Saenz-Aguirre, A., Zulueta, E., Fernandez-Gamiz, U., Ramos-Hernanz, J., Lopez-Guede, J. (2021). *Self-tuning Yaw Control Strategy of a Horizontal Axis Wind Turbine Based on Machine Learning*. In: Mahdavi Tabatabaei, N., Bizon, N. (eds) **Numerical Methods for Energy Applications.** Power Systems**.** Springer, Cham. This version of the article has been accepted for publication, after peer review (when applicable) and is subject to Springer Nature's AM terms of use, but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: https://doi.org/10.1007/978-3-030-62191-9_32

Self-Tuning Yaw Control Strategy of an Horizontal Axis Wind Turbine based on Machine Learning

Aitor Saenz-Aguirre, Ekaitz Zulueta, Unai Fernandez-Gamiz
Jose Antonio Ramos-Hernanz and Jose Manuel Lopez-Guede

Abstract The design procedure of a Machine Learning (ML) based yaw control strategy for an Horizontal Axis Wind Turbine (HAWT) is presented in the following chapter. The proposed yaw control strategy is based on the interaction of three different Artificial Intelligence (AI) techniques to design a ML system: Reinforcement Learning (RL), Artificial Neural Networks (ANN) and metaheuristic optimization algorithms. The objective of the designed control strategy is to achieve, after a training stage, a fully autonomous performance of the wind turbine yaw control system for different input wind scenarios while optimizing the electrical power generated by the wind turbine and the mechanical loads due to the yaw rotation. The RL algorithm is known to be able to learn from experience. The training process could be carried out online with real-time data of the operation of the wind turbine or offline, with simulation data. The use of an ANN to store the data of the matrix *Q(s,a)* related to the RL algorithm eliminates the large scale data management and simplifies the operation of the proposed control system. Finally, the implementation of a metaheuristic optimization algorithm, in this case a Particle Swarm Optimization (PSO) algorithm, allows calculation of the optimal yaw control action that responds to the compromise between the generated power increment and the mechanical loads increase due to the yaw actuation.

Keywords Wind Turbine Control; Yaw Control; Reinforcement Learning; Artificial Neural Network; Optimization; Pareto Front.

A. Saenz-Aguirre (✉) · E. Zulueta · J.M. Lopez-Guede
Automatic control and System Engineering Dep., University of the Basque Country, Nieves Cano 12, 01006 Vitoria-Gasteiz, Araba, Spain
email: asaenz012@ehu.eus

E. Zulueta
email: ekaitz.zulueta@ehu.eus

J.M. Lopez-Guede
email: jm.lopez@ehu.eus

U. Fernandez-Gamiz
Nuclear Engineering and Fluid Mechanics Dep., University of the Basque Country, Nieves Cano 12, 01006 Vitoria-Gasteiz, Araba, Spain
email: unai.fernandez@ehu.eus

J.A. Ramos-Hernanz
Electrical Engineering Dep., University of the Basque Country, Nieves Cano 12, 01006 Vitoria-Gasteiz, Araba, Spain
email: josean.ramos@ehu.eus

Abbreviation and Acronyms

ML	Machine Learning
HAWT	Horizontal Axis Wind Turbine
AI	Artificial Intelligence
RL	Reinforcement Learning
ANN	Artificial Neural Network
PSO	Particle Swarm Optimization
LCOE	Levelized Cost of Energy
MLP-BP	MultiLayer Perceptron with BackPropagation
MDP	Markov Decision Process
DP	Dynamic Programming
MC	Monte Carlo
TD	Temporal Differences
PoF	Pareto optimal Front
GA	Genetic Algorithm
DE	Differential Evolution
PID	Proportional Integral Derivative
FAST	Fatigue, Aerodynamics, Structure and Turbulence
NREL	National Renewable Energies Laboratory
MSE	Mean Squared Error
DM	Decision Making

III-16.1. Introduction

The gradual depletion of the fossil fuels and the atmospheric pollution originated by their combustion have brought an important growth of the renewable energy generation systems. Plus, as a result of the yearly increasing electrical power consumption, the research work with the objective of enhancing the efficiency of renewable energy systems and maximize their power production has been placed on the focus of many research institutes and universities [1].

The most important renewable energy generation source nowadays is the wind energy. Many studies showing the positive tendency of the wind energy these days can be found in the literature. For example, according to some studies presented by Rosales-Asensio et al. [2], the sustainable power production with wind origin in Denmark achieved a 40% of the power produced in the country in 2015. This same value was quite smaller in Spain, with a 17% in 2015, but having raised from a 10.4% in 2007. More recent studies elaborated by WindEurope [3], and summarized in Figure III-16.1, show remarkable increments in the wind energy installed power in 2018 especially in four countries: a 29% in Germany, a 16% in the United Kingdom, a 13% in France and a 6% in Sweden. All these data indicate the importance of the wind energy to lead the change of the electrical power generation structures towards a sustainable generation in the coming years.

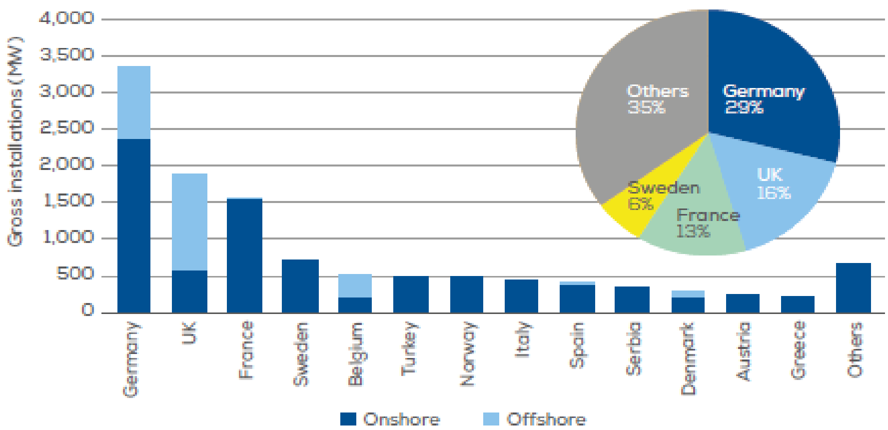


Figure III-16.1. Wind energy generation increase during year 2018 in some European countries [3]

The power generation increase in wind energy systems is tightly related to the investigation work carried out to reduce the Levelized Cost of Energy (LCOE) of the wind turbines, which encourages capital investment in the sector. An optimization exercise to reduce the LCOE of a 10 MW wind turbine is presented in the work of Nyanteh et al. [4]. One main topic of this research work is the development of advanced control strategies to optimize the performance of the wind turbines. In this context, a robust H∞ controller to enhance the operation and reduce the mechanical loads of a wind turbine is presented by Kim et al. [5]. In the work of Merabet et al. [6] a Sliding Mode Control strategy is introduced to the control system of a wind turbine.

In this chapter, the design procedure of a yaw control system of an Horizontal Axis Wind Turbine (HAWT) based on Machine Learning (ML) is presented. The designed intelligent control system is based on the interrelation of a Reinforcement Learning (RL) algorithm, detailed in the work of Watkins et al. [7], an Artificial Neural Network (ANN) in form of a MultiLayerPerceptron with BackPropagation (MLP-BP), presented by Erdogmus et al. [8], and a Particle Swarm Optimization (PSO) metaheuristic algorithm, introduced in the work of Ho et al. [9].

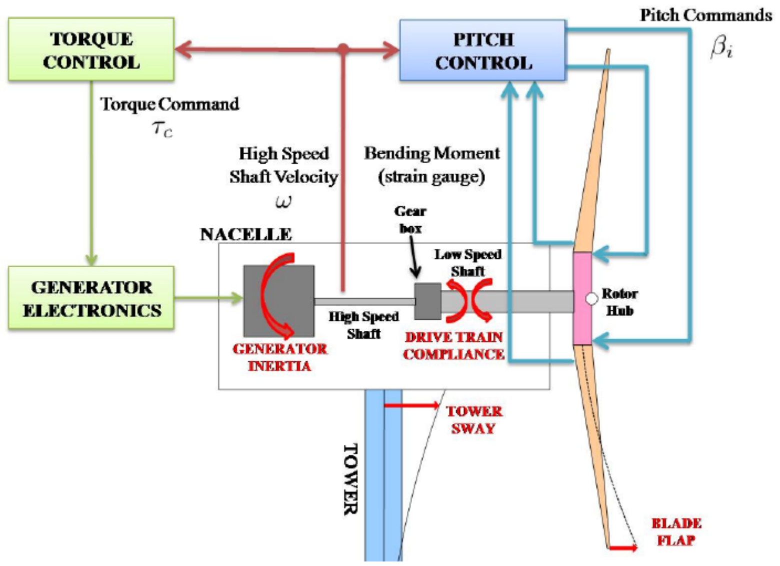
The objective of the ML based control strategy developed in this chapter is to achieve a fully autonomous performance of the yaw system of the wind turbine based on its own experience, which could be acquired via an offline training, i.e., when the wind turbine is paused, or an online training, i.e., during operation of the wind turbine. An offline training process is proposed in this chapter. However, a continuous online training process with real data acquired during operation of the wind turbine to continuously learn from experience could be implemented as well. The MLP-BP is used to store the data of the matrices $Q(s,a)$ related to the RL algorithm and manage them as continuous functions, $Q(s(t),a(t))$. This process avoids quantification and large data management problems. The combination of an RL strategy and an ANN is widely known as Deep Reinforcement Learning [10,11]. An example of the RL and ANN based yaw control strategy to autonomously maximize the power generated by a wind turbine is presented in the work of Saenz-Aguirre et al. [12]. Finally, with the introduction of additional features based on the multivariable PSO optimization algorithm, an increment of the power generated by the wind turbine with a considerable reduction of the mechanical loads due to the yaw rotation is expected to be achieved.

