to decide which Actor learning procedure is most appropriate for our problem, we have carried out preliminary experiments comparing CACLA with Deep-CACLA [29], deep Q-network (DQN) [39] (in this approachwe explored two learning rates 10^{-4} and 10^{-5}), and deep deterministic policy gradient (DDPG) [40] (in this approach we explored two learning rates 10^{-4} and 10^{-5} for the critic and 10^{-5} and 10^{-6} for the actor). We allowed 50 training episodes to each approach.As explained in Section 5.5 below, each RL experiment is time consuming due to the VSWT dynamical simulation, so these exploratory experiments were not exhaustive. Table 1 contains the best results for each approach, and the computation time of each approach relative to that of CACLA. CACLA achieve the greater average reward in both controller models *Vidal** and *Boukhezzar**, besides CACLA computational time requirements is several orders of magnitude smaller than that of the deep RL approaches, as shown in Table 1. One reason for this significant difference lies in the specifics of the initialization of CACLA that can not be used by the deep RL architectures.

3.4. Convergence issues

There are no theoretical results that ensure general convergence of both CACLA and the competing deep RL approaches, hence much of the work follows heuristic guidelines. The overall process is a strongly non-linear optimization process. Often, the best we can obtain is a good suboptimal parameter setting, close enough to the global optima to be useful in practice. To ensure the convergence of the RL, the adequate management of the exploration is crucial. While exploration is active, the RL process is allowed to jump between local optima attractors. Once exploration isswitched off, the RL optmization process remains in the attractor of a, hopefully, good suboptimal solution. Another critical issue for convergence is the initialization of the agent. To speed up convergence, we have initialized the modules to the best PID parameter settings achieved by conventional procedures.

The CACLA is an on-policy approach that updates the parameters of the policy applying actions selected by the policy being learned. On the other hand, off-policy algorithms are able to learn from experience tuples where the actions are drawn from any other policy [31]. The literature suggests that on-policy approaches are more unstable with longer convergence

Table 1
Comparative performance of CACLA and deep RL alternatives on preliminary experiments, measured by the achieved average reward, and by the relative computational cost (CACLA computation time is the unit).

	Vidal*	Boukhezzar*	Vidal*	Boukhezzar*
	Average reward		Computational cost	
CACLA	0.821	0.956	1	1
Deep-CACLA	0.652	0.768	33.65	24.2
DQN	0.126	0.208	134.77	98.33
DDPG	0.390	0.130	143.84	115.45

times but more adaptable to changing environments than off-line approaches [41–43]. In the [44] seminal paper on achieving human level performance by RL, two techniques were reported to improve convergence of on-policy RL algorithms, namely *experience replay* and *target function freezing*.Algorithm 1 specifies the CACLA implementation and the application of these two features. Experience replay uses a fixed-size buffer to store experience tuples $E = \{\langle s, a, s\prime, r\rangle\}$, where new tuples are introduced at random preserving the maximum allowed buffer size. The experience buffer is used in step 7 of the algorithm to compare the actual action generated applying the current policy against the best valued stored action.Experience replay reduces the temporal correlation between tuples improving the learning performance. Target function freezing is a technique that evolves two sets of parameters of the Actor function learned: the *online weights* that are updated every time step, and the *target weights*. The latter are a copy of the online weights updated every m time steps , which stays frozen during this period. Online weights are the ones actually used to generate the on-policy action decision.

Algorithm 1 An episode of the CACLA enriched with experience replay and target function freezing.

Notation:card() is the cardinality of the set, mod() is the modulus operation. N_e maximum size of the set of experience E. m is the interval between renewal of the frozen target function parameters. $\vec{\theta}^f$ and $\vec{\theta}_t$ are the frozen and current parameters of the Actor's policy functional approximation $\pi_t\left(s_t;\vec{\theta}\right)$. $\vec{w}_t$ are the parameters of the current Critic's state value function approxmation $V^{\pi_t}(s_t;\vec{w}_t)$. $O(t)$ is an*Ornstein-Ühlenbeck* process.

Input: Initial parameter configurations, $\vec{\theta}^f = \vec{\theta_0}, \vec{w}_0$, s_0, and specific settings of N_e, and m

For t in $\{1,\ldots,t_{max}\}$

1: Observe current state s_t, if s_t is a final state then exit.

2: The Actor computes the next action $a_t = \pi_t\left(s_t;\vec{\theta}_t\right) + O(t)$

3: Applying action a_t the system trasfers to state s_{t+1} and receives reward r_{t+1}

4: Include $e = \langle s_t, a_t, r_{t+1}, s_{t+1}\rangle$in the set of experience E with probability p_e

(a) if card$(E) > N_e$ then remove older e_t from E until card$(E) = N_e$

5: Compute the temporal difference error $\delta_t = r_{t+1} + \gamma \cdot V^{\pi_t}(s_{t+1};\vec{w}_t) - V^{\pi_t}(s_t;\vec{w}_t)$

6: Update Critic weights $\vec{w}_{t+1} \leftarrow \vec{w}_t + \alpha^c \cdot \delta_t \cdot \phi^a(s_t)$,

7: Update Actor policy $\vec{\theta}_{t+1} \leftarrow \vec{\theta}_t + \alpha^a \cdot \left(a_t - \arg\max_{V(s_t\in E)}\left\{a \in E \middle| a = \pi\left(s_t;\vec{\theta}^f\right)\right\}\right) \cdot \phi^a(s_t) \quad$ *if* $\delta_t > 0$,

8: if mod$(t,m) = 0$ then $\vec{\theta}^f = \vec{\theta}_t$

4. Actor-critic VSWT controller optimization

In this section, we formulate our proposed architecture to learn simultaneously the parameters of the blade pitch β, and the generator torque T_g controllers. A single reward signal combines the response of both subsystems, looking for the optimal performance of the whole VSWT. The overall control scheme is depicted in Fig. 1: the Actor learns the parameters of a baseline controller that maximize the expected future rewards estimated by the Critic. This architecture differs from previous published approaches by using different state subspaces for the Actor-Critic learner ($s_l \in S_l$), and the baseline controller ($s_c \in S_c$), therefore the complete observable state space is composed of both input subspaces $S = S_l \cup S_c$. The rationale supporting this approach is that the baseline controller may have different optimal parameter values in different operational state subspaces corresponding to different working conditions. In the experiments, variables v and T_g characterize the working condition.

4.1. Parameter space and state space

The Actor-Critic learns both the weight vector $\vec{\theta}^c$ corresponding to linear functional approximation of the current policy $V^{\vec{\theta}}(s_l)$, and different parameter vectors corresponding to each baseline controller. For clarity, in Fig. 1 we have grouped these parameter vectors into two sets of weights:

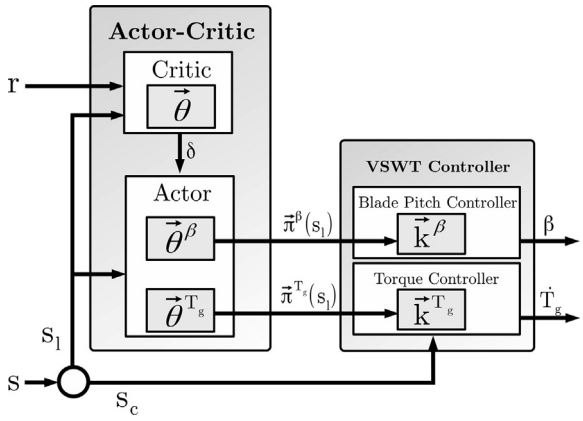


Fig. 1. Diagram of the proposed baseline controller parameter optimization architecture based on Actor-Critic RL.

