Fernandez-Gauna, Borja, Fernandez-Gamiz, Unai, and Graña, Manuel. 'Variable Speed Wind Turbine Controller Adaptation by Reinforcement Learning' **Integrated Computer-Aided Engineering**, vol. 24, no. 1, pp. 27-39, 2017. Copyright © 2017 IOS. DOI: 10.3233/ICA-160531

Variable speed wind turbine controller adaptation by reinforcement learning

Borja Fernandez-Gauna[a,b], Unai Fernandez-Gamiz[b] and Manuel Graña[a,c,*]
[a]*Computational Intelligence Group of the University of the Basque Country (UPV/EHU), Basque, Spain*
[b]*Polytechnical School, UPV/EHU, Spain*
[c]*ENGINE Centre, Wroclaw University of Technology, Wroclaw, Poland*

Abstract. The control of Variable Speed Wind Turbines (VSWT) to achieve optimal balance of power generation stability and rotor angular speed is impeded by the non-linear dynamics of the turbine-wind interaction and sudden changes of wind direction and speed. Conventional approaches to design VSWT controllers are not adaptive. However, the wind shear phenomenon introduces a strongly non-stationary environment that requires adaptive control approaches with minimal human intervention, i.e. very little supervision of the adaptation process. Reinforcement Learning (RL) allows minimally supervised learning. Specifically, Actor-Critic is designed to deal with continuous valued state and action spaces. In this paper we apply an Actor-Critic RL architecture to improve the adaptation of the conventional VSWT controllers to changing wind conditions. Simulation results on a benchmark VSWT model under strongly changing wind conditions show that Actor Critic RL approach with functional approximation provide great enhancement over state-of-the-art VSWT controllers.

Keywords: Wind-turbine, control, reinforcement, learning, adaptive

1. Introduction

With the growing demand for renewable (green) energies, the use of Wind Turbines (WT) has got a great impulse and its acceptance is widely spread, becoming a big part of the energy market in some countries, like Spain where WT peak production in specific days accounted for over 30% of the country's electrical power production. Their main disadvantage is that the energy generation depends on the wind conditions. Therefore, much effort is being put to improve their performance under the most challenging conditions [32], to ensure that they are capable of a steady production, so that they can be a reliable energy source. The control of WT poses some strong challenges: (a) it is a multi-objective control task, (b) it involves multiple control variables, (c) the system has very complex dynamics, (d) the need of adaptive controllers [2,4,22,24,37,41] to reduce the maintenance costs, and (e) the requirement of different control strategies depending on the operation region [25].

The works reported in the literature tackling the WT control problem have two shortcomings. Firstly they are based on the assumption of the detailed knowledge of an accurate dynamical model of the interaction between the WT and the environment. Secondly, they are not adaptive and thus, they are unable to compensate for model inaccuracies or non-stationary environments as it is often the case in WT operation. They follow a classical control theory approach, applying conventional analytical techniques [7,8,49], multi-model quadratic control [21], sliding control [5], non-linear H_∞ control [8], k-step ahead prediction [29], and fuzzy logic reasoning systems [1,17,34,38,45,53] to provide more flexible control. They can be quite optimal in very narrow conditions, however they need some mechanism for automatic tuning to changing environment conditions. Moreover, the WT control is a multi-objective problem, but most of the referred ap-

*Corresponding author: Manuel Graña, Department of CCIA, Facultad de Informatica, University of the Basque Country (UPV/EHU) Paseo Manuel Lardizabal, Donostia-San Sebastian, Spain. Tel.: +34 943018000; E-mail: manuel.grana@ehu.es.

proaches assume a single control objective. The few published multiobjective WT control approaches, treat the objectives (i.e., rotor speed and electrical power generation) as independent, so that the problem is decomposed into as many independent control problems as objectives.

We have found few attempts to use Reinforcement Learning (RL) in WT control design [26,44]. In this paper, we consider two baseline WT multi- sion: variable controllers, which will be denoted by the names of their authors: *Boukhezzar* [7] and *Vidal* [49]. Scalarized Multi-Objective Reinforcement Learning (MORL) [39] is used to improve these controllers with respect to user-defined criteria. The process is as follows: First, we build a Value Function Approximation (VFA) [10] of the baseline controller (either Boukhezzar or Vidal). Second, the VFA model is used by the MORL agent in an Actor-Critic framework [13] aiming to improve the baseline controller through interaction with the environment, following an online exploration/exploitation strategy. The user's objectives are introduced in the MORL as the reward function after scalarization of the multi-objective function, which is achieved by a weighted combination of the single-objective functions, where the weights are set by the user according to *a priori* defined priorities. The overall problem tackled here is among the most challenging in the context of current RL research [16,27,36], and might be useful to demonstrate the value of RL for practical real life problems.

The structure of the paper is as follows: In Section 2 we review the basic concepts involved in Variable-Speed Wind Turbines (VSWT) control, and the dynamical system model used for the simulation in our experiments. Section 3 offers some background on RL methods with continuous state and action spaces, defining the Actor-Critic methods that we will be using in the experiments. Then, we present how a conventional controller is approximated in Section 4. In Section 5 we describe the design of the experiments and the results. Finally, we give our conclusions in Section 6.

2. Variable speed wind turbines

A WT extracts kinetic energy from the wind and transforms it into electrical power. Theoretically, the power potential from the wind P_w is given by $P_w = \frac{1}{2}\rho \cdot \pi \cdot R^2 \cdot \upsilon^3$, where υ is the wind speed in m/s, ρ is the air density in kg/m^3, and R is the radius in meters of the external circumference drawn by the rotor blade tips. The ratio of power actually converted into electricity is called the *power coefficient* given by the WT-specific power coefficient function $c_p(\lambda,\beta)$, which is itself a function of the angle of the rotor blades β and the *tip speed ratio* $\lambda = \omega_r \cdot R/\upsilon$, where ω_r is the rotor speed (rad/s). The aerodynamic power P_a (in W) captured by a wind generator is then given by the expression:

$$P_a = \rho\frac{1}{2}\pi \cdot R^2 \cdot c_p(\lambda,\beta) \cdot \upsilon^3, \tag{1}$$

and the aerodynamic torque T_a (in N·m) can be calculated from the following relation:

$$T_a = P_a/\omega_r. \tag{2}$$

There are two kinds of WT designs [33,34]: fixed speed and variable speed. *Fixed Speed Wind Turbines* (FSWT) consist of induction generators directly coupled to the electricity transmission grid running at a nearly constant rotational speed. They are relatively cheap, and require little maintenance, but they are aerodynamically efficient only within a short range of wind speeds, they draw big amounts ofreactive power (stored energy that returns to the source), and suffer strong structural loads. *Variable Speed Wind Turbines* (VSWT), on the other hand, adjust the rotor speed using a blade pitch controller. They are decoupled from the grid by power electronic converters, which introduce power losses, consequently VSWT electrical conversion is less efficient than FSWT. On the other hand, VSWT are aerodynamically efficient for a wider range of wind speeds than FSWT, compensating in the long run the electrical conversion inefficiency and other additional costs. The main trend in industry in the last years is to favor VSWT over FSWT. New control strategies such as the one presented in this paper are key to further take advantage of variable speed mechanisms improving the quality of the power and reducing the maintenance costs.

2.1. Control goals

The VSWT controllers aim to fulfill three goals:

– Control of electrical power generation P_e, to maintain a reference power output P_{ref}, minimizing the power generation error $e_p = P_{ref} - P_e$. In this work, P_{ref} is assumed to be the nominal output power P_{nom} of the VSWT as specified by the manufacturer.