This chapter is structured as follows: the objectives and applications of the proposed yaw control strategy are presented in Section III-16.2. Section III-16.3 details the theoretical basis of the different Artificial Intelligence (AI) techniques used to design the ML system. The design procedure of the yaw control system based on ML is exposed in Section III-16.4. Finally, Section III-16.5 presents the conclusions.

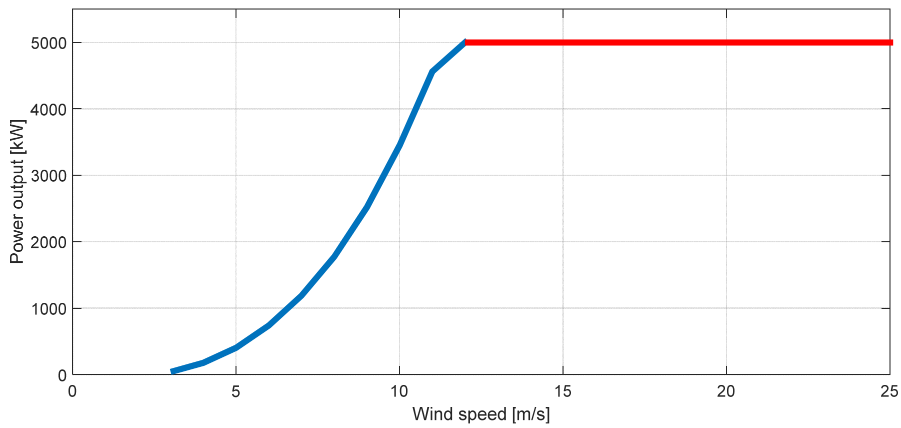
III-16.2. Objectives and Applications

The main factor that determines the power output of a wind turbine is the wind incident to its rotor. However, the wind is originated as a result of very complex meteorological processes, which, as stated by Bivona et al. [13], are very complex to model, and can, thus, suffer from unpredictable important variations. Some wind gusts can even exceed the safe wind speed operation range of the wind turbine and endanger its correct performance. To avoid this issue, a control system is implemented in the wind turbines.

The control system of a wind turbine is formed by different control strategies designed to regulate the rotational speed of the rotor in the whole range of operating points of the wind turbine. A scheme of the different control loops oriented to regulate the rotational speed of the rotor is presented in Figure III-16.2 (a). As a result of these control strategies, the power output of the wind turbine is predefined for the whole range of wind speed values in which the turbine operates. The curve that relates the power output of the wind turbine with the wind speed is known as the power curve. The power curve of the NREL 5MW wind turbine, presented in the work of Jonkman et al. [14], is illustrated in Figure III-16.2 (b).



(a)



(b)

Figure III-16.2. (a) Scheme of the torque and pitch control strategies of a wind turbine [15] (b) Power curve of the NREL 5MW wind turbine

There are two main control loops to regulate the rotational speed of the wind turbine: The torque loop and the pitch loop. Each one of them is active in a different zone of the power curve. The torque loop, explained in detail in the work of Harris et al. [16], is active in the partial power zone of the power curve, plotted in blue color in Figure III-16.2 (b). On the other hand, in the rated power zone, plotted in red color in Figure III-16.2 (b), the objective is to reduce the power received by the wind turbine from the wind by means of the pitch control, explained in the work of Harris et al. [16].

The main control objective in the partial power zone is to maximize the power the wind turbine extracts from the wind, which can be expressed as in Eq. (III-16.1).

$$P_{opt} = \frac{1}{2} \cdot \rho \cdot C_P \cdot A \cdot v^3 \quad [W] \qquad \textit{(III-16.1)}$$

where ρ [kg/m^3] is density of the air, C_P [-] is the power coefficient, A [m^2] is the area covered by the rotor and v is the wind speed.

However, in order to express the real power the wind turbine extracts from the wind, the misalignment between the incident wind and the rotor must be considered, commonly known as the yaw angle. The expression is shown in Eq. (III-16.2).

$$P = P_{opt} \cdot cos^3(\theta_{yaw}) \ [W] \qquad \textit{(III-16.2)}$$

where θ_{yaw} is the yaw angle.

As it can be observed in Eq. (III-16.2), a correct alignment of the wind turbine with the direction of the incident wind can make the power generated by the wind turbine increase considerably. The control system that allows a correct alignment of the wind turbine with respect to the incident wind is the yaw control. A detailed explanation about the yaw control system of a 5 kW wind turbine is introduced in the work of Yücel et al. [17]. Hence, an adequate design of the yaw control strategy of the wind turbine can be translated into a considerable increment of the power generated by the system.

On the other hand, as a result of the high inertia values of the mechanical components that participate in the yaw rotation, remarkable mechanical loads arise in different elements of the wind turbine. The physical effect that explains these loads is known as the gyroscopic effect. An study of possible control strategies aimed to attenuate the high mechanical loads resulting from the gyroscopic effect are presented in [18,19]. Additionally, an analysis of the mechanical loads generated as a consequence of the yaw rotation is presented in the work of Shariatpanah et al. [20].

As a result, an adequate design of the yaw control strategy allows not only maximization of the power generated by the wind turbine, but also reduction of the mechanical loads in several elements of the wind turbine, and, thus, to increment its lifetime.

The objectives of the proposed yaw control strategy are:

- Achieve a fully autonomous and self-tuning yaw control strategy to be implemented in the wind turbine.
- Design a control strategy based on ML that can continuously learns from its own experience.
- Selection of the optimal yaw control action (maximal power and minimal loads possible) for every possible scenario of the wind turbine operation.

The main applications of the designed yaw control strategy are:

- Increment of the power produced by the wind turbine, with the consequent enhancement of its efficiency, and the reduction of the LCOE.
- Reduction of the mechanical loads originated as a result of the yaw rotation, with the consequent increment of the lifetime of the mechanical components of the wind turbine, and the reduction of the LCOE.

III-16.3. Machine Learning and Artificial Intelligence techniques

The AI is the science that studies the projection of the human intelligence in technological machines. In other words, the AI is the science that analyses the possibility to develop smart behavior patterns in technological machines. The AI is considered to exist since the time of ancient Greek civilizations. In fact, there are Greek myths about mechanical systems designed to emulate the human behavior. Later, during 19th and 20th centuries, the development of the first computers is considered as an attempt to emulate the working principle of the human brain in terms of calculations and memory. Nowadays, with the technological advances in the field of the informatics and the existence of very large amounts of data to be processed, the AI is on the focus of the research work.

The field of the AI is composed by numerous different techniques, which, in general, have been developed to emulate the human intelligence or decision making capability, as it is explained in the work of Wang et al. [21]. The most important AI techniques are the RL, ANNs, Fuzzy Logic, bio-inspired or metaheuristic optimization algorithms and Bayesian Networks. Each AI techniques serves to a determined goal and could be used individually or in interrelation with other AI techniques.

Bayesian Networks [22,23] are probability based networks that allow selection of the best action when a priori probabilities are known. Metaheuristic optimization algorithms [24,25] allow selection of the optimal action when the process is defined in a cost function. Bayesian Networks and optimization algorithms emulate the capability of the human brain to make decisions.

One of the most important features that offers the AI is the capability of the systems to learn automatically. This feature of self-learning is commonly known as ML, as it is explained in detail in the work of Fadlullah et al. [26]. The ML has undergone an important boom after the development of the ANNs, which are able to continuously learn from very large amounts of data. RL is another type of ML, in which the systems learns to make the best decisions in a given environment by using its own experience.

With the technological boom and the increasing processing capability of the processors a new learning method known as Deep Learning [26] has been born, in which new and amplified configurations of ANNs are used for the ML process. In the same way, the Deep Reinforcement Learning [26] method has also been created, which combines the use of the RL algorithm and ANNs to store the matrix *Q(s,a)* related to the RL algorithm.

The AI has numerous applications nowadays. Optimization algorithms are widely used in the business and public sectors due to their good results to reduce costs and increase gain margins [27]. Big companies like Amazon use predictions based on online searches and ML to carry out market studies and maximize their profits. The application SIRI of Apple brand mobile phones is a digital personal assistant that uses ML techniques to continuously self-learn. RL techniques are used in different fields such as the continuous learning of manufacturing robots, in Fanuc for instance, or to predict optimal trading strategies in the financial sector.

The self-tuning ML based yaw control strategy presented in this chapter makes use of three different AI techniques: RL, ANN and metaheuristic optimization algorithms. This section is structured as follows: the theoretical background of the RL is explained in Subsection III-16.3.1. Subsection III-16.3.2 analyses the theory behind the ANNs. And, finally, the theoretical basis of the optimization algorithms is introduced in the Subsection III-16.3.3.

III-16.3.1. Reinforcement Learning

RL [10,28-30] is an AI technique, corresponding to a type of ML, in which a determined system learns from the experience of its own interaction with the environment in which it is placed. As it is stated in the work of Sutton et al. [28], the training process of a RL algorithm is achieved by trial and error with the objective of maximizing a reward function defined numerically and by mapping of situations to actions.

A pipeline with the basic operating principle of a RL algorithm is presented in Figure III-16.3. A defined agent which is in a determined environment receives information of its state ($s \in S$) and decides to take the action ($a \in A$). As a result of this action, the agent receives information of its new state and the immediate reward of the action ($r \in R$). The objective of the RL algorithm is to find a map of states to actions, known as policy, to maximize the long-term reward in different situations. In other words, the RL controller selects the future actions with regard to the experiences of a whole range of actions in predefined states. The experiences are obtained by trial and error by interaction with a dynamic environment, as exposed in the work of Kaelbling et al. [31].