- $\vec{\theta}^{\beta}$ denotes the vector of parameters for the linear functional approximation of the blade pitch β controller, and
- $\vec{\theta}^{T_g}$ denotes the vector of parameters for the linear functional approximation of the generator torque T_g controller.

Therefore, the Actor generates two vector valued policies corresponding to the parameters of each subsystem controller: $\vec{\pi}^{\beta}(s_l)$ and $\vec{\pi}^{T_g}(s_l)$. The dimension of each of these policies is equal to the number of parameters of the target controller. For each subsystem policy generated, the Actor uses a different perturbation signal, which must be calibrated independently with an appropriate amplitude factor. This perturbation amplitude depends on the value range of the baseline controller parameter. Using continuous VFA, the parameters of each baseline controller will change smoothly over time while it will be keeping its topology. The output of the overall controller system is the aggregation of the outputs of the baseline controllers, namely the blade pitch β and the generator torque rate $\dot{T}_g$, which are a function of the state variables in s_c and s_l. That means that we effectively extend the baseline controller $\beta, \dot{T}_g : S_c \rightarrow \mathbb{R}$ to a controller with augmented input space $\beta, \dot{T}_g : S \rightarrow \mathbb{R}$.

4.2. ACRL initialization

For a successful and time efficient ACRL training convergence, it is very important to initialize the Actor policies appropriately. Starting from random initialization produces huge waste of computation time, compromising convergence [45–48]. The proposed architecture allows to use heuristic parameter values $\hat{k}_i^{\beta}$, $i = 1 \dots n$ and $\hat{k}_i^{T_g}$, $i = 1 \dots m$ to initialize the Actor policy weights ($\vec{\theta}^{\beta}$ and $\vec{\theta}^{T_g}$, respectively). Assuming a Linear VFA $f(s) = \sum \vec{\theta} \cdot \vec{\phi}(s)$ with normalized activation factors, i.e. $\sum \phi(s) = 1$, each baseline policy weights is initialized as follows:

$$\vec{\theta}_0^{\beta} = \left[\hat{k}_1^{\beta} \dots \hat{k}_1^{\beta}, \dots, \hat{k}_n^{\beta} \dots \hat{k}_n^{\beta}\right], \tag{18}$$

$$\vec{\theta}_0^{T_g} = \left[\hat{k}_1^{T_g} \dots \hat{k}_1^{T_g}, \dots, \hat{k}_m^{T_g} \dots \hat{k}_m^{T_g}\right]. \tag{19}$$

This procedure sets the best parametrization of the baseline controller as the starting point of the learning process. After the initialization procedure, the following will hold for any state $s_l \in S_l$:

$$\vec{\pi}_0^{\beta}(s_l) = \left[\hat{k}_1^{\beta} \dots \hat{k}_n^{\beta}\right], \tag{20}$$

$$\vec{\pi}_0^{T_g}(s_l) = \left[\hat{k}_1^{T_g} \dots \hat{k}_m^{T_g}\right], \tag{21}$$

which means that, initially:

$$\beta\left(s_c, \vec{\pi}_0^{\beta}(s_l)\right) = \beta\left(s_c, \hat{k}_1^{\beta} \dots \hat{k}_n^{\beta}\right), \tag{22}$$

$$\dot{T}_g\left(s_c, \vec{\pi}_0^{T_g}(s_l)\right) = \beta\left(s_c, \hat{k}_1^{T_g} \dots \hat{k}_n^{T_g}\right). \tag{23}$$

5. Experiments

To validate our approach, we have conducted a series of simulation experiments to compare the two state of the art baseline controllers, denoted *Vidal* and *Boukhezzar*, against their ACRL optimized versions, denoted *Vidal** and *Boukhezzar**,

respectively. In this section, we report the settings used in our experiments and the results we obtained in the simulations. Following the standard in VSWT literature, we measure controller performance by the *absolute power error* (in Watts):

$$E_P = \int_t |P_e - P_n|. \tag{24}$$

Following the RL standard scheduling, the interaction between the controllers/agents and the simulated VSWT was structured in episodes, each episode consisting of 300 s simulated time.

5.1. Experimental design

The experimental work was divided into two phases:

- **Heuristic parameter tuning of the baseline controllers:** consisting in a grid search in parameter space looking for the minimum E_p for each controller.For every different combination of parameter values, we ran a simulation episode using a constant wind profile of 17 m/s. These parameter values provide the reference performance as well as the initial condition for the ACRL optimization.
- **ACRL controller optimization**. This second battery of experiments used a training/evaluation methodology [30]. The agent was allowed to learn the control of the system for a total of 100 training episodes, using a randomly selected wind profile with a mean wind speed in range $[14, 17]\, m/s$. Every 10 training episodes, the controller was evaluated using a wind profile with mean wind speed of $17\, m/s$. The agent wasn't allowed to do any exploration during evaluation episodes.

5.2. Dynamic model

We used *OpenFAST* [8][1] in all our experiments, so that we have highly accurate reproduction of the complex dynamics of the actual physical system. *OpenFAST* is a state-of-the-art simulation suite certified by the *Germanischer Lloyd Windenergie GmbH* and mantained by the *National Renewable Energy Laboratory* (NREL) with support from the *US Department of Energy's Wind Energy Technology Office*. The specific model we used is the 5 MW reference VSWT presented in [13]. The OpenFAST parameter values used accross all the experiments are reported in Table 2. The wind profiles were generated with TurbSim, an stochastic, full-field, turbulent-wind simulator included in the *OpenFAST* simulation suite. Our simulations used a control time-step of $\Delta t = 0.00125$, as recommended by NREL developers. Using that setting, it took us approximately 6 s to compute 1 s of simulated time on a standard desktop computer.

5.3. Grid search tuning of the baseline controllers

To find the best parameters for each baseline controller, we carried out a grid search over the values shown in Table 3, testing all the possible combinations for every controller. This amounts to $(9 \times 7 \times 9 \times 4) = 2268$ evaluation runs of OpenFAST for the *Vidal* controller and $(9 \times 9 \times 4) = 324$ evaluation runs for *Boukhezzar*. This difference is due to the different number of parameters of the baseline models: *Vidal* controller has 4 parameters, whereas *Boukhezzar* has only 3. The best performance found for *Vidal* was $E_p = 99.160\,(\pm 474.139)\, kW$, setting parameter values as $\widehat{A} = 10^{-1}, \widehat{K}_\alpha = 5 \cdot 10^4$, $\widehat{K}_p = 10^{-2}$ and $\widehat{K}_i = 0$. On the other hand, *Boukhezzar* best performance found is $E_p = 21.764\,(\pm 220.198)\, kW$, setting parameter values as $\widehat{C}_0 = 1, \widehat{K}_p = 10^{-2}$ and $\widehat{K}_i = 0$.