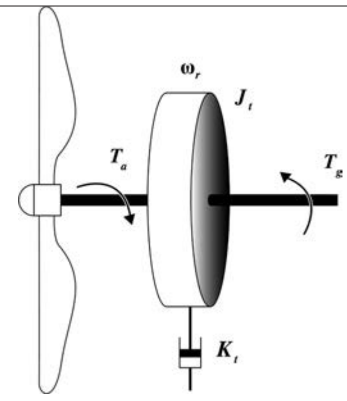

Fig. 1. Schematic representation of the one-mass model of a VSWT. T_a is the aerodynamic torque, ω_r is the speed of the rotor, K_t is the total external damping, J_t is the total inertia of the turbine, and T_g is the generator torque.

- Minimize the rotor angular speed error $e_\omega = \omega_{ref} - \omega_r$. Likewise, we assume that the reference rotor angular speed ω_{ref} is the VSWT's nominal value ω_{nom}.
- Minimize the transient loads of the control variables.

2.2. Dynamical model

In our computational experiments, we use the most common dynamical model in the VSWT control literature [5,7,8,29,40,49]. The schema of this one-mass system is represented in Fig. 1. The equation describing the dynamics of the rotor speed is as follows:

$$\dot{\omega}_r = \frac{T_a - K_t\omega_r - T_g}{J_t}, \tag{3}$$

where K_t and J_t are the total external damping (in N·m/rad) and the total inertia of the turbine (in kg·m^2), respectively. From Eq. (2), the electrical power P_e produced by the generator can be calculated as follows:

$$P_e = T_g \cdot \omega_r. \tag{4}$$

The power coefficient function is unique to each wind turbine type, and manufacturers usually provide a look-up table for operation purposes. Some approximation methods have been proposed when this table is not available. We have used the following numerical approximation [29]:

$$c_p(\lambda,\beta) = 0.5\left(\frac{116}{\lambda_1} - 0.4\cdot\beta - 5\right)\exp\left(\frac{-16.5}{\lambda_1}\right), \tag{5}$$

where β is the angle of the rotor blades in degrees, and

$$\lambda_1 = \frac{1}{\frac{1}{(\lambda + 0.08\cdot\beta)} - \frac{0.035}{(\beta^3+1)}}. \tag{6}$$

In our experiments, parameters were set according to the specifications of the Controls Advanced Research Turbine available at the National Wind Technology Center in Golden, Colorado.

2.3. VSWT baseline controllers

Classical VSWT control techniques often use the blade pitch to control the rotor speed when the VSWT is in the operational region below the nominal-speed, and the generator torque to control the power output when it is in the operational region above the nominal-speed. Modern multi-variable control approaches, however, control both the blade pitch β (in the following equations, the pitch is given in radians) and the generator torque T_g (in N·m) [7,8,29,40,49]. In our computational experiments, we have improved by RL two recently proposed multi-variable controllers, which we call the baseline controllers:

The *Boukhezzar* controller [7] defined by the equations:

$$\dot{T}_g = \frac{1}{\omega_r}\left(c_0 e_p - \frac{1}{J_t}\left(T_a T_g - K_t \omega_r T_g - T_g^2\right)\right), \tag{7}$$

$$\dot{\beta} = K_p e_\omega. \tag{8}$$

where c_0 and K_p are the adaptation gains of the controller's two outputs.

The *Vidal* controller [49] defined by equations:

$$\dot{T}_g = \frac{1}{\omega_r}\left[-T_g\left(a\omega_r + \dot{\omega}_r\right) + aP_{ref} + K_\alpha sgn(e_p)\right], \tag{9}$$

$$\dot{\beta} = \frac{1}{2}K_p e_\omega\left(1 + sgn(e_\omega)\right) + K_i\int_0^t e_\omega \cdot dt. \tag{10}$$

where a and K_α parameters control the torque controller convergence time. Parameters K_p and K_i are the gains of the proposed blade pitch PI controller.

Wind measurement and prediction is often very noisy, so it is not often directly used by control modules. It is mainly used for cut in and cut off of the turbine when there is too slow or too fast wind. Therefore, current controller designs use the relation between the generator speed and the wind speed to avoid explicitly using a wind speed measurement in the control algorithm. Vidal and Boukhezzar controllers, instead of using the direct measurements of wind speed, derive the control information from the 1-mass model that relates the generator torque and the wind speed.

3. Background on reinforcement learning

3.1. Reinforcement learning

The interaction between an agent and its environment in RL is modeled by Markov Decision Processes (MDP) [46], which are defined by a state space S, an action space A, a transition function P, and a reward function R giving a measure of how good the current system state is. Learning the control of a system modeled as a MDP (S, A, P, R) is the search for an action selection policy $\pi : S \rightarrow A$ maximizing the expected accumulated reward for any state of the controlled system $s \in S$. Accumulated discounted rewards define the function to be maximized, called the *value function* $V^{\pi}(s)$, which defines the value of being in state s and following policy π thereafter as

$$V^{\pi}(s) = E^{\pi} \sum_{k=1}^{\infty} \cdot \gamma^{k-1} \; s_t = s \qquad (11)$$

where s_t is the state observed in time step t, r_t is the reward, and γ is the discounting factor [0, 1].

3.2. Value function approximation

Real-world control problems often require the use of continuous state and action spaces. The state space $S \subseteq R^n$ is spanned by the state variables $x_1, x_2 \ldots x_n$, and the action space is defined as $A \subseteq R^m$, spanned by m control variables $u_1, u_2 \ldots u_m$. RL methods build estimations of the value function V in order to evaluate the current policy to make decisions, and to update it during the learning process. In the case of multi-dimensional continuous states $\underline{s} = [x_1 x_2 \ldots x_n]$, the value function $V^{\pi}(\underline{s})$ can not be approximated in tabular form, so that a Value Function Approximator (VFA) [10,48,50] must be built. VFA can be linear combinations of some local basis functions, such as Radial Basis Functions (RBF), or global non-linear functions, such as neural networks approximations [14, 43,51]. A continuous state-action policy with multiple outputs $\underline{\pi}(\underline{s})$ has an n-dimensional continuous input space and an m-dimensional output space, and can be decomposed into m single-output policies of n-dimensional input: $\underline{\pi}(\underline{s}) = \pi^1(\underline{s})\, \pi^2(\underline{s}) \ldots \pi^m(\underline{s})$. We will denote by $\underline{a} = [u_1 u_2 \ldots u_m]$ the action vectors generated by these multidimensional policies. VFAs using linear combinations of RBFs map the input space S into a feature space Φ building a map $\varphi : S \rightarrow \Phi$. The feature space is a real-valued vector space $\Phi \in R^f$, spanned by a set of f features. Each feature corresponds to an RBF, characterized by a center point $\overleftarrow{c} \in S$. The centers of the RBFs can be disposed in a grid sampling the state space. Assuming that state variables can be taken independently (i.e. there are no interactions), a specific set of center points $c_{i,j} \in R$ is defined for the i-th state variable $\{c_{i,1}, c_{i,2} \ldots c_{i,f_i}\}$ distributed along its range of values $[\min_i, \max_i]$. The designer may want to use a different number of feature centers f_i for each state variable x_i. Each possible combination of center points for each state variable are associated with a different feature using some mapping function $\psi(i, j)$ that gives the index k of the center point $c_{i,k}$ associated with the j-th feature of the i-th state variable. For any given state $\underline{s}$, the feature vector $\varphi = [\varphi_1(\underline{s})\, \varphi_2(\underline{s}) \ldots \varphi_f(\underline{s})]$ is calculated using activation functions $\varphi_j(\underline{s})$, i.e. Gaussian Radial Basis Functions (RBF):

$$\varphi_j(\underline{s}) = \prod_{i=1}^{n} \exp\left(- \frac{\| x_i - c_{i,\psi(i,j)} \|^2}{2\sigma^2} \right), \qquad (12)$$

where the parameter σ is the spread of the Gaussian function shaping the activation function, and $x_1, x_2 \ldots x_n$ are the values of the n state variables in $\underline{s} = [x_1 x_2 \ldots x_n]$. The value function V is approximated as the inner product $V(\underline{s}) = \underline{\theta}' \cdot \varphi(\underline{s})$, where $\underline{\theta}_t = [\theta_1 \theta_2 \ldots \theta_f]$ is the vector of weights to be learned by the RL algorithm. Action selection policy will also be represented using a RBF based VFA decomposition, therefore we need to specify notational differences $\underline{\theta}^V$ and φ^V from $\underline{\theta}^{\pi,i}$ and $\varphi^{\pi,i}$ corresponding to the VFA of V and $\underline{\pi}'$, respectively.