Figure III-16.3. Basic pipeline of a RL algorithm [12]

The main elements of a RL algorithm are:

- State ($s \in S$): Defines the state of an agent that is placed in a determined environment.
- Action ($a \in A$): Defines the action taken by an agent that is in a defined state ($s \in S$) in a determined environment.
- Reward ($r \in R$): Defines the immediate reward received by an agent that takes a certain action ($a \in A$) in a given state ($s \in S$).
- Policy (π): It is a mapping of the actions ($a \in A$) to the states ($s \in S$). Thus, it defines the behavior of the agent.
- Long-term reward (R_t): Indicates the long term reward received by the agent if a certain action ($a \in A$) in a given state ($s \in S$) is taken. The long-term reward is the value to be maximized.

The long-term reward R_t of a RL algorithm can be numerically calculated in different ways. The most widely-used expression is based on the addition of the immediate rewards ($r \in R$) received by the agent during a determined period of time and using a discount factor γ, as it is shown in Eq. (III-16.3).

$$R_t = \sum_{k=t}^{t+T} \gamma_k \cdot r_{t+k+1} \qquad (III\text{-}16.3)$$

where the discount factor γ is set to $0 < \gamma < 1$.

From now on, in order to refer to the function that indicates the long-term reward R_t expected by the agent a new expression is shown in Eq. (III-16.4).

$$E\left(\sum_{k=t}^{t+T} \gamma_k \cdot r_{t+k+1}\right) \qquad (III\text{-}16.4)$$

One important aspect related to the RL algorithms is that the environment in which the agent is placed is defined as a Markov Decision Process (MDP). This means that the environment transitions are independent on past states and exclusively depend on the current state ($s \in S$) and the action taken ($a \in A$). Therefore, the expressions of the state and reward transitions are presented in Eq (III-16.5) and Eq. (III-16.6), respectively.

$$p_{ss'}^{a} = p\,\{s_{t+1} = s' \,|\, s_t = s\,, a_t = a\,\} \qquad (III\text{-}16.5)$$

$$R_{ss'}^{a} = E\,\{r_{t+1} \,|\, s_t = s, a_t = a,\ s_{t+1} = s'\} \qquad (III\text{-}16.6)$$

The policy π followed by the agent defines the mapping of actions to states and, thus, dictates the criteria to take determined actions. Hence, the policy π defines the probability to select each action ($a \in A$) in each determined state ($s \in S$). As a result, the expected long-term reward with respect to the current state ($s \in S$) and the policy π followed, known as $V^{\pi}(s)$, and the expected long-term reward with respect to the current state ($s \in S$), the current action ($a \in A$) and the policy π followed, known as $Q^{\pi}(s,a)$, can be numerically calculated as shown in Eq. (III-16.7) and Eq. (III-16.8), respectively.

$$V^{\pi}(s) = E_{\pi}\,\{R_t \,|\, s_t = s\} = E_{\pi}\left\{\sum_{k=t}^{t+T} \gamma_k \cdot r_{t+k+1} \,|\, s_t = s\right\} \qquad (III\text{-}16.7)$$

$$Q^{\pi}(s,a) = E_{\pi}\,\{R_t \,|\, s_t = s, a_t = a\} = E_{\pi}\left\{\sum_{k=t}^{t+T} \gamma_k \cdot r_{t+k+1} \,|\, s_t = s,, a_t = a\right\} \qquad (III\text{-}16.8)$$

The optimal values of both $V^{\pi}(s)$ and $Q^{\pi}(s,a)$ can be expressed as in Eq. (III-16.9) and Eq. (III-16.10).

$$V(s) = \max(V^{\pi}(s)) \qquad (III\text{-}16.9)$$

$$Q(s,a) = \max(Q^{\pi}(s,a)) \qquad (III\text{-}16.10)$$

The objective of the RL algorithm is to find the optimal mapping of actions to states so that the value of the *Q(s,a)* expressed in Eq. (III-16.10) is maximized for each par of state ($s \in S$) and action ($a \in A$). To that end, there are 3 different methods to solve a MDP process: Dynamic Programming (DP), Monte Carlo (MC) method and Temporal Differences (TD). In the following lines an explanation on each one of them is introduced.

- <u>Dynamic Programming</u>

The DP method, explained in detail in the works of Bertsk et al. [32-34], is based on the knowledge of a model of the environment in which the agent is placed. That means that the state transitions $p_{ss'}^{a}$, see Eq. (III-16.5), and the reward transitions $R_{ss'}^{a}$, see Eq. (III-16.6), can be calculated analytically. As a result, the value of $V^{\pi}(s)$ and $Q^{\pi}(s,a)$ can also be represented analytically using Bellman equations, as shown in Eq. (III-16.11) and Eq. (III-16.12).

$$\begin{aligned} V^{\pi}(s) &= E_{\pi}\,\{r_{t+1} + \gamma \cdot V^{\pi}(s_{t+1}) \,|\, s_t = s\} \\ &= \sum_{a} \pi(s,a) \sum_{s_{t+1}} p_{ss'}^{a} \cdot [R_{ss'}^{a} + \gamma \cdot V^{\pi}(s_{t+1})] \end{aligned} \qquad (III\text{-}16.11)$$

$$Q^{\pi}(s,a) = E_{\pi}\{r_{t+1} + \gamma \cdot Q^{\pi}(s_{t+1}, a_{t+1})| s_t = s, a_t = a\}$$
$$= \sum_{a} \pi(s,a) \sum_{s_{t+1}} p^{a}_{ss'} \cdot [R^{a}_{ss'} + \gamma \cdot Q^{\pi}(s_{t+1}, a_{t+1})] \quad \text{(III-16.12)}$$

The numerically calculated values of $V^{\pi}(s)$ and $Q^{\pi}(s,a)$ are used to perform an iterative algorithm in which every action ($a \in A$) of every possible state ($s \in S$) is considered and the policies π that maximize the value of *Q(s,a)* are to be found.

One of the biggest drawbacks of this method is the computational cost, since for the calculation of each policy π calculations related to a great number of states and actions have to be performed. A pseudocode of the DP algorithm is presented in Algorithm III-16.1.

Dynamic Programming algorithm

```
% Initialize Q(s,a) randomly
Q(s,a) = Q(s,a)_ini
% Start the DP algorithm
while (Δ > β) do
% Initialize the minimum Q(s,a) improvement
  Δ = 0
% Define the actual state and select an action
  s ← (s ∈ S)
  a ← (a ∈ A)
% Calculate Q(s,a)
  Q(s,a) ← Eq. (III-16.12)
% Calculate the Q(s,a) improvement
  Δ = abs(Q_ant(s,a) - Q(s,a))
  Q_ant(s,a) = Q(s,a)
end
```

Algorithm III-16.1. Pseudocode of a DP based RL algorithm

- Monte Carlo method

The MC method [35,36] is based on the assumption that a model of the environment is unknown, and thus, its performance depends on the experimental data. Since the model is unknown, the values of the state transitions $p^{a}_{ss'}$, see Eq. (III-16.5), and the reward transitions $R^{a}_{ss'}$, see Eq. (III-16.6), and as a result, the values of $V^{\pi}(s)$ and $Q^{\pi}(s,a)$ cannot be analytically computed, so they are calculated as an average of the experimentally obtained reward values.

The objective is to try to calculate the value of $Q^{\pi}(s,a)$ for all the state-action pairs and find the policies π that maximize the value of *Q(s,a)*. To that end, usually stochastic policies that have probabilities greater than 0 to consider each state ($s \in S$) and action ($a \in A$) are implemented.

Different MC based algorithms can be implemented:

- Off-policy algorithms: The calculated policies are simultaneously used for the control strategy implemented in the system.
- On-policy algorithms: The policies calculated by the ongoing MC algorithm and the policies used by the control strategy implemented in the system are separated.

A pseudocode of the DP algorithm is presented in **¡Error! No se encuentra el origen de la referencia.**.

Monte Carlo algorithm

% Define a policy π and select an episode
% For each state s select an action a
s ← (s ∈ S)
a ← (a ∈ A)
% Calculate the long term reward Rt
Rt ← Eq. (III-16.3)
Rt_vec = [Rt_vec, Rt]
% Calculate Q(s,a) as a weighted average
Q(s,a) = average (Rt_vec)
% Evaluate the policy π

Algorithm III-16.2. Pseudocode of a MC method based RL algorithm

- Temporal Differences

The TD method is a combination of DP and MC methods having the advantages associated to each one of them. It is based on analytical calculation, like the DP method, but, like the MC method, it does not depend on a model of the environment. In this method, the calculations to continuously learn are performed between successive predictions instead of between predictions and the final value. Hence, the convergence is faster and the computational cost is remarkably reduced. The two principal TD based algorithms are Q-Learning, explained in detail in the works of Watkins et al. [7,37], and SARSA, introduced in the work of Adam et al. [38].

The principal difference between both methods is the calculation of the values of *Q(s,a)*. In the Q-Learning algorithm the state and actions are quantified and a matrix is obtained as a result of mapping a *Q(s,a)* value to each state-action par. However, in SARSA, the function *Q(s,a)* is considered as an exponential moving average continuous function.

The calculation of the *Q(s,a)* in SARSA algorithm can be expressed as shown in Eq. (III-16.13).

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \cdot [r + \gamma \cdot Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \tag{III-16.13}$$

The calculation of the *Q(s,a)* in Q-Learning algorithm can be expressed as shown in Eq. (III-16.14).

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \cdot [r + \gamma \cdot max_a Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \tag{III-16.14}$$

A simplified pseudocode of the SARSA and Q-Learning algorithms is presented in **¡Error! No se encuentra el origen de la referencia.**.