5.4. ACRL optimization and grid search settings

The aim of the second experimental phase was to improve over the best parameterizations of the baseline controllers. We denote the ACRL optimized versions as *Vidal** and *Boukhezzar**. The wind speed v and the electrical generator torque T_g characterize the operating region of the VSWT, denoted as $S_l = [v, T_g]$. This definition allows the RL agent to tune the controller parameters for each operating region. Regarding VFA, Gaussian RBFs have consistently shown better results than linear VFA approaches (i.e., tile-coding), hence the experiments in this paper use a Gaussian RBF with 400 features: 20 uniformly distributed points along the range of values for each input variable in S_l. The Actor applies CACLA to search for the optimal policy whilst the Critic applies a *TD* learner to estimate the valuation function. The parameters of the Ornstein-Ühlenbeck process $O(t)$ (cf. Section 3.2) are set to $\rho = 0.1$ and $\sigma = 1$. We annealed some of the learning hyperparameters, specifically: the perturbation signal amplitude added to each of the Actor's output policies ($\psi^A, \psi^{K_\alpha}, \psi^{K_p}, \ldots$), the Critic's learning rate α^c and the Actor's learning rate α^a. We set these parameters initially to some initial value linearly decaying them down to 0 at the end of the experiment. We set the Actor's initial learning rate $\alpha_0^a = 10^{-6}$ and performed a grid search over the decaying hyperparameters. For each node in the grid, corresponding to a combination of hyperparameter values, a complete realiza-

[1] The repository is hosted in: https://github.com/OpenFAST/openfast.

Table 2
OpenFAST Parameter settings used in the experiments.

Nominal electrical power	P_{nom}	$5\,MW$
Nominal rotor speed	$\omega_{r_{nom}}$	$1.26711\,rad/s$
Nominal generator speed	$\omega_{g_{nom}}$	$122.91\,rad/s$
Gear-box ratio	n_g	97
Hub height	h	$90\,m$
Rotor diameter	d	$128\,m$
Air density	ρ	$1.225\,kg/m^3$
Blade pitch	β	$[0, \pi/2]$
Blade pitch rate	$\dot{\beta}$	$[-0.1396, 0.1396]$
Generator torque	T_g	$[0, 47402.91]$
Generator torque rate	$\dot{T_g}$	$[-15000, 15000]$

Table 3
Values used in the grid search of the baseline controllers before the RL optimization phase of the experiment. Last column (*) shows the optimal setting found by grid search.

Controller	Parameter	Grid Values	*
Vidal	A	$0.01, 0.025, 0.05, 0.075, 0.1, 0.25, 0.5, 0.75, 1$	10^{-1}
	K_α	$5 \cdot 10^3, 10^4, 5 \cdot 10^4, 10^5, 5 \cdot 10^5, 10^6, 5 \cdot 10^6$	$5 \cdot 10^4$
	K_p	$0.01, 0.25, 0.05, 0.075, 0.1, 0.25, 0.5, 0.75, 1$	10^{-2}
	K_i	$0, 0.01, 0.025, 0.05$	0
Boukhezzar	C_0	$0.01, 0.025, 0.05, 0.075, 0.1, 0.25, 0.5, 0.75, 1$	1
	K_p	$0.01, 0.025, 0.05, 0.075, 0.1, 0.25, 0.5, 0.75, 1$	10^{-2}
	K_i	$0, 0.01, 0.025, 0.05$	0

tion of the RL optimization process was performed. The specific values used for the ACRL hyperparameters grid search are shown in Table 4.

Actor weights are set initially to the best parameter values found parameter grid search experiments for each baseline controller (cf. Section 5.3) as stated in Eq. 19. When starting from the *Vidal* controller, initial values are as reported in Section 5.3: $\vec{\theta}_0^A = 10^{-1}$, $\vec{\theta}_0^{K_\alpha} = 5 \cdot 10^4$ and $\vec{\theta}_0^{K_p} = 10^{-2}$. When starting from the *Boukhezzar* model, it is simplified setting constant $K_i = 0$, other initial values were $\vec{\theta}_0^{C_0} = 1$, $\vec{\theta}_0^{K_p} = 10^{-2}$ and $\vec{\theta}_0^{K_i} = 0$. In both cases, the weights of the Critic were initialized with null values ($\vec{\theta}_0^c = 0$).

In order to align the reward signal with the control goal (i.e. minimize the absolute mean power error E_p), the reward function is defined as:

$$r = \max\left(1 - \left|\frac{P_e - P_n}{\mu}\right|, -1\right), \tag{25}$$

where μ is the tolerance parameter ($\mu = 5 \cdot 10^5$ in our experiments). This reward function returns a real value bound in the range $[-1, 1]$ that is proportional to the absolute power error. It is positive when the error is within the tolerance region and negative outside it. Fig. 2 shows the shape of the reward function.

5.5. Computation time requirements for ACRL simulations.

Each instance of the ACRL learning experiment consisted of 100 training episodes and 11 evaluation episodes. Each episode duration is $300\,s$ in simulated time. *OpenFAST*ratio of real time computation to simulated time is $\simeq 6$. It does not allow any kind parallelization. In a rough calculation, the total CPU time required for the OpenFAST simulation of the VSWT dynamics on a single CPU core to run a single ACRL experiment instance is approximately $111 \cdot 300 \cdot 6 \approx 2 \cdot 10^5\,s$, equivalent 2.31 days of execution. Therefore, the time requirements of the ACRL computations are negligible compared to OpenFAST simulation. The complete set of hyperparameter grid search experiments evaluates all the possible combinations of the 4 different values given to the 4 ACRL hyperparameters for each of the baseline controllers. This makes a total of $2 \cdot 4^4 = 512$ instances of the ACRL optimization experiment. Thus, a rough estimation of the total amount of CPU time required for OpenFAST simulations is $2.31 \cdot 512 = 1,184$ days. An inhouse developed tool *SimionZoo*[2] [49], allows to distribute the experimental instances over a farm of heterogenous computers. Exploiting *SimionZoo*, we were able to run as many instances simultaneously as available CPU cores, making the most possible efficient use of the resources. The available computer farm has 330 CPU cores, reducing the required total real computation time to less than 5 days if all the computers are com-

[2] https://github.com/simionsoft/SimionZoo

Table 4
Hyperparameter values used in grid search for the optimal learning of *Vidal** and *Boukhezzar**. Last column (*) shows the optimal setting found by grid search.