3.3. Actor-Critic RL architectures

Several RL methods have been proposed in the literature to learn the control of systems with continuous states and actions. We require online and model-free methods (that don't assume the knowledge of an accurate model of the environment), because they promise adaptive learning to environments with unknown or even slowly changing dynamics. Because they allow for continuous state and action spaces, the most appropriate are Actor-Critic architectures [13,15], which consist of two separate structures: an *Actor*, which implements a policy $\underline{\pi}$, (i.e. carries out the decisions), and a *Critic*, which builds the estimation of the actor's policy's value V^{π}.

The Actor-Critic learning cycle proceeds as follows:

– The actor receives observation of the state $\underline{s}_t$,

- The actor generates an action vector $\underline{a}_t$ according to its own policy $\underline{\pi}$.
- The action is executed, so that the environment reaches a new state $\underline{s}_{t+1}$ with associated reward r_{t+1} received by the agent.
- The critic uses the reward to update its estimation of the policy's value function in state $\underline{s}_t$.
- The updated value function is used by the actor to update its policy $\underline{\pi}$.

The Actor's greedy optimal policy is to choose the with maximal value in the current estimation state-action value functions. However, for the learning process to be able to improve on this policy, the system needs some *exploration* mechanism that allows to test actions different from the ones dictated by the greedy policy. Without exploration the agent will be deterministically selecting always the same actions in the same state. In continuous state-action spaces, exploration is achieved adding a perturbation term $\underline{\mu} = [\mu_1\mu_2 \dots \mu_m] \subseteq R^m$ to the actor generated action to obtain the action actually executed: $\underline{a} = \underline{\pi}(\underline{s}) + \underline{\mu}$, $\underline{a} = [u_1u_2 \dots u_m]$. In our simulations, each μ_i follows a normal probability distribution $N(0, \sigma_t^2)$. The variance parameter σ_t determines the breadth of exploration at time step t. It can be a fixed value or be decreased along time using some annealing process.

Both Actor and Critic update their parameter vectors $\theta^{\pi,i}$ and θ^V following some update rule determined by the specific algorithm chosen. In our experiments, we use a Temporal-Difference (λ) $(TD\,(\lambda))$ critic [46], which updates its estimates using the following rule:

$$\underline{\theta}^V_{t+1} \leftarrow \underline{\theta}^V_t + \alpha_t \left(r_{t+1} + \gamma \cdot \hat{V}_t(\underline{s}_{t+1}) - \hat{V}_t(\underline{s}_t) \right) \cdot \underline{Z}_{t+1}, \quad (13)$$

where α_t is the learning gain and the eligibility trace vector is defined

$$\underline{Z}_{t+1} = \gamma\lambda \underline{Z}_t + \underline{\varphi}^V(\underline{s}_t).$$

We use the $TD\,(\lambda)$ Critic update rule because it is computationally inexpensive, therefore it is well suited for high-dimensional real-world problems, such as WT control tasks. Low-dimensional applications might benefit from using some more advanced, but also computationally more expensive methods as Natural Actor-Critic [6], or Least Squares-based methods [3,23] such as Least Squares Policy Iteration [28] or Least Squares Policy Evaluation [35]. The actor policy parameters are updated using the Continuous Action Critic Learning Automaton (CACLA) [13,48] update rule:

$$\underline{\theta}^{\pi,i}_{t+1} \leftarrow \underline{\theta}^{\pi,i}_t + \alpha_t \cdot (u_i - \pi_i(\underline{s}_t)) \cdot \frac{\partial \pi_i(\underline{s}_t)}{\partial \underline{\theta}^{\pi,i}_t}, \quad (14)$$

for $i = 1, \dots, m$. This update rule is applied only if the Critic's last update was a positive increment, because negative shifts do not necessarily improve the policy value. The dimension of vectors $\underline{\theta}^V$ and $\underline{\theta}^{\pi,i}$ is equal to the number of features used to represent action the respective function. $\underline{Z}$ has the same number of of the features as $\underline{\theta}^V$. The number of operations per computing cycle can be reduced setting activation threshold values or a maximum number of active features. Discrete RL methods usually disregard the importance of the initial value estimations, often learning policies from scratch after initializing them either randomly or with null values. In high dimensional continuous state-action spaces though, good initialization is critical for the success of the learning algorithm [12].

3.4. Multi-Objective RL

Conventional RL deals with a scalar reward function, therefore it is only suitable for single-objective control tasks. Multi-Objective Reinforcement Learning (MORL) methods [11,31,39,47], on the other hand, deal with sets of scalar reward functions. Each reward function usually defines one of the objectives to be maximized by the control algorithm. This approach suits well some real-world problems [42,52] because often the objectives cannot be independently maximized and they can even be conflicting. Although metaheuristic search methods such as genetic algorithms have been widely used to approach multi-objective problems [18,40,52], only a few instances of MORL can still be found in the literature. The taxonomy of *online model-free* MORL approaches given in [39], classifies them depending on whether they learn a single policy or multiple policies. Single policy uses an *scalarization function* [9,30] which is a weighted combination of the objectives, prioritizing them. Learning multiple policies [31] can be beneficial because it allows to produce a costumized scalar policy specified by a scalarization weight vector after the learning phase, but it is also computationally more expensive. In this paper, we have worked with single-policy learning using a preset vector of scalarization weights.

Under the *known weights* multi-objective scenario [30], the weights prioritizing the different objectives are known in advance. The control goals are introduced in the Multi-Objective RL framework in the form of a set of o reward functions $\{R_i(\underline{s})\}_{i=1}^{o}$. Thus, the agent receives a vector of reward values at each time step. This reward vector is scalarized using a *linearized scalarization function* [31] $R(\underline{s}) = \sum_{i=1}^{o} w_i \cdot R_i(\underline{s})$. In our experiments, we have used a set of linear reward functions $R_i(\underline{s})$, each depending on a specific state variable x_i:

$$R_i(\underline{s}) = 1 - \left|(x_i - x_i^*)/t_i\right|,$$

where $\underline{s} = [x_1 x_2 \dots x_n]$. The i-th reward signal gives a measure of how far the current value of variable x_i is from some predefined reference value x_i^*. The reward has a maximum value of 1 for $x_i = x_i^*$, decreasing linearly with the euclidean distance from x_j^*. It is positive within the tolerance region limited by t_j (expressed in the same units as the variable x_j^*), and values outside this tolerance area become increasingly negative to encourage the agent toward the tolerance region.

4. Baseline controller approximation

Because learning from scratch the control of such a high-dimensional non-linear system as a VSWT is unfeasible, we propose a two-step approach to build an Actor-Critic system improving a baseline controller: First, the Actor's policy is initialized using a VFA approximation of the baseline (either *Vidal* or *Boukhezzar*) controller's output. Secondly, the Actor-Critic agent is allowed to control the system for exploration and online learning of an improved controller configuration. We consider in this section the details of how the baseline controllers are approximated, we assume that the outputs of the baseline controller can be expressed as a set of policies $\left\{\pi^i(\underline{s}_i)\right\}_{i=1}^{m}$ each involving possibly different subsets of variables that span state subspaces $\{\underline{s}_i \in S_i\}_{i=1}^{m}$.