SARSA algorithm	Q-Learning algorithm
% Initialize Q(s,a) randomly	**% Initialize Q(s,a) randomly**
Q(s,a) = Q(s,a)_ini	Q(s,a) = Q(s,a)_ini
% Start the SARSA algorithm	**% Start the Q-Learning algorithm**
while (s ∈ Episode) do	while (s ∈ Episode) do
% Define the actual state and select an action	**% Define the actual state and select an action**
s ← (s ∈ S)	s ← (s ∈ S)
a ← (a ∈ A)	a ← (a ∈ A)
% Observe r and s_{t+1}	**% Observe r and s_{t+1}**
% Define the next action	**% Define the next action**
a_{t+1} ← (a ∈ A)	a_{t+1} ← (a ∈ A)
% Calculate Q	**% Calculate Q**
Q(s,a) ← Eq. (III-16.13)	Q(s,a) ← Eq. (III-16.14)
s = s_{t+1}	s = s_{t+1}
% Evaluate the policy π	**% Evaluate the policy π**
end	end
(a)	*(b)*

Algorithm III-16.3. (a) Pseudocode of a SARSA based RL algorithm (b) Pseudocode of a Q-Learning based RL algorithm

III-16.3.2. Artificial Neural Networks

ANNs correspond to a branch of the AI intended to mimic the performance of a biological brain. Biological brains are composed by millions of neurons distributed in layers and widely interconnected between them. Through these interactions between neurons the information flow from one neuron to another occurs. Furthermore, the information flow happens always in one direction, which can be either forwards or backwards. ANNs [39-41], which try to emulate this behavior, are digital systems with a variable number of neurons distributed in a structure similar to that of biological networks and with a similar functionality.

According to the work of Yang [42], the first standard artificial neuron design was introduced by W. McCulloch and W. Pitts in 1943 and, after that, they have undergone an important development. Nowadays they are very precious especially for their good performance in parallel processing, distributed memory alongside the number of neurons and the adaptability to the environment and the generalization capability.

ANNs are a compound of a variable number of neurons distributed in different ways and with a different type of interconnections. An individual neuron, shown in Figure III-16.4, is the smallest element of an ANN and presents the following structure:

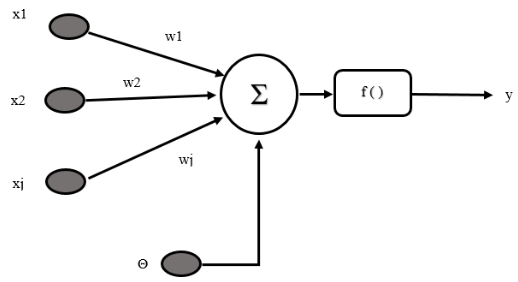


Figure III-16.4. Neuron of an ANN

The main elements of an artificial neuron are:

- Inputs (x_j): Define the inputs to the neuron.
- Input weights (w_j): Define the weights of each input to the neuron.
- Propagation rule (h_i): It defines the combination of the different inputs of the neuron before the activation function. The most common propagation rule is the linear combination of the product of each input and its weight. Moreover, usually another parameter commonly expressed as θ is added. Therefore, the propagation rule can be mathematically expressed as shown in Eq. (III-16.15).

$$h\,(x_1, \dots, x_j, w_1, \dots, w_j) = \sum_{j=1}^{n} w_j \cdot x_j - \theta \qquad \textit{(III-16.15)}$$

- Activation function (f_i): The activation function defines the activation state of the neuron. Additionally, it represents the output of the neuron.

  If it is an on/off neuron, the activation function of the neuron can be expressed as in Eq. (III-16.16). The output of this neuron is discrete, and corresponds to the original one introduced by W. McCulloch and W. Pitts in 1943.

$$y = \begin{cases} 1 & if \sum_{j=1}^{n} w_j \cdot x_j \geq \theta \\ 0 & if \sum_{j=1}^{n} w_j \cdot x_j < \theta \end{cases} \qquad \textit{(III-16.16)}$$

However, when a continuous output of the neuron is desired, usually a sigmoid function [43] is used as the activation function, as shown in Eq. (III-16.17).

$$f\ (x) = \frac{1}{1 + e^{-\beta \cdot x}} \qquad \textit{(III-16.17)}$$

where the value associated to the exponential factor is $\beta > 0$.

ANNs are formed by compound of a variable number of neurons in different structures and interconnection patterns. The neurons are divided in layers. As it is shown in Figure III-16.5, usually in a standard ANN there are 3 different neuron layers: The input layer (contains the input neurons, which are in number the same as the inputs of the ANN), the hidden layer (contains the processing neurons) and the output layer (contains the output neurons, which are in number the same as the outputs of the ANN). The number of hidden layers and the number of neurons in each hidden layer is adaptable and can be modified by the designer of the ANN.

There are different types of ANNs. The classification can be done according to different factors:

According to the number of layers:

- Sinple layer: ANNs with only one neuron layer.
- MultiLayer: ANNs with the neurons distributed in more than one layer.

According to the information flow:

- Feedforward: The information flow occurs exclusively in the direction from the input layer to the output layer. One example of this architecture is the MultiLayer Perceptron.
- Feedback or recurrent: There is information flow in both directions, i.e., from the input layer to the output layer and from the output layer to the input layer direction. One example of this architecture is the NonLinear Autoregressive Neural Network.

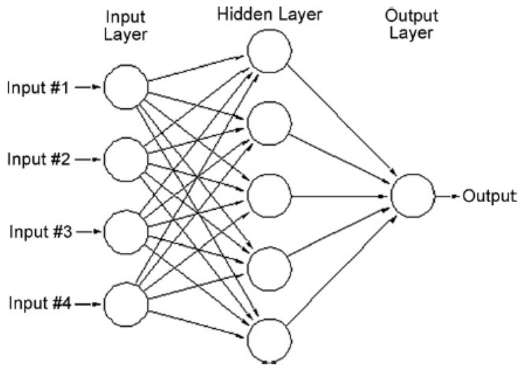


Figure III-16.5. Layer based structure of an ANN [44]

The training algorithms of the ANNs are the responsible for making the ANN learn from its input values. There are two main ANN training method groups: The supervised learning and the unsupervised learning. As it is exposed in the work of Chen et al. [39], the supervised learning adjusts the values of the weights related to the interconnection between neurons with the objective of minimizing the error existent between the output of the ANN and the reference output. That means that a reference output for the ANN should be known in the case of a supervised learning. The most important application of the supervised learning is for regressions or modelling of systems. One of the most used examples of supervised learning is the Back-Propagation algorithm.

The unsupervised learning do not need an output reference and the ANN is trained with numerous input patterns to explore the relation between them and categorize them. The most important application of the unsupervised learning is the clustering of data. A combination of supervised and unsupervised learning methods in an hybrid learning strategy is also possible.

III-16.3.3. Optimization Algorithms

Optimization algorithms are techniques designed and aimed to find the maximum/minimum or optimal solution of a determined function or problem. First optimization algorithms were introduced in the 20th century. Nowadays, optimization algorithms are applied to a grand variety of applications. As it is explained in the work of Yang et al. [42], one of the biggest application fields of the optimizations algorithms is the industrial engineering world, where the reduction of costs, the increment of the efficiency and the optimization of the industrial processes have become of capital importance.

Among the advantages of the optimization algorithms is the existence of processors with high capacity that allow the execution of the optimization algorithms at a considerable speed. The main drawback of the optimization algorithms is the necessity to have reliable models of the process that is to be optimized.

There are several types of optimization algorithms. In the work of Yang et al. [42], a classification of the optimization algorithms is proposed:

According to the existence of derivatives in the algorithm:

- Gradient based: They have derivatives in their optimization code. For instance, Gauss-Newton methods belong to this group.
- Gradient-free: They do not have derivatives in their optimization code. For example, the Nelder-Mead downhill simplex method belongs to this group.

According to the existence of randomness in the algorithm:

- Deterministic: There is no randomness in the optimization code. The result is always the same if the initial point is the same. An example of a deterministic optimization algorithm is the Hill climbing algorithm without random start.
- Stochastic: There is randomness in the optimization code. Hence, the result is not always the same even if the initial point is the same. An example of a stochastic optimization algorithm is the Genetic Algorithm (GA).

According to the mobility:

- Local: They typically converge to local optima because they do not have the ability to escape from them as a result of the lack of randomness in the optimization code. They are usually deterministic algorithms.
- Global: They always try to find the global optima. They have the ability to escape local optima as a result of the randomness in the optimization code. They are usually stochastic algorithms.

An important group inside the optimization algorithms is the bio-inspired or metaheuristics algorithms, which are inspired in natural processes to solve optimization problems. The metaheuristic algorithms [45-47] have been widely studied in the literature. There are many metaheuristic algorithms: Ant algorithm [48], bee algorithm [49], simulated annealing [50], GA, PSO, Differential Evolution (DE), etc.

The most widely used metaheuristic optimization algorithms are introduced in the following lines.

- Genetic Algorithms

The GAs [51,52] are metaheuristic optimization algorithms inspired in the evolution of biological individuals proposed by Charles Darwin. The GAs were first developed by Holland [53] during the decades of the 1960s and the 1970s.

As it is explained in detail in the work of Wang et al. [45], a GA is formed by a population of chromosomes or individuals, each one of them representing a solution to the optimization problem. Additionally, a fitness/cost function that evaluates each one of the individuals and provide them with a fitness value is necessary. The fitness function evaluates the specification fulfillment corresponding to each one of the individuals and, thus, the fitness value is an indicator of the chances of survival and reproduction of each one of the individuals. As a result, the individuals with the best fitness value tend to survive and the algorithm evolves towards the optimal solutions.