Experiment	Hyperparameter	Grid Values	*
*Vidal**	ψ_0^A	$10^{-2}, 2.5\cdot10^{-2}, 5\cdot10^{-2}$	$2.5\cdot10^{-2}$
	$\psi_0^{K_\alpha}$	$10^2, 10^3, 10^4, 10^5$	10^4
	$\psi_0^{K_p}$	$2.5\cdot10^{-2}, 5\cdot10^{-2}, 10^{-2}, 5\cdot10^{-3}$	$5\cdot10^{-3}$
	α_0^c	$10^{-2}, 5\cdot10^{-3}, 10^{-3}, 5\cdot10^{-4}$	10^{-5}
*Boukhezzar**	$\psi_0^{C_0}$	$10^{-2}, 5\cdot10^{-3}, 10^{-3}, 5\cdot10^{-4}$	10^{-2}
	$\psi_0^{K_p}$	$10^{-3}, 5\cdot10^{-4}, 10^{-4}, 5\cdot10^{-5}$	$5\cdot10^{-5}$
	$\psi_0^{K_i}$	$10^{-3}, 5\cdot10^{-4}, 10^{-4}, 5\cdot10^{-5}$	$5\cdot10^{-4}$,
	α_0^c	$10^{-2}, 5\cdot10^{-3}, 10^{-3}, 5\cdot10^{-4}$	10^{-2}

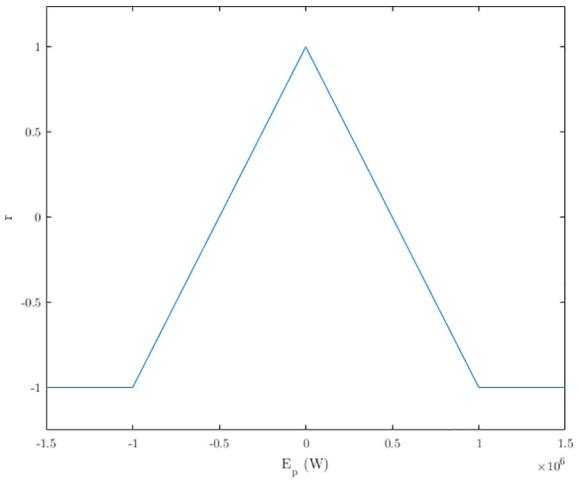


Fig. 2. The reward function used in our experiments. The output is a function of the absolute power error E_p.

pletely devoted to the project, which is not very often. For reproducibility purposes, we have published the project file needed to reproduce the experiments at the *Computational Intelligence Group*'s web page.[3]

5.6. ACRL grid search results

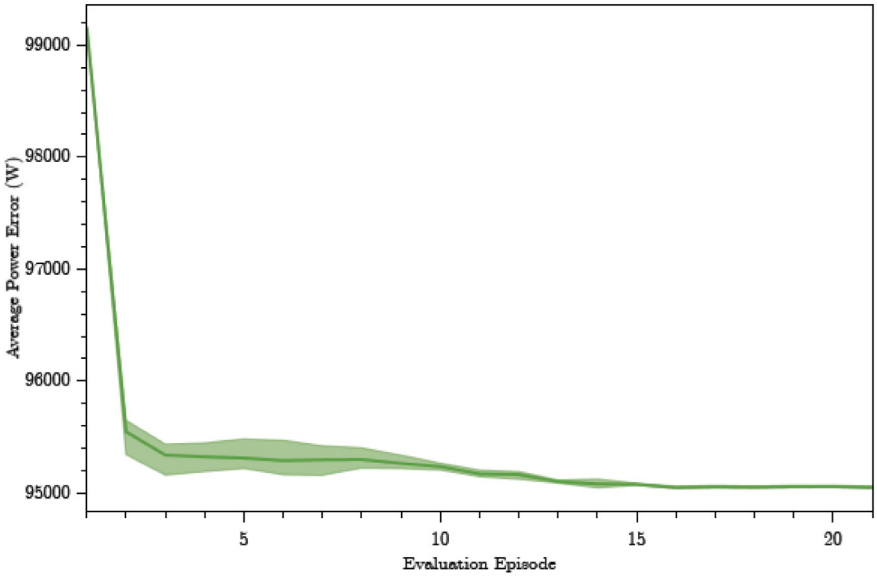
The best results achieved by the ACRL improved controller *Vidal** were obtained with the following settings of the hyperparameters: $\psi_0^A = 2.5\cdot10^{-2}, \psi_0^{K_\alpha} = 10^4$, $\psi_0^{K_p} = 5\cdot10^{-3}$, and $\alpha_0^c = 10^{-5}$. The best performing hyperparameter settings for *Boukhezzar** were: $\psi_0^{C_0} = 10^{-2}, \psi_0^{K_p} = 5\cdot10^{-5}$, $\psi_0^{K_i} = 5\cdot10^{-4}$, and $\alpha_0^c = 10^{-2}$. In Table 5 we compare the average reward and the average power error of the baseline controllers with the corresponding optimal controller found by ACRL. In both cases, the ACRL improved controller outperforms the baseline controller significantly. Regarding the average power error, *Vidal** and *Boukhezzar** are, respectively, 4.15% and 19.12% more efficient than the best the baseline controllers *Vidal* and *Boukhezzar*. The standard deviation of the power error also decreases by 4.58% and 24.30%, respectively, which means that the ACRL optimized controllers are more stable than the baseline controllers. This significant performance improvement suggests that adding a RL learner to the control loop of a VSWT may be worth the cost.

For a visual qualitative assessment of the different learning evolution of the *Vidal** and *Boukhezzar** model settings, Figs. 3a, 3b, 4a, and 4b show the three best performing learning instances of each model. Figs. 3b and 3a show, respectively, the average reward and absolute generated power error of the three best performing *Vidal** controllers computed in the evaluations episodes during the learning process. The dark green solid line represents the average values of the three averaged value series, and the light green area shows their [min, max] bounds. The performance measure values of the first evaluation episode correspond precisely to those of the baseline controllers (cf. Table 5). As the ACRL process progresses, the performance of the optimized controller *Vidal** improves, decreasing the average absolute power error and increasing the average reward. The plot shows that the performance variance of the three best *Vidal** controllers is rather low.

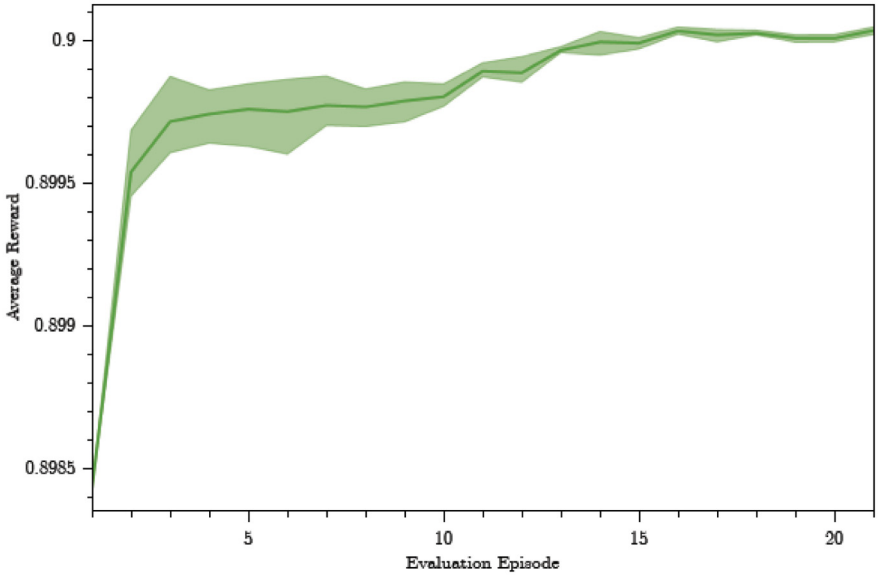
[3] The project file and the *SimionZoo* binaries for *Windows/Linux* can be downloaded fromhttp://ehu.eus/ccwintco/index.php?title=SimionZooProjects.