4.1. Distribution of the VFA center points

We approximate the controller using a set of Gaussian RBFs Eq. (12) for each state variable, with each i-th state variable's center points denoted by $c_{i,j}$, $j = 1, \dots, f_i$. We have used two different center point placement distribution functions $x : \mathrm{N} \in [0,1) \rightarrow \mathrm{R} \in [0,1]$ mapping the index j of a center point $c_{i,j}$ to a position along the desired range of values $[\min_i, \max_i\}]$:

Uniform distribution:

$$x^u(i) = \min_i + \left(i - \frac{1}{2}\right) \Big/ f_i \cdot (\max_i - \min_i). \quad (15)$$

Cubic distribution:

$$x^c(i) = \min_i + \left(\left(i - \frac{1}{2}\right) \Big/ f_i\right)^3 \cdot (\max_i - \min_i). \quad (16)$$

Error variables e_p and e_ω are best approximated with the cubic function. The bounds of these state variables are set $\min_i = -\max_i$, so that most of the center points are distributed in the vicinity of the zero error point. This allows a more accurate representation of the policy inside the tolerance region. Uniform distribution has been used for the rest of state variables, for which zero is not a distiguished value.

4.2. Weight initialization

The weights of each VFA feature dimension of the Actor $\theta^{\pi,i}$, $i = 1, \dots, m$ are initialized as follows, to approximate the output of the baseline controller:

$$\theta_j^{\pi,i} = \pi^i(\underline{s}),$$

where π_i is the i-th output variable of the policy, and the state $\underline{s}$ is the vector of center points associated with the i-th feature: $\underline{s} = \left[c_{1,\psi(1,i)} \dots c_{n,\psi(n,i)}\right]$. Because each output of the two baseline controllers depends on a different set of state variables, the set of state variables used by each output of the controller is also different:

- T_g, T_a, ω_r and e_p for the Boukhezzar torque controller Eq. (7).
- e_ω for the PI controller proposed by Boukhezzar Eq. (8).
- T_g, ω_r, $\dot{\omega}_r$ and e_p for the Vidal torque controller Eq. (9).
- e_ω and $e_\omega\, dt$ for the *PID* controller proposed by Vidal Eq. (10).

For the Critic estimation of the value function, we use the union of the sets of variables on which depend the Actor outputs: T_g, T_a, ω_r, e_p and e_w for the estimation of the value of the Boukhezzar controller; T_g, ω_r, $\dot{\omega}_r$, e_p, e_ω and $e_\omega dt$ for the Vidal controller.

5. Experiments

We conduct a set of experiments with the *Boukhezzar* and *Vidal* baseline controllers to assess the improvement provided by the Actor-Critic RL. We denote the baseline policies resulting from the *Boukhezzar* and *Vidal* baseline controllers as $\hat{\pi}_b$ and $\hat{\pi}_v$, respectively. In this section we will first comment on the precise parameter settings for the computational experiments, secondly we report the results achieved.

5.1. Experimental design

We first initialize an Actor whose policy approximates the baseline controller as described in Section 4. At each experimental run, the Actor performs 1000 episodes, each 360 s long, of interaction with the VSWT simulation model. Runs were repeated applying two different schedules for the exploration parameter σ (Section 3): a linearly decaying, and a constant value. We denote policies learned with a decaying σ and $\hat{\pi}_v^*$, and policies learned with constant σ and $\hat{\pi}_v^{**}$. In any case, the initial value is set to $10^{-5} \cdot (\max_i - \min_i\})$. In the case of the linear decay, the value of σ is updated at the end of each episode, ensuring that it reaches 0 in the final evaluation episode. The Actor's learning gain was fixed to $a = 0.1$ and the Critic's learning gain to $a = 0.01$ (no attempt was made to tune these parameters). The time step of the control algorithms is 0.01 s, and the simulation integration step is set to $2.5 \cdot 10^{-3}$ s.

The parameters of the baseline controllers must be tuned empirically, using as a starting point those referred to in [49]. The best Boukhezzar controller results were obtained with parameter values $c_0 = 10$, $K_p = 1$ and $K_i = 0$. The best Vidal torque controller performance indices were obtained with parameters $a = 1$ and $K_a = 6 \cdot 10^3$ used in combination with the PI controller proposed by Boukhezzar Eq. (8), instead of the original Vidal blade pitch controller Eq. (10).

Four reward signals are used to model the control goals, each one is a function of a different state variable x_i with tolerance value t_i. The four state variables associated with these rewards are those to be minimized according to the control objectives set in Section 2: e_p, e_ω, T_g and β. The baseline values of the state variables are $x_j^* = 0$, because in fact they model some form of error. The tolerance values are set equal to either the mean value (error variables) or the standard deviation (control variables) of values taken by these variables when the system is under the baseline controllers π_b and π_v. The weights of the scalarization function are all set to 1, because we give the same importance to all performance indices.

During the learning phase, we have only considered the 3 VFA features per state variable with the highest duration and other features are difficult to predict.

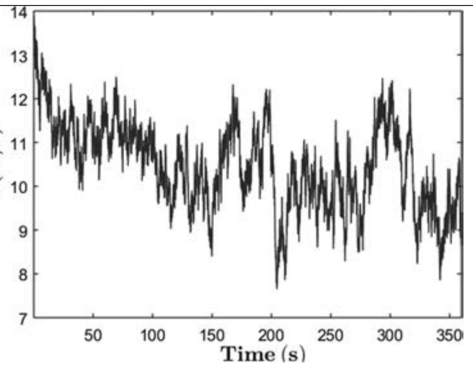

Fig. 2. Wind profile used for evaluation purposes in the computational experiments.

activation to calculate the feature vector because otherwise the number of updates required every time step by $\hat{\pi}_b^*$ is intractably high. This strategy reduces the number by $\hat{\pi}_b^{**}$ of updated features to $3n$. The number of VFA feature $\sigma =$ center values is set as follows:

- $\hat{\pi}_{T_g}^b(e_p, \omega, T_a, T_g)$ is approximated using 80 center values for each state variable (a total amount of 80^4 features).
- $\hat{\pi}_{T_g}^v(e_p, \omega, \dot{\omega}, T_g)$ is approximated using 80 center values for each state variable (a total amount of 80^4 features).
- $\hat{\pi}_\beta^b(e_\omega)$ and $\hat{\pi}_\beta^v(e_\omega)$ only depend on e_ω and thus might be approximated with a higher number of features: 10^4.
- The value functions $\hat{V}^b(e_p, \omega, T_a, T_g, e_\omega)$ and $\hat{V}^v(e_p, \omega, \dot{\omega}, T_g, e_\omega)$ are approximated with 50 center values for each state variable. Because e_ω is a function of ω, we neglect the latter, having a total number of 50^4 of features.

The parameters of the one-mass VSWT model match those of the *Controls Advanced Research Turbine* available at the *National Wind Technology Center* in Golden, Colorado (Table 1). The wind profiles were generated using TurbSim [20], a wind turbulence simulator commonly used in the literature. Seven different mean speeds were used to generate the wind profiles used in the learning episodes, and another is used for evaluation purposes. Before each learning episode, the wind profile was generated using a different random seed and randomly selecting one of the seven mean wind speeds (9, 9.5, 10, 10.5, 11, 11.5 and 12 m/s). On the other hand, the profile used in all the evaluation episodes was unique and had a mean wind speed of 10.25 m/s. The profile used in the evaluation episodes is plotted in Fig. 2. Note that it is a non stationary process, very noisy and with varying local trends whose

Table 1
Performance statistics of the Boukhezzar controller (π_b), the approximated policy without any learning ($\hat{\pi}_b$) and the policies learned with the two different exploration schedules ($\hat{\pi}_b^{*}$ and $\hat{\pi}_b^{**}$)