The execution of a GA could be summarized in the following 5 steps:

1) Initialization. Traditionally, the population is formed by a group of randomly generated individual solutions.

2) Evaluation. The fitness of each individual in the population is evaluated.

3) Selection. Individuals are selected based on the fitness value to breed a new generation.

4) Evolution. New individuals are created through crossover and mutation operations. The new population is composed of the individuals in the new generation and a few individuals from the previous generation.

5) Termination. Steps 2 to 4 are repeated until a termination condition has been reached.

- Particle Swarm Optimization

The PSO [54] algorithms are metaheuristic optimization algorithms inspired in the behavior of a group of particles, referred as swarm, in a search space and evolving towards an optimal solution. As it is explained in the work of Wang et al. [46], this algorithm is widely used due to its high robustness, small number of tunable parameters and its easy implementation.

As introduced in the work of Khan et al. [46], each particle is a possible solution to the optimization problem, and is associated with a position vector $x_{i,t}$ and a velocity vector $v_{i,t}$. Exactly as in the case of the GAs, in a PSO algorithm there must be a fitness function that evaluates the specification fulfillment of each particle and provides them with a fitness values.

The velocity and position update of each particle is calculated with the following expressions presented in Eq. (III-16.18) and Eq. (III-16.19), respectively.

$$v_{i,t+1} = H \cdot v_{i,t} + \varphi_1 \cdot (x\_opt_{i,t} - x_{i,t}) + \varphi_2 \cdot (x\_global\_opt_{i,t} - x_{i,t}) \quad \textit{(III-16.18)}$$

$$x_{i,t+1} = x_{i,t} + v_{i,t+1} \cdot \Delta \mathrm{t} \quad \textit{(III-16.19)}$$

where H [kg·m^2] is the inertia constant of the system, φ_1 [-] is the exploitation factor, φ_2 [-] is the exploration factor, $x\_opt_{i,t}$ [m] is the best solution of the particle and $x\_global\_opt_{i,t}$ [m] is the best solution of the whole swarm.

As it can be observed in Eq. (III-16.18), the velocity of each particle is computed with regard to the personal best fitness obtained by that particle and the global best fitness obtained by the whole swarm. By modifying factors φ_1 [-] and φ_2 [-] the exploration and exploitation capability of the algorithm can be configured. Furthermore, the inertia constant H [kg·m^2] defines the movement capacity of the particles.

The execution of a PSO could be summarized in the following 5 steps:

1) Initialization. The swarm population is randomly formed.
2) Evaluation. The fitness of each individual particle is evaluated.
3) Modification. The best position of each particle, the best position of the whole swarm and each particle's velocity are computed.
4) Update. Move each particle to the new position.
5) Termination. Steps 2 to 4 are repeated until a termination condition has been satisfied.

- Differential Evolution

The DE algorithm is metaheuristic optimization algorithm inspired in the evolutionary principles and intended to solve global optimization problems. The original DE algorithm was first introduced in the work of Storn and Price [55]. The DE algorithm starts with randomly initialized solution vectors, and like the GA algorithm is based on the principles of mutation, crossover and selection. Nevertheless, in contrast to GAs, the DE algorithms operate over each dimension of the solution separately.

The mutation is based on the recombination of three randomly chosen vectors, as shown in Eq. (III-16.20).

$$v_{i,t+1},= x_{a,t} + T \cdot (x_{b,t} - x_{c,t}) \quad \textit{(III-16.20)}$$

where $T \in [0,2]$ is commonly known as the differential weight and is a parameter adjustable by the designer.

After the mutation a crossover stage based on the crossover probability $C_r \in [0,1]$ is performed and the fitness of the individuals is evaluated. The selection stage is based on the fitness value of each individual, exactly as in the case of the GAs.

The execution of a DE algorithm could be summarized in the following 5 steps:

1) Initialization. Traditionally, the population is formed by a group of randomly generated individual solutions.
2) Mutation. The mutation shown in Eq. (III-16.20) is performed.
3) Crossover. The individual of the new generation are formed based on the crossover probability.
4) Selection. The individuals are selected based on their fitness value to breed a new generation

5) <u>Termination</u>. Steps 2 to 4 are repeated until a termination condition has been reached.

- <u>Multiobjective optimization. Pareto optimal Front</u>

A multiobjective optimization [56-58] problem is that in which more than one objective is to be optimized. In contrast to single-objective optimization problems, in multiobjective cases there is not only one unique optimal solution, but a set of optimal solutions that respond to the trade-off or compromise necessity between the objectives to be optimized.

The concept of optimization of multiobjective problems was generalized in the work of Vilfredo Pareto [59] in 1896. In these type of problems a solution is dominated if there is any other solution that has a better (higher or lower depending on the context of the optimization problem) fitness value for all the objectives to be optimized. If there is no such a solution, i.e., if the improvement of one objective causes the degrading of any other objective, then the solution is known as non-dominated. The set of non-dominated solutions is known as the Pareto optimal Front (PoF). A PoF of a double-objective optimization problem is illustrated in Figure III-16.6.

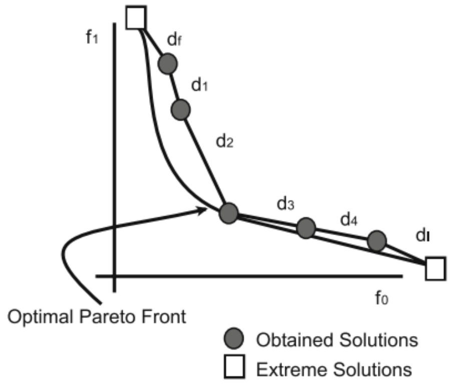


Figure III-16.6. PoF of a double-objective optimization problem [57]

As it can be observed in Figure III-16.6, all the points forming the continuous line marked as the PoF respond to the same principle of compromise necessity between the objectives. If the value of one of the objectives is improved, the other one degrades. An algorithm to find the PoF of a multiobjective problem could be implemented in any of the previously presented metaheuristic optimization algorithms.

III-16.4. Machine Learning based wind turbine yaw control

An adequate alignment of the wind turbine rotor with respect to the incoming wind by means of the yaw system of the wind turbine enables increment of the power generation. Nevertheless, as it was exposed in Section III-16.2 of this chapter, the enhancement of the generated power is achieved at cost of an increase of the mechanical loads in different elements of the wind turbine, especially the yaw bearings. Hence, an adequate design and tuning of the yaw control system is of great importance to both optimize the power generation of the wind turbine and ensure its safe operation. The absence of an adequate control strategy could result in an excessively aggressive yaw activity, which could endanger the safety of the mechanical components of the wind turbine and reduce their lifetime.

Usually, classical control structures based on PIDs have been used for the design of the yaw control strategy of the wind turbine [20,60]. However, these classical control structures show some drawbacks in form of “wind up” of the integral action and posterior big oscillations, which can result in an undesired increment of the mechanical loads. As a result, some advanced control strategies for the yaw angle control of a wind turbine are proposed in the literature. In this context, Song et al. [61] present a control strategy based on a Model Predictive Control and Bharani et al. [62] introduce a Fuzzy Logic based control strategy for this purpose. In the work of Saenz-Aguirre et al. [12], an ANN based RL control strategy is proposed for the yaw control of a wind turbine.

In this chapter, with the objective of achieving an improved performance of the yaw control system of a wind turbine, a ML based wind turbine yaw control system is exposed. A block diagram of the proposed ML based yaw control strategy is presented in Figure III-16.7.

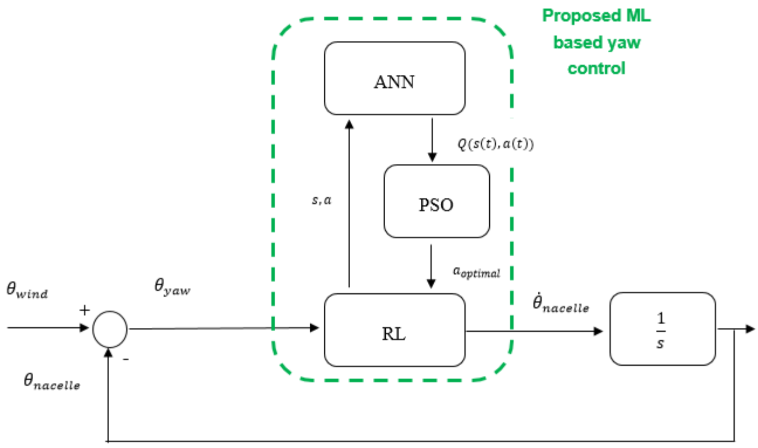


Figure III-16.7. Pipeline of the proposed ML based yaw control

The proposed yaw control system is based on the following AI techniques:

- A RL algorithm that learns from its own experience and enables the wind turbine to select the optimal decision in each scenario of its operation.
- An ANN to store the data of the matrix *Q(s,a)* of the RL algorithm.
- A PSO and PoF based optimization algorithm to select the set of optimal actions that respond to the compromise necessity between the power increment and the mechanical loads associated to the yaw rotation.

This section is structured as follows: the design procedure of the RL algorithm applied to the ML based yaw control is explained in Subsection III-16.4.1. Subsection III-16.4.2 presents the structure and design process of the MLP-BP neural network. The design of the PSO and PoF based algorithm is explained in Subsection III-16.4.3. Finally, the Decision Making (DM) algorithm is exposed in Subsection III-16.4.4.

III-16.4.1. Yaw Control RL

The RL algorithm developed for the yaw control of a HAWT presents multiple state, action and immediate reward variables. The objective of the multivariable structure is an improved characterization of the system in the most accurate way possible. To that end, 2 state variables, 2 action variables and 2 immediate reward variables are considered in the proposed RL algorithm.