Table 5
Average (± standard deviation) rewards and absolute power error of the best ACRL trained model instances *Vidal** and *Boukhezzar**, compared with the baseline models *Vidal* and *Boukhezzar*.

Controller	Average reward	Average power error (*kW*)
Vidal	0.898(±0.416)	99.160(±21.77)
*Vidal**	0.900(±0.412)	95.041(±21.27)
Boukhezzar	0.975(±0.201)	21.764(±14.83)
*Boukhezzar**	0.976(±0.193)	17.602(±12.91)

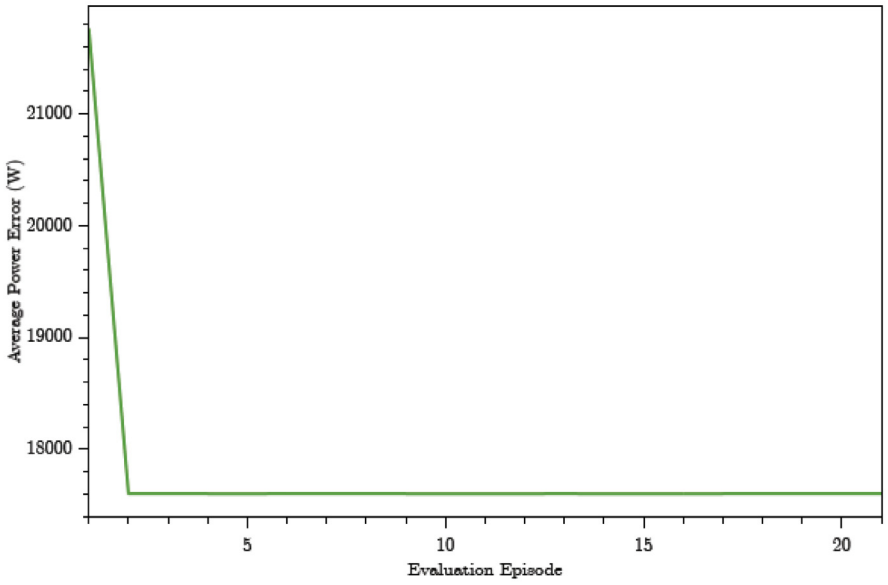


(a) Average absolute generated power error in Watts.

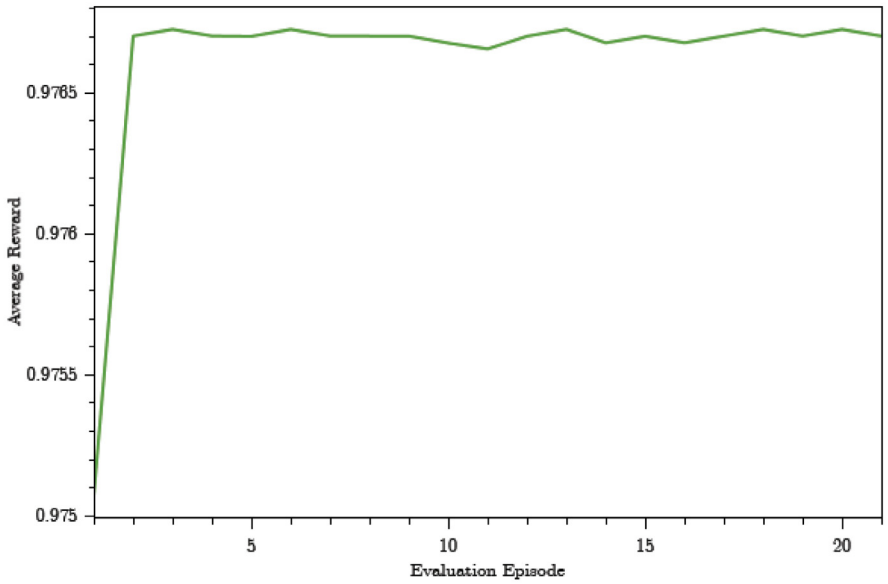


(b) Average rewards received by the Actor/Critic learner.

Fig. 3. Average values of the performance indicators of the three best *Vidal** controllers learned during the evaluation episodes. The solid lines represent the mean value and the soft band around it represents the [min, max] bounds of the three series.



(a) Average absolute generated power error in Watts.

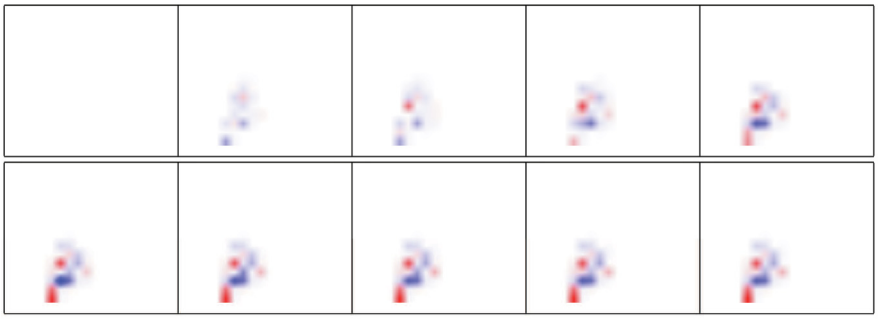


(b) Average rewards received by the Actor/Critic learner.

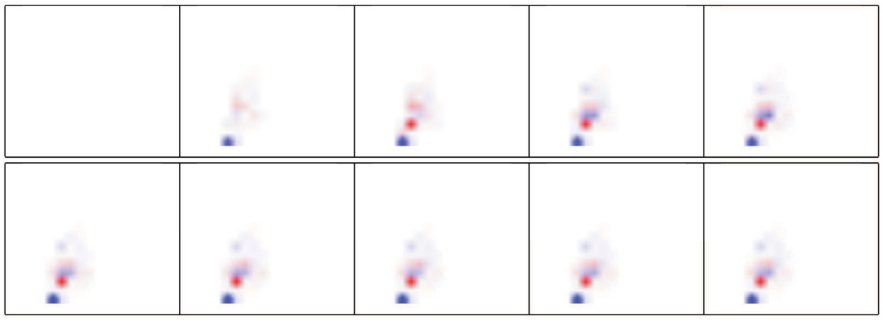
Fig. 4. Average values of the performance indicators of the three best *Boukhezzar** controllers learned during the evaluation episodes. The solid lines represent the mean value and the soft band around it represents the [min, max] bounds of the three series.

Figs. 4a and 4b plot the performance of the three best ARL optimized *Boukhezzar** controllers which is so similar that the light green showing the [min, max] bounds of the three series is barely distinguishable. Learning process for *Boukhezzar** controller model appears to be much more stable than for *Vidal** controller model.We have examined the results and concluded that two of the hyperparameters ($\psi_0^{K_i}$ and α_0^c) had little influence on the overall performance for any combination with $\psi_0^{C_0} = 10^{-2}$ and $\psi_0^{K_p} = 5 \cdot 10^{-5}$.