	π_b	$\hat{\pi}_b$	$\hat{\pi}_b^{*}$	$\hat{\pi}_b^{**}$
e_p	17.949 (± 27.540)	274.010 (± 2,326)	7.464 (± 25.661)	6.785 (± 23.559)
$\overline{r_1}$	−0.795	−26.401	−0.253	−0.321
e_ω	0.122 (± 0.095)	0.151 (± 0.138)	0.122 (± 0.093)	0.122 (± 0.096)
$\overline{r_2}$	−5.105	−6.587	−5.107	−5.092
β	0.131 (± 0.092)	0.140 (± 0.096)	0.130 (± 0.090)	0.131 (± 0.092)
$\overline{r_3}$	0.429	0.370	0.413	0.425
T_g	136,771 (± 4,931)	136,722 (± 6,440)	136,713 (± 4,807)	136,818 (± 4,886)
$\overline{r_4}$	0.695	0.596	0.686	0.694
$\sum_{i=1}^{4} w_i \cdot r_i$	−4.467	−4.438	−3.328	−3.285

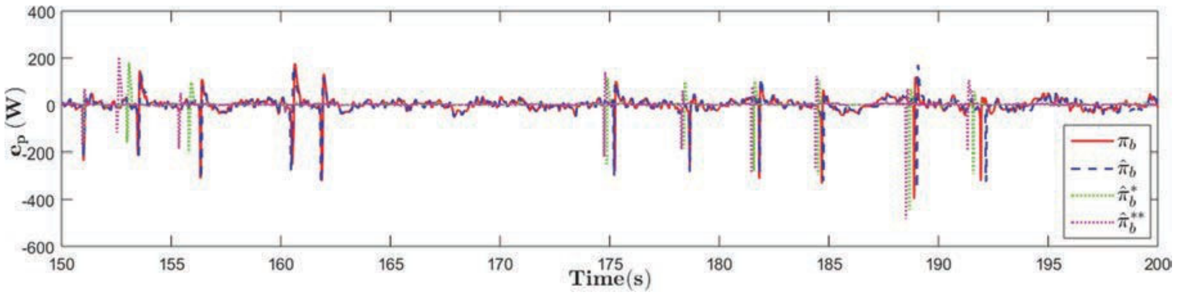
Fig. 3. Evolution of power error (e_p) in Watts in a complete episode for the baseline Boukhezzar controller π_b, the approximated $\hat{\pi}_b$ and the controllers learned using two different exploration schedules: $\hat{\pi}_b^{*}$ and $\hat{\pi}_b^{**}$.

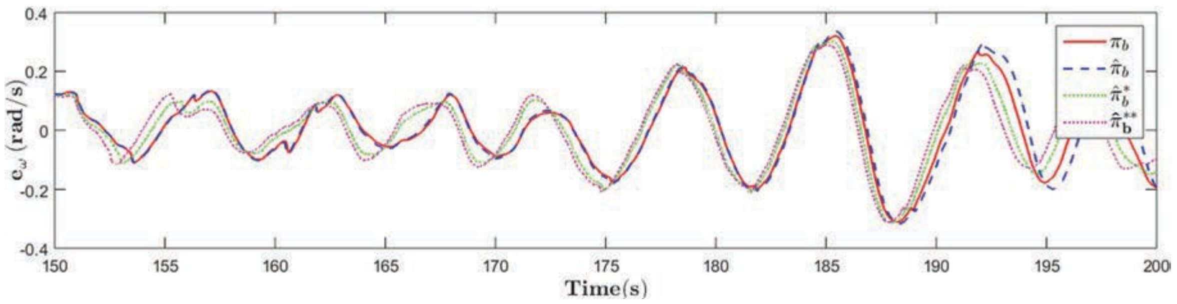
Fig. 4. Rotor speed error (e_ω) in rad/s in a complete episode for the baseline Boukhezzar controller π_b, the approximated $\hat{\pi}_b$ and the controllers learned using two different exploration schedules: $\hat{\pi}_b^{*}$ and $\hat{\pi}_b^{**}$.

5.2. Results

5.2.1. Performance measures

In order to compare the performance of the different controllers, we have calculated the following statistics ($|x|$ denotes the sum of the absolute values):

- $|\overline{e_p}|$ to measure the power control quality,
- $|\overline{e_\omega}|$ to measure the rotor angular speed control quality,
- $std(\beta)$ and $std(T_g)$ to measure the load effected on the control signals.

5.2.2. Boukhezzar controller

Figures 3 and 4 plot the values of e_p and e_ω during an evaluation episode 360 s long. Figures 5 and 6 plot the outputs of the controllers (T_g and β). The differences between the controllers cannot be easily ascertained in some cases and Figs 3–6 show the little adjustments performed by the learning algorithm to the original behaviour of the controller. After the learning phase, the controllers learned show small differences that improve the original controller. Because these differences are the result of random exploration instead of an analytical reasoning process, they need not follow

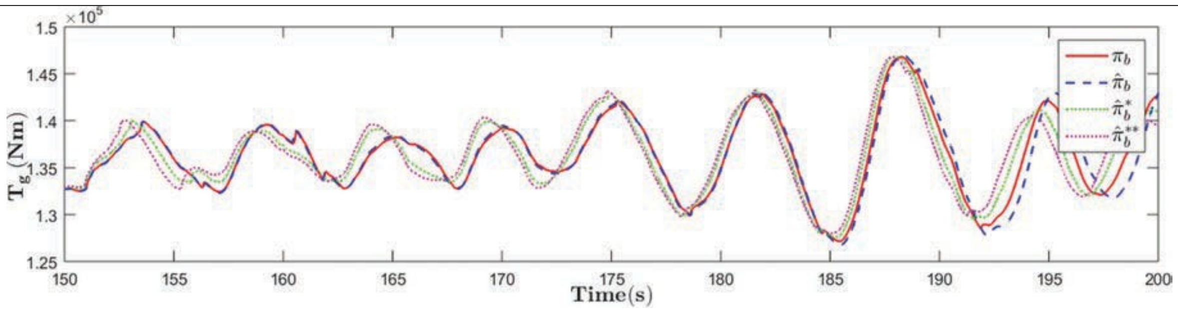

Fig. 5. Generator torque (T_g) in N·m in a complete episode for the baseline Boukhezzar controller π_b, the approximated $\hat{\pi}_b$ and the controllers learned using two different exploration schedules: $\hat{\pi}_b^{□}$ and $\hat{\pi}_b^{□□}$.

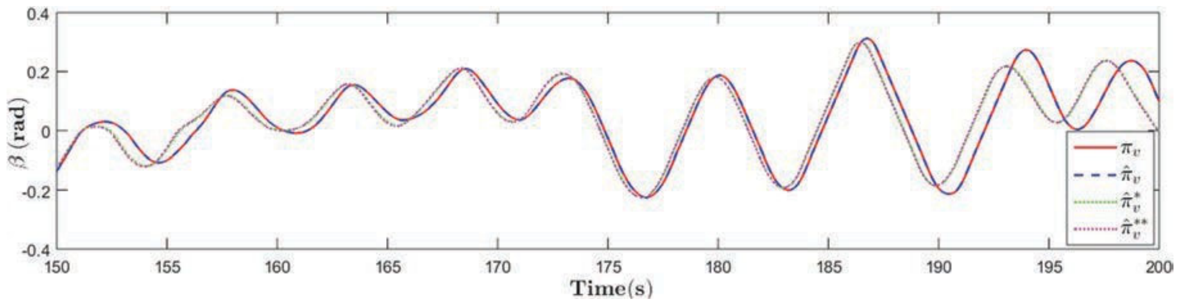

Fig. 6. Blade pitch angle (β) in rad in a complete episode for the baseline Boukhezzar controller π_b, the approximated $\hat{\pi}_b$ and the controllers learned using two different exploration schedules: $\hat{\pi}_b^{□}$ and $\hat{\pi}_b^{□□}$.