The states *s* are:

- StateYawA [deg]: This state defines the orientation difference between the wind incident to the rotor and the nacelle of the wind turbine. As it can be observed in Eq. (III-16.2), the power output of the wind turbine is affected by this misalignment angle. The expression to calculate the value of this state is shown in Eq. (III-16.21).

$$\theta_{yaw} = \theta_{wind} - \theta_{nacelle} \tag{III-16.21}$$

- StateWindS [m/s]: This state defines the wind speed value incident to the rotor. As it has been shwon stated in Figure III-16.2 (b), the power output of the wind turbine is not directly proportional to the wind speed value, but it depends on the operation zone of the wind turbine, which depends on the wind speed value. As a result, it is important to know the wind speed value because it will help characterize the possible power gain achievable with the yaw rotation of the wind turbine.

The actions *a* are:

- ActionYawK [deg/s]: This action defines the proportional gain associated to the yaw rotational speed controller. The expression to calculate the yaw rotational speed is shown in Eq. (III-16.22).

$$\Omega_{yaw} = ActionYawK \cdot \theta_{yaw} \tag{III-16.22}$$

As it can be seen in Eq. Eq. (III-16.22), the higher the value of the action ActionYawK [deg/s] is, the higher the yaw rotational speed will be.

- ActionYaw [deg]: This action defines the limit associated to the yaw rotation. In some cases, due to mechanical actuator problem or safety issues, the yaw rotation of the nacelle is limited to a certain value. The expression to note the rotation range allowed by this action is shown in Eq. (III-16.23).

$$\Delta\theta_{yaw} \in [-\text{ActionYaw}, \text{ActionYaw}] \qquad \textit{(III-16.23)}$$

The immediate rewards *r* are:

- RewardP [%]: This immediate reward defines the power gain achieved by the wind turbine when a certain yaw action is performed. The expression to compute this immediate reward is shown in Eq. (III-16.24).

$$RewardP = \frac{P\_control - P\_no\_control}{P\_no\_deviation} \cdot 100 \qquad \textit{(III-16.24)}$$

  As it can be observed in Eq. (III-16.24), in order to calculate the power gain 3 different scenarios related to the yaw actuation of the wind turbine are considered. The scenario P_control refers to the scenario in which the yaw control of the wind turbine is active and the nacelle of the wind turbine rotates to the yaw command provided by the yaw control and at the provided yaw speed value. The scenario, P_no_control refers to the scenario in which the yaw control of the wind turbine is not active, and, thus, the orientation of the wind turbine nacelle is fixed. Finally, the scenario P_no_deviation refers to the scenario in which the nacelle of the wind turbine is perfectly aligned with the direction of the wind incoming to the rotor. Hence, this value refers to the maximum power that the wind turbine can generate with a defined value of the wind speed.

- RewardM [N·m]: This immediate reward defines the value of the mechanical moment in the yaw bearings. The value of this immediate reward has been defined with the mechanical moment in the yaw bearings because it has been found as the most critical mechanical load when performing a yaw rotation. Different mechanical load values, or even a weighted average of them, could be considered as the immediate reward to be considered by the proposed ML based yaw control algorithm.

As it was stated in Subsection III-16.3.1 of this chapter, in a RL algorithm the calculation of the values *Q(s,a)* for each state-action par is associated to the long-term reward considering a discount factor γ, see Eq. (III-16.3). In the RL algorithm proposed in this chapter there are 2 different immediate rewards *r*. Therefore, 2 different matrices *Q(s,a)* will result in the algorithm. The expression to calculate the matrices *Q(s,a)* using the immediate rewards *r* is shown in Eq. (III-16.25).

$$Q(s,a) = \sum_{i=0}^{i=T} r_{t+i} \cdot \gamma^i \qquad \textit{(III-16.25)}$$

The expression in Eq. (III-16.25) is applied to both the immediate rewards *r* considered in the ML based yaw control algorithm presented in this paper and the expression of both matrices *Q(s,a)* are obtained and presented in Eq. (III-16.26) and Eq. (III-16.27). The discount factor γ is set to 1 in both cases because it is considered that all the values in the time horizon are equally important.

$$Q\_P(s,a) = \frac{\frac{1}{T}\int_t^{t+T} (P\_control - P\_no\_control) \cdot dt}{\frac{1}{T}\int_t^{t+T} P\_no\_deviation \cdot dt} \cdot 100 \quad [\%] \qquad \textit{(III-16.26)}$$

$$Q\_M(s,a) = \int_t^{t+T} RewardM\,(t) \cdot dt \qquad [N \cdot m] \qquad \textit{(III-16.27)}$$

After definition of the states *s*, actions *a*, immediate rewards *r* and the expressions of the matrices *Q_P(s,a)* and *Q_M(s,a),* simulations of the performance of the wind turbine to obtain data for the training process of the RL algorithm are carried out. The simulations are carried out with the aeroelastic code FAST [63] and the wind turbine model NREL 5MW, presented in the work of Jonkman et al. [14], both designed by the National Renewable Energies Laboratory (NREL) in the USA.

The objective of the training process of the RL algorithm is to obtain the data related to all possible actuation scenarios associated to the yaw control of the wind turbine. Thus, in the design process presented in this chapter, an offline training process of the wind turbine with all the possible considered wind speed values and the yaw control actions is proposed. Thus, simulations with StateWindS=3:1:17 [m/s], StateYawA=-90:10:90 [deg], ActionYawK=0.1:0.1:1 [-] and ActionYaw=-90:10:90 [deg] have been carried out with the aeroelastic code FAST. The values of the matrices *Q_P(s,a)* and *Q_M(s,a)* are calculated with the data obtained from the simulations, see Eq. (III-16.26) and Eq. (III-16.27).

The fact that the RL training is performed offline indicates that it is independent from the actual operating conditions of the wind turbine. Nevertheless, an online training process of the RL algorithm during operation of the wind turbine and linked to the actual operating conditions could be possible to keep the system learning from real field data and its own experience.

III-16.4.2. Yaw Control MLP-BP

A MLP-BP neural network is designed to store the data of the matrices *Q_P(s,a)* and *Q_M(s,a)* corresponding to the RL algorithm. The objective of storing these matrices as continuous functions *Q_P(s(t),a(t))* and *Q_M(s(t),a(t))* is to eliminate the necessity of large amount of data management, which could result problematic in the implementation of the control strategy in the control system of the wind turbine, due to memory issues. Additionally, with the use of an ANN to store the data of the RL algorithm, the replacement policy of the RL algorithm is incorporated, since the ANN learns from the new calculated values. This aspect is of great importance if an online training of the RL algorithm during operation of the wind turbine is implemented. In that case, the ANN continuously learns from new calculated values and the accuracy of the functions *Q_P(s(t),a(t))* and *Q_M(s(t),a(t))* increase.

The selected topology of the ANN designed to store the data of the matrices *Q_P(s,a)* and *Q_M(s,a)* is a MLP-BP. A MLP-BP is a neural network based on neurons of the type perceptron and with a variable number of hidden layers. The characteristic of the MLP-BP is that the information flow occurs exclusively in the direction from input neurons to output neurons and not in reverse. The BP training process is a supervised training strategy in which the theoretical output of the ANN and the real output of the ANN are compared and the weights of the neurons are adjusted to minimize this error.

The designed MLP-BP neural network presents 4 inputs and 2 outputs. A pipeline with the input and outputs of the designed MLP-BP neural network is presented in Figure III-16.8. Internally, the MLP-BP presents a structure with one input layer with 4 neurons, two hidden layers with 75 neurons and 25 neurons respectively and one output layer with 2 neurons.

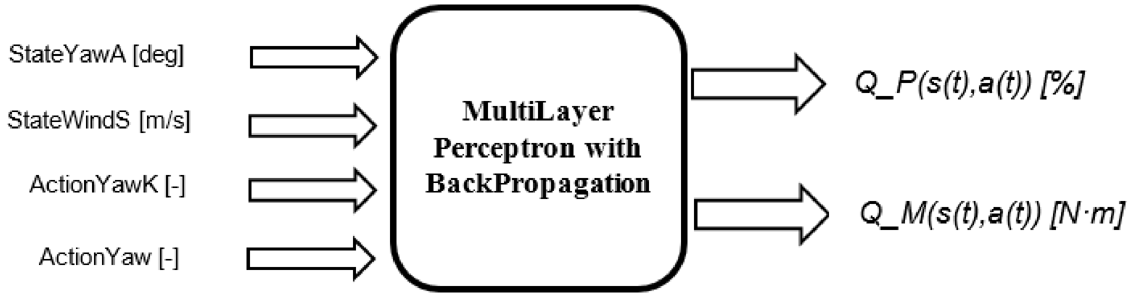


Figure III-16.8. Input and outputs of the MLP-BP designed for the ML based yaw control strategy

The learning rate for the training process of the MLP-BP has been set to $1\cdot 10^{-50}$. The training ratio, validation ratio and test ration have been set to 90 %, 5 % and 5 %, respectively. After the training process, correlation coefficients of 0.9999 and Mean Squared Error (MSE) of $1.62\cdot 10^{-6}$ are obtained. The high value of the correlation coefficient and the low value of the MSE are indicators of a correct training process and that the MLP-BP is good enough to be used in the ML based yaw control strategy proposed in this chapter.

III-16.4.3. Yaw Control PSO and PoF

As it was stated in Section III-16.2 of this chapter, the yaw actuation of a wind turbine allows alignment of the rotor of the wind turbine with the direction of the incoming wind and, thus, the power generated by the wind turbine can be maximized in some scenarios. Nevertheless, this power gain is achieved at cost of high mechanical loads in several components of the wind turbine, especially the yaw bearings, which could endanger the safe operation of the wind turbine or reduce its lifetime.