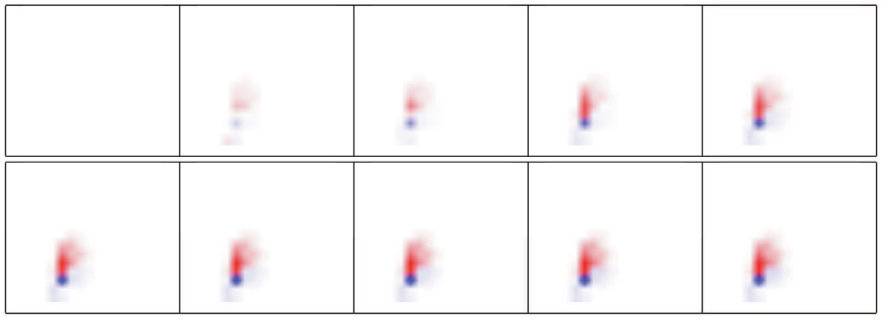
Figs. 5 and 6 show 10 snapshots of each function approximation learned while carrying out the ACRL optimization of *Vidal** and *Boukhezzar**, respectively. We represent the response of the function approximation of each parameter as a heat-map where bright red and blue correspond to the maximum and minimum value, respectively. All other values are



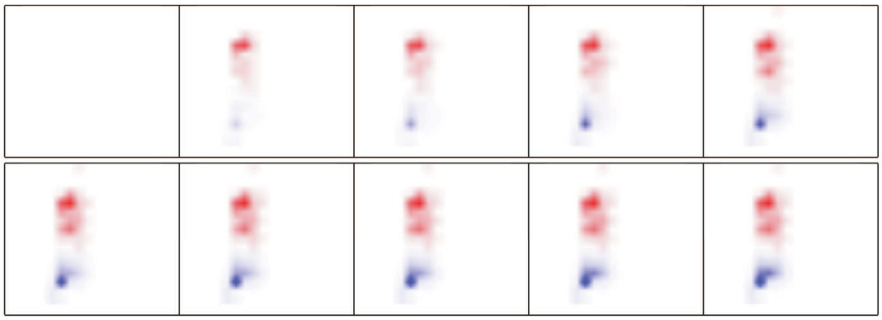
(a) Visual representation of $Vidal^*$ controller parameter A at different points in time.



(b) Visual representation of $Vidal^*$ controller parameter K_α at different points in time.

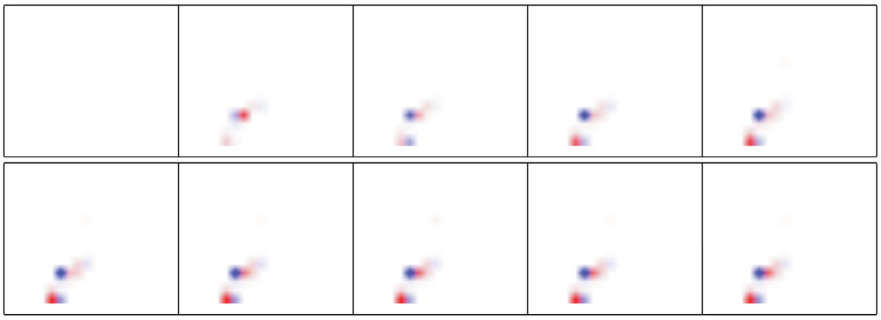


(c) Visual representation of $Vidal^*$ controller parameter K_p at different points in time.

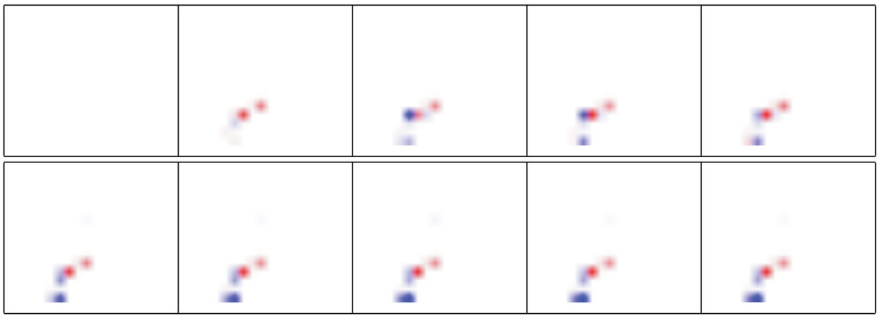


(d) Visual representation of the value function learned by the Critic at different points in time.

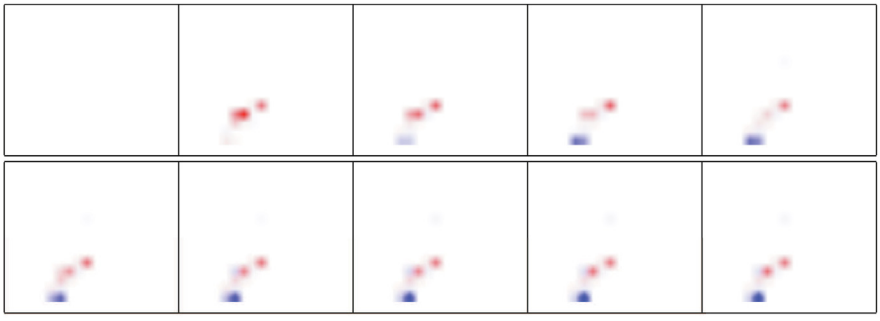
Fig. 5. Snapshots of the four functions learned during the learning of *Vidal** (the three policies and the value function) represented as heat-maps where red represents the highest value and blue represents the lowest value. Snapshots are ordered by time from left to right and from top to bottom.



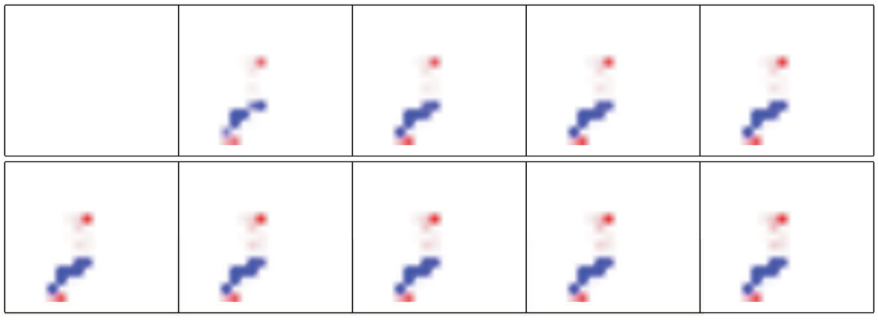
(a) Visual representation of $Boukhezzar^*$controller parameter C_0 at different points in time.



(b) Visual representation of $Boukhezzar^*$ controller parameter K_p at different points in time.



(c) Visual representation of $Boukhezzar^*$ controller parameter K_i at different points in time.



(d) Visual representation of the value function learned by the Critic at different points in time.