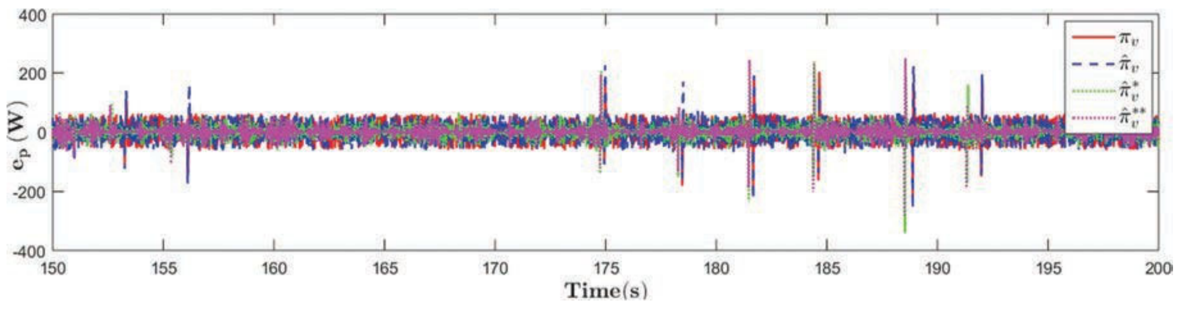

Fig. 7. Power error (e_p) in Watts in a complete episode for the baseline Vidal policy π_V , its functional approximation $\hat{\pi}_V$, and RL tuned policies $\hat{\pi}_V^{□}$ and $\hat{\pi}_V^{□□}$ using two different exploration schedules: $\hat{\pi}_V^{□}$ and $\hat{\pi}_V^{□□}$.

an underlying logic. The results can be better understood from the statistics given in Table 1.

First conclusion, after analysis of the plots and the performance statistics, is that the VFA model $\hat{\pi}_b$ is, indeed, a good approximation of the baseline controllers π_b: both Figs 3 and 4 show that the policies' outputs are very similar. Around $t = 300$, the rotor speed has transitory oscillations, and eventually makes the power drop when the rotor speed reaches its local minimum, but otherwise, the behavior of the approximated actor mimics quite effectively the output of π_b. Observation of Fig. 3 and the values of $|e_p|$ shows that the power error incurred by the approximated actor is an order of magnitude greater because of this power drop.

The power error is a complex non-linear function of the control variables and might be expected to amplify the output differences. Another conclusion looking at Fig. 3 is that the policies derived by Actor-Critic RL produce a more stable output. In Table 1, the performance statistics show that the CACLA tuned controllers $\hat{\pi}_b^*$ and $\hat{\pi}_b^{**}$ have a lower mean power error: respectively, $|e_p|$ is reduced by factors 0.584 and 0.621 relative to the error achieved by the baseline controller.

On the other hand, the rotor speed error e_ω is very similar except for the approximated policy, and the differences can otherwise be neglected. The differences

Table 2
Performance statistics of the Vidal controller (π_b), the approximated policy without any learning ($\hat{\pi}_b$) and the policies learned with the two different exploration schedules ($\hat{\pi}_b^{*}$ and $\hat{\pi}_b^{**}$)

	π_V	$\hat{\pi}_V$	$\hat{\pi}_V^{*}$	$\hat{\pi}_V^{**}$
e_p	24.955 (± 21.837)	24.534 (± 22.224)	16.629 (± 20.531)	13.295 (± 19.430)
$\overline{r_1}$	−1.495	−1.453	−0.663	−0.330
e_ω	0.121 (± 0.095)	0.122 (± 0.095)	0.124 (± 0.097)	0.121 (± 0.096)
$\overline{r_2}$	−5.074	−5.086	−5.218	−5.070
β	0.130 (± 0.090)	0.130 (± 0.090)	0.131 (± 0.092)	0.131 (± 0.092)
$\overline{r_3}$	0.421	0.420	0.425	0.425
T_g	136,718 (± 4,836)	136,721 (± 4,843)	136,789 (± 4,947)	136,832 (± 4,870)
$\overline{r_4}$	0.681	0.681	0.676	0.688
$\sum_{i=1}^{4} w_i \cdot \overline{r_i}$	−4.467	−4.438	−3.328	−3.285

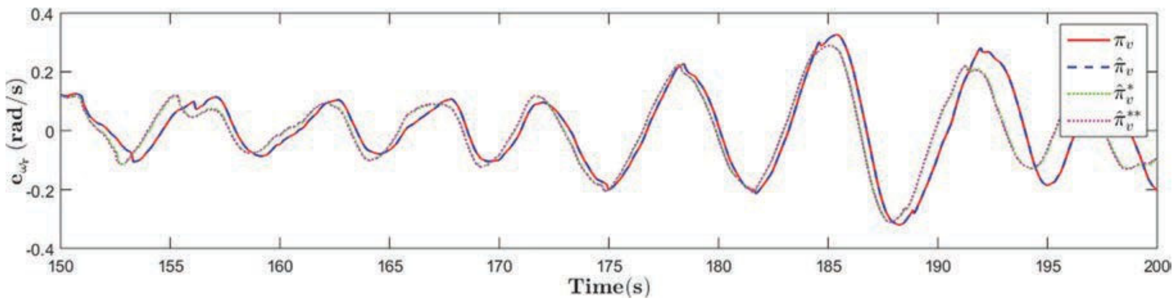

Fig. 8. Rotor speed error (e_ω) in rad/s in a complete episode for the baseline Vidal policy π_V , its functional approximation $\hat{\pi}_V$, and RL tuned policies $\hat{\pi}_V^{*}$ and $\hat{\pi}_V^{**}$ using two different exploration schedules: $\hat{\pi}_V^{*}$ and $\hat{\pi}_V^{**}$.

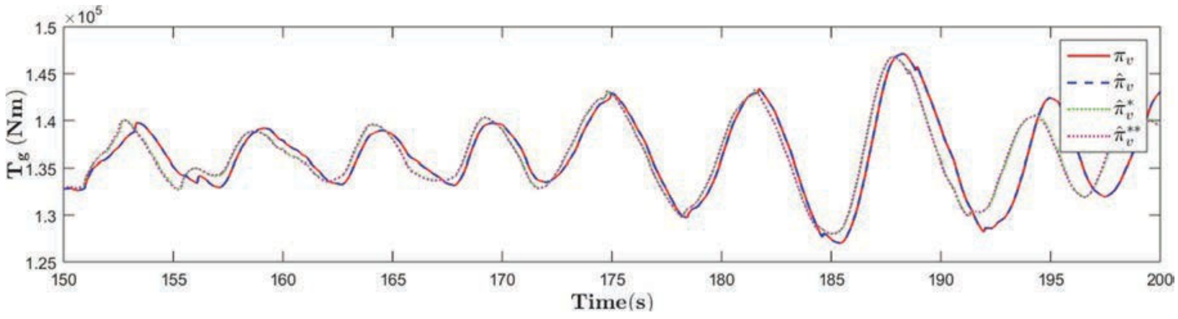

Fig. 9. Generator torque (T_g) in $N \cdot m$ in a complete episode for the baseline Vidal controller π_V, the approximated $\hat{\pi}_V$ and the controllers learned using two different exploration schedules: $\hat{\pi}_V^{*}$ and $\hat{\pi}_V^{**}$.

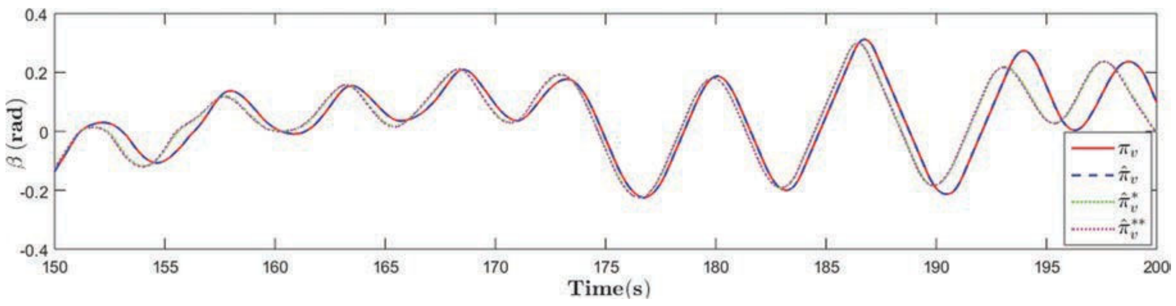

Fig. 10. Blade pitch angle (β) in rad in a complete episode for the baseline Vidal controller π_V, the approximated $\hat{\pi}_V$ and the controllers learned using two different exploration schedules: $\hat{\pi}_V^{*}$ and $\hat{\pi}_V^{**}$.