After carrying out simulations with the aeroelastic code FAST for the training process of the RL algorithm, the following tendency of the states *s* and actions *a* corresponding to the RL algorithm has been observed:

- An increased value of the state StateYawA [deg] causes the value of the immediate reward RewardP [%] to have greater values. As a result of an increased value of the state StateYawA [deg] the yaw actuation is usually more important and the immediate reward RewardM [N·m] is increased.
- An increased value of the state StateWindS [m/s] makes the value of the immediate reward RewardP [%] to be smaller. This fact depends on the wind speed value that determines the operating zone of the wind turbine. In some cases, the StateWindS [m/s] is so high that despite the StateYawA [deg] the system keeps operating in the rated power zone and no RewardP [%] can be achieved. The immediate reward RewardM [N·m] get bigger with greater StateWindS [m/s] values.
- An increased value of the ActionYawK [-] makes the immediate reward RewardP [%] to be higher, since the yaw rotation is performed at a greater rotational speed. The immediate reward RewardM [N·m] gets bigger as well.
- An increased value of the ActionYaw [deg] makes the immediate reward RewardP [%] to be higher, since a longer rotation of the wind turbine rotor is allowed. The immediate reward RewardM [N·m] gets bigger as well.

The objective of the PSO and PoF based optimization algorithm designed in this paper is to obtain a set of optimal yaw actions, ActionYawK [-] and ActionYaw [deg], that respond to the compromise necessity between RewardP [%] and RewardM [N·m].

A pseudocode of the PSO and PoF based optimization algorithm designed for the ML based yaw control strategy presented in this chapter is shown in Algorithm III-16.4.

PSO and PoF optimization algorithm

```
% Initialization
φ1_max = φ1_max
φ2_max = φ2_max
H = H
P =P                    % Population size
n=n                     % Number of iterations
a=a_ini(2,P)
% Definition of the states (s ∈ S)
s(1) ← StateYawA
s(2) ← StateWindS
% Start the PSO algorithm
while (iter<n) do
 for 1:1:P
% Evaluate the current particle
  r=MLP-BP(s,a)
% Evaluate its introduction to the PoF
  if r(1)<r1_global && r(2)<r2_global
   r_PoF= MLP-BP (s,a_ant)
   PoF=[PoF,r_PoF]
   a_PoF=[a_PoF,a_ant]
   end
% Generate the new swarm
  φ1 = random(φ1_max)
  φ2 = random(φ2_max)
  v ← Eq. (III-16.18)
  x ← Eq. (III-16.19)
 end
end
```

Algorithm III-16.4. Pseudocode of the PSO and PoF based optimization algorithm

As it can be observed in **¡Error! No se encuentra el origen de la referencia.**, the output of the PSO and PoF optimization algorithm is a set of optimal solutions, known as PoF, that respond to the compromise necessity between the power gain and the mechanical loads due to the yaw rotation. To calculate this PoF the optimization algorithm makes use of the functions *Q_P(s(t),a(t))* and *Q_M(s(t),a(t))* as the fitness functions. The states of the system, StateYawA [deg] and StateWindS [m/s], are defined and the fitness value of different set of actions, ActionYawK [-] and ActionYaw [deg], is evaluated. The final optimal solutions are the solutions in which one of the fitness values cannot be increased without degrading the other one.

The implemented PSO and PoF optimization algorithm show correct results in a variety of state scenarios, StateYawA [deg] and StateWindS [m/s], of the wind turbine:

- When the wind turbine operates in the partial power zone a more aggressive yaw actuation is translated in a higher power gain but at cost of incremented mechanical loads.
- When the wind turbine operates in the rated power zone and the value of the yaw misalignment is high enough to move the operation of the wind turbine out of the rated power zone, a more aggressive yaw actuation is translated in a higher power gain but at cost of incremented mechanical loads.
- When the wind turbine operates in the rated power zone and the value of the yaw misalignment is not high enough to move the operation of the system out of the rated power zone, a more aggressive yaw actuation is translated in zero power gain and incremented mechanical loads, which makes the yaw actuation useless.

III-16.4.4. Yaw Control DM

The DM algorithm selects one of the optimal actions proposed as the result of the PSO-PoF optimization algorithm. The DM algorithm proposed in this chapter considers the mechanical loads as the limiting factor when selecting the yaw actuation and it could be summarized as follows:

- The solutions that suppose a value of the function *Q_M(s(t),a(t))* higher than a predefined threshold are not taken into consideration due to safety issues.
- From the set of solutions that are taken into consideration, the one with the highest value of the function *Q_P(s(t),a(t))* is selected.

Other different approaches for the selection of the yaw optimal actuation based on more complex principles could also be evaluated and implemented.

III-16.5. Conclusions

The design procedure of a ML based yaw control algorithm for a HAWT based on AI techniques has been presented in this chapter. The proposed yaw control strategy is aimed to improve the performance of classical yaw control strategies by means of the use of AI techniques, which emulate the performance of natural processes to provide digital systems with intelligence and self-learning capability. The self-learning capability is the main characteristic of the ML.

The proposed ML based yaw control strategy makes use of three different AI techniques for the development of the control strategy. The RL algorithm maps actions to states and thus allows the development of a policy in the wind turbine that selects the best actions in different wind turbine operation scenarios. The ANN provides a very important learning capability and allows a continuous learning process in the wind turbine, as well as, a simplified data management by storage of large amounts of data as continuous functions. Finally, the PSO and PoF based optimization algorithm allows to select the actions that maximize the power output of the wind turbine and minimize the mechanical loads generated as a result of the yaw rotation.

The most important capability of the proposed ML based yaw control strategy is the self-tuning. As a result of the self-learning capability of the ML system, there is no need for tuning a closed loop for the yaw angle control of the wind turbine. Therefore, the risk associated to a possible inadequate tuning of this control loop is erased. In fact, an inadequate control tuning could cause considerable power generation losses or high mechanical loads that could endanger the safe operation of the wind turbine.

Simulations of the proposed ML based yaw control strategy with the aeroelastic code FAST show promising results in comparison to other more simple controllers based on the classical control theory. The most visible improvements are increased generated power values and considerable mechanical load reductions in the yaw bearings of the wind turbine for different wind scenarios.

Funding: This research was partially funded by Fundation VITAL Fundazioa.

Acknowledgments: The authors are grateful to the Government of the Basque Country and the University of the Basque Country UPV/EHU through the SAIOTEK (S-PE11UN112) and EHU12/26 research programs, respectively.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhao X., Yan Z., Xue Y., Zhang X.: Wind Power Smoothing by Controlling the Inertial Energy of Turbines With Optimized Energy Yield, IEEE Access, 2017, 5, pp. 23374-23382.
2. Rosales-Asensio E., Borge-Diez D., Blanes-Peiro J., Perez-Hoyos A., Comenar-Santos A.: Review of wind energy technology and associated market and economic conditions in Spain, Renewable & Sustainable Energy Reviews, 2019, 101, pp. 415-427.
3. WindEurope.: Wind energy in Europe in 2018. Trends and Statistics, 2019.
4. Nyanteh Y., Schneider N., Netter D., Wei B., Masson P.J.: Optimization of a 10 MW Direct Drive HTS Generator for Minimum Levelized Cost of Energy, IEEE Trans.Appl.Supercond., 2015, 25, (3), pp. 1-4.
5. Kim Y.-.: Robust data driven H-infinity control for wind turbine, Journal of the Franklin Institute, 2016, 353, (13), pp. 3104-3117.
6. Merabet A., Ahmed K.T., Ibrahim H., Beguenane R.: Implementation of Sliding Mode Control System for Generator and Grid Sides Control of Wind Energy Conversion System, IEEE Transactions on Sustainable Energy, 2016, 7, (3), pp. 1327-1335.
7. Watkins C.J.C.H., Dayan P.: Q-learning, Mach.Learning, 1992, 8, (3), pp. 279-292.
8. Erdogmus D., Fontenla-Romero O., Principe J.C., Alonso-Betanzos A., Castillo E.: Linear-least-squares initialization of multilayer perceptrons through backpropagation of the desired response, IEEE Trans.Neural Networks, 2005, 16, (2), pp. 325-337.
9. Ho S.L., Lo E.W.C., Wong H.C.: A particle swarm optimization-based method for multiobjective design optimizations, IEEE Trans.Magn., 2005, 41, (5), pp. 1756-1759.
10. Zhang D., Han X., Deng C.: Review on the research and practice of deep learning and reinforcement learning in smart grids, CSEE Journal of Power and Energy Systems, 2018, 4, (3), pp. 362-370.
11. Yang Z., Merrick K., Jin L., Abbass H.A.: Hierarchical Deep Reinforcement Learning for Continuous Action Control, IEEE Transactions on Neural Networks and Learning Systems, 2018, 29, (11), pp. 5174-5184.
12. Saenz-Aguirre A., Zulueta E., Fernandez-Gamiz U., Lozano J., Lopez-Guede J.M.: Artificial Neural Network Based Reinforcement Learning for Wind Turbine Yaw Control, Energies, Jan 2019.
13. Bivona S., Bonanno G., Burlon R., Gurrera D., Leone C.: Stochastic models for wind speed forecasting, Stochastic models for wind speed forecasting, 2010, 52, (2), pp. 1157-1165.
14. Jonkman J.M., Butterfield S., Musial W., Scott G.: Definition of a 5MW Reference Wind Turbine for Offshore System Development, National Renewable Energy Laboratory (NREL), 2009.
15. J.H. L., L.Y. P., A. W.: Control of Wind Turbines: Past, Present, and Future, American Control Conference 2009, June 2009 St. Louis (USA).
16. Harris M., Hand M., Wright A.: LIDAR for Turbine Control, NREL Technical Report NREL/TP-500-39154., 2005.
17. M. Y., S. Ö.: Design and Efficiency of 5 kW Wind Turbine Without Gearbox, Controlled by Yaw and Pitch Drivers, Çanakkale Onsekiz Mart Üniversitesi Fen Bilimleri Enstitüsü Dergisi, 4, (1), pp. 74-87.
18. Ahrens M., Kucera L., Larsonneur R.: Performance of a magnetically suspended flywheel energy storage device, IEEE Transactions on Control System Technology, 1996, 4, (5), pp. 495-502.