Fig. 6. Snapshots of the four functions learned during the learning of *Boukhezzar** (the three policies and the value function) represented as heat-maps where red represents the highest value and blue represents the lowest value. Snapshots are ordered by time from left to right and from top to bottom.

assigned a color using linear interpolation over the red-blue color map. The maximum and minimum values are calculated using the whole set of snapshots for a more coherent visualization of the functions. The axes of these plot representations are the range of values of the blade pitch and the generator torque. For each controller, we image the three Actor policies corresponding to the three baseline controller parameters and the value function learned by the Critic. The most interesting aspect of these heat-maps is that all policies show peaks and valleys, which means that the Actor performed an independent tuning of the parameters with respect to the input state variables. This shows that the proposed control architecture is not decreasing or increasing the rate uniformly, but tuning them depending on the working conditions characterized by the input variables in S_l (v and T_g).

6. Conclusions

This paper presents an Actor-Critic RL architecture to optimize the parameters of two baseline VSWT controllers (*Vidal* and *Boukhezzar*) using different sets of input variables for the RL agent and for the baseline controller. Additional state variables, which are independent of the baseline controller parameters, characterize the working condition of the system, allowing the ACRL agent to fine-tune the controller parameters differently for each working condition. The proposed architecture optimizes the parameters of each subsystem controller in order to minimize the overall electrical power error. This is in contrast with conventional approaches to VSWT control that decompose the control problem into two separate control subtasks, each relying on the other to collaborate toward the common control goal.

The approach has been validated using the state-of-the-art VSWT simulation suite OpenFAST. This is, up to the best of our knowledge, the first time a RL-based approach to VSWT control has been validated in such a realistic simulation environment. Computational experiments have shown that our architecture is able to improve over the baseline controllers, learning the parameters that maximize the global performance of the VSWT as a function of input variables not considered by the original baseline controllers. The improvement suggests that RL-enhanced controllers may be worth the cost of the required investment.

The computational experiments have used linear VFAs because they can be evaluated and updated faster than non-linear VFAs, making them more suitable for real-time control. Nevertheless, as graphical processing units (GPU) are evolving fast, the technology may allow soon the use of Deep RL methods in real-time control problems with very fast control time step requirements (i.e, VSWT control). This is a very interesting line of research we plan to follow in the future.

CRediT authorship contribution statement

Borja Fernandez-Gauna: Software, Investigation. **Manuel Graña:** Resources, Investigation. **Juan-Luis Osa-Amilibia:** Resources, Supervision. **Xabier Larrucea:** Methodology, Supervision.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been partially supported by FEDER funds through MINECO project TIN2017-85827-P, MCIN project PID2020-116346 GB-I00, and project KK-202000044 of the Elkartek 2020 funding program of the Basque Government.

References

[1] A. Ilas, P. Ralon, A. Rodriguez, M. Taylor, Renewable power generation costs in 2017 (2018)..
[2] N. Khezami, N.B. Braiek, X. Guillaud, Wind turbine power tracking using an improved multimodel quadratic approach, ISA Trans. 49 (3) (2010) 326–334.
[3] B. Beltran, T. Ahmed-Ali, M. Benbouzid, High-order sliding-mode control of variable-speed wind turbines, Ind. Electron. IEEE Trans. 56 (9) (2009) 3314–3321, https://doi.org/10.1109/TIE.2008.2006949.
[4] B. Boukhezzar, H. Siguerdidjane, Nonlinear control of a variable-speed wind turbine using a two-mass model, Energy Convers. IEEE Trans. 26 (1) (2011) 149–162, https://doi.org/10.1109/TEC.2010.2090155.
[5] A. Merabet, J. Thongam, J. Gu, Torque and pitch angle control for variable speed wind turbines in all operating regimes, in: Environment and Electrical Engineering (EEEIC), 2011 10th International Conference on, 2011, pp. 1–5, https://doi.org/10.1109/EEEIC.2011.5874598.
[6] J.O.M. Rubio, L.T. Aguilar, Maximizing the performance of variable speed wind turbine with nonlinear output feedback control, Proc. Eng. 35 (2012) 31–40, https://doi.org/10.1016/j.proeng.2012.04.162.
[7] B. Fernández-Gauna, U. Fernandez-Gamiz, M. Graña, Variable speed wind turbine controller adaptation by reinforcement learning, Integr. Comput.-Aided Eng. 24 (1) (2017) 27–39, https://doi.org/10.3233/ICA-160531.
[8] J. Jonkman, M.J. Buhl, Fast user's guide, Tech. Rep. NREL/TP-500-38230, National Renewable Energy Laboratory, 1617 Cole Blvd. Golden, CO (2005). URL: https://www.nrel.gov/docs/fy06osti/38230.pdf..
[9] Y. Vidal, L. Acho, N. Luo, M. Zapateiro, F. Pozo, Power control design for variable-speed wind turbines, Energies 5 (8) (2012) 3033–3050.
[10] B. Boukhezzar, L. Lupu, H. Siguerdidjane, M. Hand, Multivariable control strategy for variable speed, variable pitch wind turbines, Renew. Energy 32 (8) (2007) 1273–1287.