Table 3
Parameters used in our experiments

Air density	ρ	1.29 kg·m
Turbine external damping	K_t	400 N·m/rad
Turbine inertia	J_t	$3.92 \cdot 10^5$ kg·m^2
Nominal electrical power	P_{nom}	600 kW
Nominal rotor speed	ω_{nom}	42 rpm
Tower height	h	36.6 m
Blade pitch	β	$[-5, 30]$ deg.
Generator torque	T_g	$[0, 162]$ N ·m
Blade pitch angular speed	$\dot{\beta}$	$[-10, 10]$ deg/s

among the standard deviation of the two control signals are also very small. In all the cases, the VFA approximated policy gets slightly worse results than the baseline policy. The CACLA improved policies are both very similar to the original controller. The mean reward values support these results, showing that they are aligned with the performance measures defined: the higher the reward value, the higher the performance index. The total sum of the different mean rewards also shows that the learning algorithm is able to produce better policies with respect to the reward functions and the scalarization function used in the experiment: the sum of the average rewards show -2.753 and -2.651 against the -3.776 scored by the baseline controller. None of the CACLA learned policies improves on both objectives the original controller, but the quality loss in rotor speed is compensated by the great improvement in electrical power control.

5.2.3. *Vidal controller*

We have plotted in Figs 7 and 8 the error variables measured during the evaluation episode of the original Vidal controller, the VFA approximated policy, and the policies learned by CACLA. Figures 9 and 10 show the values of the control variables, and Table 2 displays the performance statistics of each controller.

Surprisingly, the policy VFA $\hat{\pi}_V$ has a slightly lower rotor speed error $|e_\omega|$ than the original controller. Nevertheless, the rest of the performance scores in Table 2 are worse (though surprisingly similar) for the approximated controller as expected.

As in the previous experiment, CACLA learned policy does not improve the baseline controller in all objective indices. However, $\hat{\pi}_V^*$ outperforms the baseline controller with respect to two of the four performance indices (with small decreases in the other two), and π_V^{**} outperforms the base controller with respect to three of the four indices. An interesting observation suggesting robustness of the CACLA approach is that the exploration schedule does not seem to have a great influence on the learning results. The constant gain schedule obtains better results in the two experiments, but the differences are not statistically significant.

We recall that the performance improvement is measured during the learning phase via a linear scalarization function. This means that the improvement can be guided by the user by setting different reward weights w_i or different tolerance values t_i. These values should be chosen depending on the specific preferences of the system designer. For lack of space, we have only reported the results obtained using equal weights for all the objectives and tolerances based on the performance of the baseline controllers, but different weights and tolerances can be easily set to change the priorities between the control objectives.

6. Conclusions

In this paper we present a novel approach to improve an existing controller using model-free online scalarized Multi-Objective Reinforcement Learning. First, a baseline controller is approximated using Value Function Approximation, and then, this approximation is adapted online by an actor-critic RL agent using the scalarized multi-objective reward function. In our experiments, we have used an instance of Actor-Critic learning: the Actor implements a policy and the Critic estimates this policy's value. By means of exploring outputs different from the actor's policy and observing the critic's value updates, the system is able to learn better control solutions.

We have carried out computational experiments of this approach on a one-mass mathematical model of a Variable-Speed Wind Turbine, successfully improving two different controllers from the literature [7,49]. The results show that the performance of the RL tuned controllers improves significantly the baseline controllers in a tough non-stationary wind scenario test, achieving adaptation to the changing conditions. This improvement is achieved without fine tuning of the RL learning parameters. This suggests that there is even further room for improvement. The VSWT control is a very challenging application of this approach because of the complex underlying dynamic system. Using RL for online adaptation opens some interesting possibilities, such as adding other relevant measured variables unused by the baseline controller (i.e., the wind speed). Besides, model-free RL-based controllers are not prone to underperform because of an inaccurate model, and they can also seamlessly benefit from additional techniques such as noise filtering of the input variables [19]. Our future work will focus on learning multiple policies simultaneously and bringing thus the problem from the *known weights* scenario to the more complex and even more appealing *unknown weights* scenario [30], that allows the user to set the weights after the learning process, thus virtually providing a solution for any set of weights. Another venue of improvement to our approach would be to devise an automatic method to decide the number of features per state variable and their distribution [10].

Acknowledgments

GIC participates at UIF 11/07 of UPV/EHU. The Computational Intelligence Group is funded by the Basque Government with grant IT874-13. Manuel Graña was supported by EC under FP7, Coordination and Support Action, Grant Agreement Number 316097, ENGINE European Research Centre of Network Intelligence for Innovation Enhancement.

References

[1] H. Adeli and S.L. Hung, Machine learning-neural networks, *Genetic Algorithms, and Fuzzy Systems Wiley J and Sons*, eds, John Wiley and Sons, (1995).

[2] H. Adeli and H. Kim, *Wavelet-based Vibration Control of Smart Buildings and Bridges Francis Tand*, ed., Boca Raton, Florida: CRC Press; (2009).

[3] H. Adeli and H. Kim, Wavelet-hybrid feedback least mean square algorithm for robust control of structures, *Journal of Structural Engineering* **130**(1) (2004), 128–137.

[4] F. Amini and M. Zabihi-Samani, A wavelet-based adaptive pole assignment method for structural control, *Computer-Aided Civil and Infrastructure Engineering* **29**(6) (2014), 464–477.

[5] B. Beltran, T. Ahmed-Ali and M. Benbouzid, High-order sliding-mode control of variable-speed wind turbines, *Industrial Electronics, IEEE Transactions on* **56**(9) (Sep 2009), 3314–3321.

[6] S. Bhatnagar, R. Sutton, M. Ghavamzadeh and M. Lee, Natural actor-critic algorithms, *Automatica, International Federation of Automatic Control* **45**(11) (2009), 2471–2482.

[7] B. Boukhezzar, L. Lupu, H. Siguerdidjane and M. Hand, Multivariable control strategy for variable speed, variable pitch wind turbines, *Renewable Energy* **32**(8) (2007), 1273–1287.

[8] B. Boukhezzar and H. Siguerdidjane, Nonlinear control of a variable-speed wind turbine using a two-mass model, *Energy Conversion, IEEE Transactions on* **26**(1) (Mar 2011), 149–162.

[9] T. Brys, K.V. Moffaert, K.V. Vaerenbergh and A. Nowé, On the behaviour of scalarization methods for the engagement of a wet clutch, in: *International Conference on Machine Learning and Applications (ICMLA)*, IEEE, (2013).

[10] L. Bussoniu, R. Babuska, B.D. Schutter and D. Ernst, *Reinforcement Learning and Dynamic Programming Using Function Approximators*, CRC Press, (2010).

[11] M.M. Drugan and A. Nowe, Designing multi-objective multi-armed bandits algorithms: A study, in: *The 2013 International Joint Conference on Neural Networks (IJCNN)* (Aug 2013), 1–8.

[12] B. Fernandez-Gauna, J.L. Osa and 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*, Springer International Publishing, (2014), 125–133.

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

[14] K.J. Gurubel, A.Y. Alanis, E.N. Sanchez and S. Carlos, A neural observer with time-varying learning rate: Analysis and applications, *International Journal of Neural Systems* **24**(1) (2014), 1450011 (15 pages).