19. Zheng S., Yang J., Song X., Ma C.: Tracking Compensation Control for Nutation Mode of High-Speed Rotors With Strong Gyroscopic Effects, IEEE Trans.Ind.Electron., 2018, 65, (5), pp. 4156-4165.
20. Shariatpanah H., Fadaeinedjad R., Rashidinejad M.: A New Model for PMSG-Based Wind Turbine With Yaw Control, IEEE Trans.Energy Convers., 2013, 28, (4), pp. 929-937.
21. Wang X., Li X., Leung V.C.M.: Artificial Intelligence-Based Techniques for Emerging Heterogeneous Network: State of the Arts, Opportunities, and Challenges, IEEE Access, 2015, 3, pp. 1379-1391.
22. Castro P.A.D., Zuben F.J.V.: Learning Ensembles of Neural Networks by Means of a Bayesian Artificial Immune System, IEEE Trans.Neural Networks, 2011, 22, (2), pp. 304-316.
23. Khanafer R.M., Solana B., Triola J., et al.: Automated Diagnosis for UMTS Networks Using Bayesian Network Approach, IEEE Transactions on Vehicular Technology, 2008, 57, (4), pp. 2451-2461.
24. Ma H., Simon D., Siarry P., Yang Z., Fei M.: Biogeography-Based Optimization: A 10-Year Review, IEEE Transactions on Emerging Topics in Computational Intelligence, 2017, 1, (5), pp. 391-407.
25. León-Aldaco S.E.D., Calleja H., Alquicira J.A.: Metaheuristic Optimization Methods Applied to Power Converters: A Review, IEEE Transactions on Power Electronics, 2015, 30, (12), pp. 6791-6803.
26. Fadlullah Z.M., Tang F., Mao B., et al.: State-of-the-Art Deep Learning: Evolving Machine Intelligence Toward Tomorrow's Intelligent Network Traffic Control Systems, IEEE Communications Surveys Tutorials, 2017, 19, (4), pp. 2432-2455.
27. Dostál P.: The Use of Optimization Methods in Business and Public Services, Zelinka I., Snášel V., Abraham A. (eds) Handbook of Optimization. Intelligent Systems Reference Library, 2013, 38, (Springer, Berlin, Heidelberg).
28. Jagodnik K.M., Thomas P.S., Bogert A.J.v.d., Branicky M.S., Kirsch R.F.: Training an Actor-Critic Reinforcement Learning Controller for Arm Movement Using Human-Generated Rewards, IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2017, 25, (10), pp. 1892-1905.
29. Mongillo G., Shteingart H., Loewenstein Y.: The Misbehavior of Reinforcement Learning, Proc IEEE, 2014, 102, (4), pp. 528-541.
30. Sutton R.S., Barto A.G.: Reinforcement Learning: An Introduction, MIT Press, 1998, (Cambridge, MA, USA).
31. Kaelbling L.P., Littman M.L., Moore A.W.: Reinforcement learning: A survey, J. Artif. Intell. Res., 1996, 4, (1), pp. 237-285.
32. Bertsekas D.P.: Abstract Dynamic Programming, Belmont, MA, USA:Athena Scientific, 2013.
33. Bertsekas D.P.: Dynamic Programming and Optimal Control: Approximate Dynamic Programming, Belmont, MA, USA:Athena Scientific, 2012, 2.
34. Bertsekas D.P.: Value and Policy Iterations in Optimal Control and Adaptive Dynamic Programming, IEEE Transactions on Neural Networks and Learning Systems, 2017, 28, (3), pp. 500-509.
35. Kao K., Wu I., Yen S., Shan Y.: Incentive Learning in Monte Carlo Tree Search, IEEE Transactions on Computational Intelligence and AI in Games, 2013, 5, (4), pp. 346-352.
36. Coulom R.: Efficient selectivity and backup operators in Monte-Carlo tree search, Proc. 5th Int. Conf. Comput. Games, 2006, pp. 72-83.
37. Watkins C.J.C.H.: Learning from Delayed Rewards, PhD thesis, King's College, Cambridge, UK, May 1989.
38. Adam S., Busoniu L., Babuska R.: Experience Replay for Real-Time Reinforcement Learning Control, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2012, 42, (2), pp. 201-212.
39. Chen S.H., Jakeman A.J., Norton J.P.: Artificial Intelligence techniques: An introduction to their use for modelling environmental systems, Mathematics and Computers in Simulation, 2008, 78, pp. 379-400.
40. Yao X.: Evolving artificial neural networks, Proc. IEEE, 1999, 87, (9), pp. 1423-1447.
41. Oong T.H., Isa N.A.M.: Adaptive Evolutionary Artificial Neural Networks for Pattern Classification, IEEE Trans.Neural Networks, 2011, 22, (11), pp. 1823-1836.

42. Yang X.S.: Optimization and Metaheuristic Algorithms in Engineering, in: Metaheursitics in Water, Geotechnical and Transport Engineering (Eds. X. S. Yang, A. H. Gandomi, S. Talatahari, A. H. Alavi), Elsevier, 2013, pp. 1-23.
43. Jain A.K., Mao J., Mohiuddin K.M.: Artificial neural networks: a tutorial, Computer, 1996, 29, (3), pp. 31-44.
44. Rounds S.A.: Development of a neural network model for dissolved oxygen in the Tualatin River, Oregon, Proceedings of the Second Federal Interagency Hydrologic Modeling Conference, Las Vegas, NV, 2002.
45. Wang L., Shen J.: A Systematic Review of Bio-Inspired Service Concretization, IEEE Transactions on Services Computing, 2017, 10, (4), pp. 493-505.
46. Khan B., Singh P.: Selecting a Meta-Heuristic Technique for Smart Micro-Grid Optimization Problem: A Comprehensive Analysis, IEEE Access, 2017, 5, pp. 13951-13977.
47. Bala A., Ismail I., Ibrahim R., Sait S.M.: Applications of Metaheuristics in Reservoir Computing Techniques: A Review, IEEE Access, 2018, 6, pp. 58012-58029.
48. Liao T., Socha K., Oca M.A.M.d., Stützle T., Dorigo M.: Ant Colony Optimization for Mixed-Variable Optimization Problems, IEEE Transactions on Evolutionary Computation, 2014, 18, (4), pp. 503-518.
49. Xiang Y., Zhou Y., Tang L., Chen Z.: A Decomposition-Based Many-Objective Artificial Bee Colony Algorithm, IEEE Transactions on Cybernetics, 2019, 49, (1), pp. 287-300.
50. Bandyopadhyay S., Saha S., Maulik U., Deb K.: A Simulated Annealing-Based Multiobjective Optimization Algorithm: AMOSA, IEEE Transactions on Evolutionary Computation, 2008, 12, (3), pp. 269-283.
51. Srinivas M., Patnaik L.M.: Genetic algorithms: A survey, Computer, 1994, 27, (6), pp. 17-26.
52. Anderson-Cook C.M.: Practical Genetic Algorithms, Oxfordshire, U.K.: Taylor & Francis, 2005.
53. Holland J.H.: Adaptation in Natural and Artificial Systems, Ann Arbor, MI, USA: Univ. Michigan Press, 1975.
54. Kennedy J., Eberhar R.C.: Particle swarm optimization, Proc. IEEE Int. Conf. Neural Netw., Perth, WA, Australia, Jul 1995, pp. 1942-1948.
55. Storn R., Price K.: Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim, 1997, 11, (4), pp. 341-359.
56. Ehrgott M., Gandibleux X.: A survey and annotated bibliography of multiobjective combinatorial optimization, OR-Spektrum, 2000, 22, (4), pp. 425-460.
57. Durillo J.J., Nebro A.J., Garc\'ia-Nieto J., Alba E.: On the Velocity Update in Multi-Objective Particle Swarm Optimizers, in Coello Coello C.A., Dhaenens C., Jourdan L. (Eds.): Advances in Multi-Objective Nature Inspired Computing (Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 45-62.
58. Coello Coello C.A., Dhaenens C., Jourdan L.: Multi-Objective Combinatorial Optimization: Problematic and Context, in Coello Coello C.A., Dhaenens C., Jourdan L. (Eds.): Advances in Multi-Objective Nature Inspired Computing (Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 1-21.
59. Pareto V.: Cours D'Economie Politique, F. Rouge, Lausanne, 1896, I, (II).
60. Karakasis N., Mesemanolis A., Nalmpantis T., Mademlis C.: Active yaw control in a horizontal axis wind system without requiring wind direction measurement, IET Renewable Power Generation, 2016, 10, (9), pp. 1441-1449.
61. Song D., Yang J., Fan X., et al.: Maximum power extraction for wind turbines through a novel yaw control solution using predicted wind directions, Energy Conversion and Management, 2018, 157, pp. 587-599.
62. Bharani R., Jayasankar K.C.: Yaw Control of Wind Turbine Using Fuzzy Logic Controller, Power Electronics and Renewable Energy Systems, 2015, 326, pp. 997-1006.
63. NREL NWTC FAST Version 7. Available online: https://nwtc.nrel.gov/FAST7/ (accessed on 21 Oct 2018).