[11] A. Evans, V. Strezov, T.J. Evans, Assessment of sustainability indicators for renewable energy technologies, Renew. Sustain. Energy Rev. 13 (5) (2009) 1082–1088, https://doi.org/10.1016/j.rser.2008.03.008.
[12] C.L. Archer, M.Z. Jacobson, Evaluation of global wind power, J. Geophys. Res.: Atmospheres 110 (D12) (2005), https://doi.org/10.1029/2004JD005462.
[13] J. Jonkman, S. Butterfield, W. Musial, G. Scott, Definition of a 5-mw reference wind turbine for offshore system development, National Renewable Energy Laboratorys, Golde, CO, USA, 2009.
[14] J. Ziegler, Nichols, Optimum settings for automatic controllers, Trans. ASME 64 (1942) 759–768.
[15] R. Saravanakumar, D. Jena, Validation of an integral sliding mode control for optimal control of a three blade variable speed variable pitch wind turbine, Int. J. Electr. Power Energy Syst. 69 (2015) 421–429, https://doi.org/10.1016/j.ijepes.2015.01.031.
[16] B. Torchani, A. Sellami, G. Garcia, Variable speed wind turbine control by discrete-time sliding mode approach, ISA Trans. 62 (2016) 81–86, sI: Control of Renewable Energy Systems. doi: 10.1016/j.isatra.2016.01.001..
[17] D. Song, J. Yang, M. Dong, Y.H. Joo, Model predictive control with finite control set for variable-speed wind turbines, Energy 126 (2017) 564–572, https://doi.org/10.1016/j.energy.2017.02.149.
[18] A. Staino, B. Basu, Dynamics and control of vibrations in wind turbines with variable rotor speed, Eng. Struct. 56 (2013) 58–67, https://doi.org/10.1016/j.engstruct.2013.03.014.
[19] C. Viveiros, R. Melicio, J. Igreja, V. Mendes, Supervisory control of a variable speed wind turbine with doubly fed induction generator, Energy Rep. 1 (2015) 89–95, https://doi.org/10.1016/j.egyr.2015.03.001.
[20] A. Iqbal, D. Ying, A. Saleem, M.A. Hayat, K. Mehmood, Efficacious pitch angle control of variable-speed wind turbine using fuzzy based predictive controller, Energy Rep. 6 (2020) 423–427, the 6th International Conference on Power and Energy Systems Engineering. doi: 10.1016/j.egyr.2019.11.097..
[21] I. Serban, C. Marinescu, A sensorless control method for variable-speed small wind turbines, Renew. Energy 43 (2012) 256–266, https://doi.org/10.1016/j.renene.2011.12.018.
[22] S. Prasad, S. Purwar, N. Kishor, Non-linear sliding mode control for frequency regulation with variable-speed wind turbine systems, Int. J. Electr. Power Energy Syst. 107 (2019) 19–33, https://doi.org/10.1016/j.ijepes.2018.11.005.
[23] J. Mérida, L.T. Aguilar, J. Dávila, Analysis and synthesis of sliding mode control for large scale variable speed wind turbine for power optimization, Renew. Energy 71 (2014) 715–728, https://doi.org/10.1016/j.renene.2014.06.030.
[24] H.M. Hasanien, S. Muyeen, Speed control of grid-connected switched reluctance generator driven by variable speed wind turbine using adaptive neural network controller, Electric Power Syst. Res. 84 (1) (2012) 206–213, https://doi.org/10.1016/j.epsr.2011.11.019.
[25] M. Sedighizadeh, A. Rezazadeh, Adaptive pid controller based on reinforcement learning for wind turbine control, World Acad. Sci. Eng. Technol. 2 (1) (2008) 124–129.
[26] J.-S. Kim, J. Jeon, H. Heo, Design of adaptive pid for pitch control of large wind turbine generator, in: Environment and Electrical Engineering (EEEIC), 10th International Conference on, 2011, pp. 1–4, https://doi.org/10.1109/EEEIC.2011.5874603.
[27] M. Akbari, N. Ghadimi, Reinforcement learning based pid control of wind enery conversion systems, J. Artif. Intell. Electr. Eng. 3 (2014) 759–768.
[28] I. Grondman, L. Busoniu, G. Lopes, R. Babuska, A survey of actor-critic reinforcement learning: Standard and natural policy gradients, Syst. Man Cybern. Part C 42 (6) (2012) 1291–1307.
[29] G. Leuenberger, M.A. Wiering, Actor-critic reinforcement learning with neural networks in continuous games, Proceedings of the 10th International Conference on Agents and Artificial Intelligence, vol. 2, Funchal, Madeira, Portugal, 2018, pp. 53–60, https://doi.org/10.5220/0006556500530060.
[30] R. Hafner, M. Riedmiller, Reinforcement learning in feedback control: Challenges and benchmarks from technical process control, Mach. Learn. 84 (1–2) (2011) 137–169, https://doi.org/10.1007/s10994-011-5235-x.
[31] R. Sutton, A. Barto, Reinforcement Learning: An Introduction, MIT Press, 1998.
[32] T. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, CoRR (09 2015)..
[33] J.N. Tsitsiklis, B. Van Roy, An analysis of temporal-difference learning with function approximation, IEEE Trans. Autom. Control 42 (5) (1997) 674–690.
[34] T. Hofmann, B. Schölkopf, A. Smola, Kernel methods in machine learning, Ann. Stat. 36 (3) (2008) 1171–1220, https://doi.org/10.1214/009053607000000677.
[35] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, Deterministic policy gradient algorithms, in: Proceedings of the 31st International Conference on International Conference on Machine Learning – Volume 32, ICML14, JMLR.org, 2014, pp. 387–395. URL:http://dl.acm.org/citation.cfm?id=3044805.3044850..
[36] G.E. Uhlenbeck, L.S. Ornstein, On the theory of the brownian motion, Phys. Rev. 36 (1930) 823–841, https://doi.org/10.1103/PhysRev.36.823.
[37] H. Stark, J. Woods, Probability and Random Processes with Applications to Signal Processing, third ed., Prentice Hall, New Jersey, 2002.
[38] H. v. Hasselt, M.A. Wiering, Reinforcement learning in continuous action spaces, in: 2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning, 2007, pp. 272–279. doi:10.1109/ADPRL.2007.368199..
[39] J. Zhai, Q. Liu, Z. Zhang, S. Zhong, H. Zhu, P. Zhang, C. Sun,, Lecture Notes Comput. Sci. 9947 (2016) 13–22, cited By 11. doi:10.1007/978-3-319-46687-3_2..
[40] S. You, M. Diao, L. Gao, F. Zhang, H. Wang, Applied Soft Computing Journal 95, cited By 0 (2020). doi:10.1016/j.asoc.2020.106490..
[41] C. Wang, C. Yan, X. Xiang, H. Zhou, A continuous actor-critic reinforcement learning approach to flocking with fixed-wing uavs, in: W.S. Lee, T. Suzuki (Eds.), Proceedings of The Eleventh Asian Conference on Machine Learning, Vol. 101 of Proceedings of Machine Learning Research, PMLR, Nagoya, Japan, 2019, pp. 64–79. URL: http://proceedings.mlr.press/v101/wang19a.html..
[42] C. Yan, X. Xiang, C. Wang, Fixed-wing uavs flocking in continuous spaces: A deep reinforcement learning approach, Robot. Autonomous Syst. 131 (2020), https://doi.org/10.1016/j.robot.2020.103594 103594.
[43] P. Hämäläinen, A. Babadi, X. Ma, J. Lehtinen, Ppo-cma: Proximal policy optimization with covariance matrix adaptation, in: 2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP), 2020, pp. 1–6, https://doi.org/10.1109/MLSP49062.2020.9231618.
[44] H. v. Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI'16, AAAI Press, 2016, pp. 2094–2100. URL: http://dl.acm.org/citation.cfm?id=3016100.3016191..
[45] B. Fernandez-Gauna, I. Marques, M. Graña, Undesired state-action prediction in multi-agent reinforcement learning. application to multicomponent robotic system control, Inf. Sci. 232 (2013) 309–324.
[46] B. Fernandez-Gauna, J. Lopez-Guede, M. Graña, Transfer learning with partially constrained models: application to reinforcement learning of linked multicomponent robot system control, Robot. Autonomous Syst. 61 (7) (2013) 694–703.
[47] B. Fernandez-Gauna, M. Graña, J.M. Lopez-Guede, I. Ansoategui, Reinforcement learning endowed with safe veto policies to learn the control of l-mcrs, Inf. Sci. 317 (2015) 25–47.
[48] B. Fernandez-Gauna, J.-L. Osa, M. Graña, Effect of initial conditioning of reinforcement learning agents on feedback control tasks over continuous state and action spaces, in: International Joint Conference SOCO14-CISIS14-ICEUTE14, Vol. 299 of Advances in Intelligent Systems and Computing, Springer International Publishing, 2014, pp. 125–133.
[49] B. Fernandez-Gauna, M. Graña, R. Zimmermann, Simion zoo: A workbench for distributed experimentation with reinforcement learning for continuous control tasks, Tech. rep., Arxiv. URL:http://arxiv.org/abs/1904.07817..