[15] T. Hanselmann, L. Noakes and A. Zaknich, Continuous-time adaptive critics, *Neural Networks, IEEE Transactions on* **18**(3) (May 2007), 631–647.

[16] J. Huo, Y. Gao, W. Yang and H. Yin, Multi-instance dictionary learning for detecting abnormal event detection in surveillance videos, *International Journal of Neural Systems* **24**(3) (2014), 1430010 (15 pages).

[17] X. Jiang and H. Adeli, Dynamic fuzzy wavelet neuroemulator for nonlinear control of irregular highrise building structures, *International Journal for Numerical Methods in Engineering* **74**(7) (2008), 1045–1066.

[18] L. Jia, Y. Wang and L. Fan, Multiobjective bilevel optimization for production-distribution planning problems using hybrid genetic algorithm, *Integrated Computer-Aided Engineering* **21**(1) (2014), 77–90.

[19] J.M. Jonkman, S. Butterfield, W. Musial and G. Scott, Definition of a 5-MW reference wind turbine for offshore system development, *Tech Rep Golden, CO*, USA: National Renewable Energy Laboratory, (2009).

[20] N. Kelley and B. Jonkman, TurbSim web site, [Online], 2015. Available from: https://nwtc.nrel.gov/TurbSim.

[21] N. Khezami, N.B. Braiek and X. Guillaud, Wind turbine power tracking using an improved multimodel quadratic approach, *ISA Transactions* **49**(3) (Jul2010), 326–334.

[22] H. Kim and H. Adeli, Hybrid control of smart structures using a novel wavelet-based algorithm, *Computer-Aided Civil and Infrastructure Engineering* **20**(1) (2005), 7–22.

[23] H. Kim and H. Adeli, Hybrid feedback-least mean square algorithm for structural control, *Journal of Structural Engineering* **130**(1) (2004), 120–127.

[24] H. Kim and H. Adeli, Wind-induced motion control of 76-story benchmark building using the hybrid damper-tuned liquid column damper system, *Journal of Structural Engineering* **131**(12) (2005), 1794–1802.

[25] K.H. Kim, T.L. Van, D.C. Lee, S.H. Song and E.H. Kim, Maximum output power tracking control in variable-speed wind turbine systems considering rotor inertial power, *Industrial Electronics, IEEE Transactions on* **60**(8) (Aug 2013), 3207–3217.

[26] J.Z. Kolter, Z. Jackowski and R. Tedrake, Design, analysis, and learning control of a fully actuated micro wind turbine, in: *American Control Conference (ACC) 2012* (Jun 2012), 2256–2263.

[27] M. Kwon, S. Kavuri and M. Lee, Action-perception cycle learning for incremental emotion recognition in a movie clip

using 3d fuzzy gist based on visual and eeg signals, *Integrated Computer-Aided Engineering* **21**(3) (2014), 295–310.

[28] M.G. Lagoudakis and R. Parr, Least-squares policy iteration, *Journal of Machine Learning Research* **4** (2003), 1107–1149.

[29] A. Merabet, J. Thongam and 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* (May 2011), 1–5.

[30] K.V. Moffaert, M.M. Drugan and A. Nowé, Scalarized multi-objective reinforcement learning: Novel design techniques, in: *Proceedings of the 2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, Singapore, (2013).

[31] K.V. Moffaert and A. Nowé, Multi-objective reinforcement learning using sets of pareto dominating policies, *Journal of Machine Learning Research* **15** (2014), 3483–3512.

[32] H. Morais et al., Coalition of distributed generation units to virtual power players – a game theory approach, *Integrated Computer-Aided Engineering* **22**(3) (2015), 297–309.

[33] S.S. Murthy, B. Singh, P.K. Goel and S.K. Tiwari, A comparative study of fixed speed and variable speed wind energy conversion systems feeding the grid, in: *7th International Conference on Power Electronics and Drive Systems 2007* (Nov 2007), 736–743.

[34] S.M. Muyeen, J. Tamura and T. Murata, *Stability augmentation of a grid-connected wind farm (green energy and technology) ackerman*, ed., Springer, (2008).

[35] A. Nedic and D.P. Bertsekas, Least squares policy evaluation algorithms with linear function approximation, *Discrete Event Dynamic Systems* **13**(1–2) (2003), 79–110.

[36] T. Pinto et al., Adaptive learning in agents behaviour: A framework for electricity markets simulation, *Integrated Computer-Aided Engineering* **21**(4) (2014), 399–415.

[37] M. Reischl, L. Gröll and R. Mikut, Evaluation of data mining approaches for the control of multifunctional arm prostheses, *Integrated Computer-Aided Engineering* **18**(3) (2011), 235–249.

[38] G. Rigatos, Adaptive fuzzy control for differentially flat MIMO nonlinear dynamical systems, *Integrated Computer-Aided Engineering* **20**(2) (2013), 111–126.

[39] D. Roijers, P. Vamplew, S. Whiteson and R. Dazeley, A survey of multi-objective sequential decision-making, *Journal of Artificial Intelligence Research* **48** (2013), 67–113.

[40] J.O.M. Rubio and L.T. Aguilar, Maximizing the performance of variable speed wind turbine with nonlinear output feedback control, *Procedia Engineering* **35** (2012), 31–40.

[41] A. Saleh and H. Adeli, Optimal control of adaptive building structures under blast loading, *Mechatronics* **8**(8) (1998), 821–844.

[42] A. Salkham, R. Cunningham, A. Garg and V. Cahill, A collaborative reinforcement learning approach to urban traffic control optimization, in: *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, Washington, DC, USA: IEEE Computer Society **2** (2008), 560–566.

[43] A.M. Schäfer and S. Udluft, Solving partially observable reinforcement learning problems with recurrent neural networks, in: *Workshop Proc of the European Conference on Machine Learning* (2005).

[44] M. Sedighizadeh and A. Rezazadeh, Adaptive PID controller based on reinforcement learning for wind turbine control, *World Academy of Science, Engineering and Technology* **37** (2008), 257–262.

[45] N. Siddique and H. Adeli, Computational intelligence – synergies of fuzzy logic, *Neural Networks and Evolutionary Computing Wiley*, ed., West Susse, United Kingdom: Wiley, (2013).

[46] R.S. Sutton and A.G. Barto, Reinforcement learning: An introduction, MIT Press, (1998).

[47] P. Vamplew, R. Dazeley, A. Berry, R. Issabekov and E. Dekker, Empirical evaluation methods for multiobjective reinforcement learning algorithms, *Machine Learning* **1** (2011), 1–30.

[48] H. van Hasselt, Reinforcement learning: State of the art, in: M. Wiering and M. van Otterlo, eds, Springer (2011), 207–246.

[49] Y. Vidal, L. Acho, N. Luo, M. Zapateiro and F. Pozo, Power control design for variable-speed wind turbines, *Energies* **5**(8) (2012), 3033–3050.

[50] X. Xu, L. Zuo and Z. Huang, Reinforcement learning algorithms with function approximation: Recent advances and applications, *Information Sciences* **261** (2014), 1–31.

[51] Y.B. Yang, Y.N. Li, Y. Gao, H.J. Yin and Y. Tang, Structurally enhanced incremental neural learning for image classification with subgraph extraction, *International Journal of Neural Systems* **24**(7) (2014), 1450024 (13 pages).

[52] Z. Zhu, J. Xiao, J.Q. Li, F. Wang and Q. Zhang, Global path planning of wheeled robots using multi-objective memetic algorithms, *Integrated Computer-Aided Engineering* **22**(4) (2015), 387–404.

[53] J. Zhang, M. Cheng, Z. Chen and X. Fu, Pitch angle control for variable speed wind turbines, in: *Electric Utility Deregulation and Restructuring and Power Technologies* (Apr 2008), 2691–2696.