Self-tuning Yaw Control Strategy of a 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 a 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

A. Saenz-Aguirre (✉)
Nuclear Engineering and Fluid Mechanics Department, University of the Basque Country, Eibar, Spain
e-mail: aitor.saenz@ehu.eus

E. Zulueta · J. M. Lopez-Guede
Automatic Control and System Engineering Department, University of the Basque Country, Vitoria-Gasteiz, Araba, Spain
e-mail: ekaitz.zulueta@ehu.eus

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

U. Fernandez-Gamiz
Nuclear Engineering and Fluid Mechanics Department, University of the Basque Country, Vitoria-Gasteiz, Araba, Spain
e-mail: unai.fernandez@ehu.eus

J. A. Ramos-Hernanz
Electrical Engineering Department, University of the Basque Country, Vitoria-Gasteiz, Araba, Spain
e-mail: josean.ramos@ehu.eus

N. Mahdavi Tabatabaei and N. Bizon (eds.), *Numerical Methods for Energy Applications*, Power Systems,
https://doi.org/10.1007/978-3-030-62191-9_32

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

Abbreviations 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 Back Propagation
MDP	Markov Decision Process
DP	Dynamic Programming
MC	Monte Carlo
TD	Temporal Differences
PoF	Pareto optimal Front
PID	Proportional Integral Derivative
FAST	Fatigue, Aerodynamics, Structure and Turbulence
NREL	National Renewable Energies Laboratory
MSE	Mean Squared Error
DM	Decision Making

Nomenclature

θ_{wind}	Direction of the wind
$\theta_{nacelle}$	Orientation of the nacelle
θ_{yaw}	Yaw angle
Ω_{yaw}	Yaw rotational speed
s	State of the RL algorithm
a	Action of the RL algorithm
r	Immediate reward of the RL algorithm
γ	Discount factor
$Q(s, a)$	Expected long-term reward matrix in RL algorithm
$Q\_P(s, a)$	Expected long-term power gain reward matrix in RL algorithm

$Q\_M(s, a)$	Expected long-term mechanical moment reward matrix in RL algorithm
$Q\_P(s(t), a(t))$	Expected long-term power gain reward function in RL algorithm
$Q\_M(s(t), a(t))$	Expected long-term mechanical moment reward function in RL algorithm

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. Nowadays, the most important renewable energy generation source is the wind energy. Many studies showing the positive tendency of the wind energy can be found in the literature. For example, according to some studies presented by Rosales-Asensio et al. [1], 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 [2] 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.

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, as explained in the work of Nyanteh et al. [3]. One main topic of this research work is the development of advanced control strategies to optimize the performance of the wind turbines [4–9].

In this chapter, the design procedure of a yaw control system of a Horizontal Axis Wind Turbine (HAWT) based on Machine Learning (ML) is presented. 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]. As observed in the works of Saenz-Aguirre et al. [5, 8], 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 Sect. 2. Section 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 Sect. 4. Finally, Sect. 5 presents the conclusions.

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. [12], 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 strategies aimed to regulate the rotational speed of the rotor in the whole range of operating points of the wind turbine. As a result of these 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 5 MW wind turbine, presented in the work of Jonkman et al. [13], is illustrated in Fig. 1.

The main control objective in the partial power zone, plotted in blue color in Fig. 1, is to maximize the power the wind turbine extracts from the wind, which can be expressed as in Eq. (1).

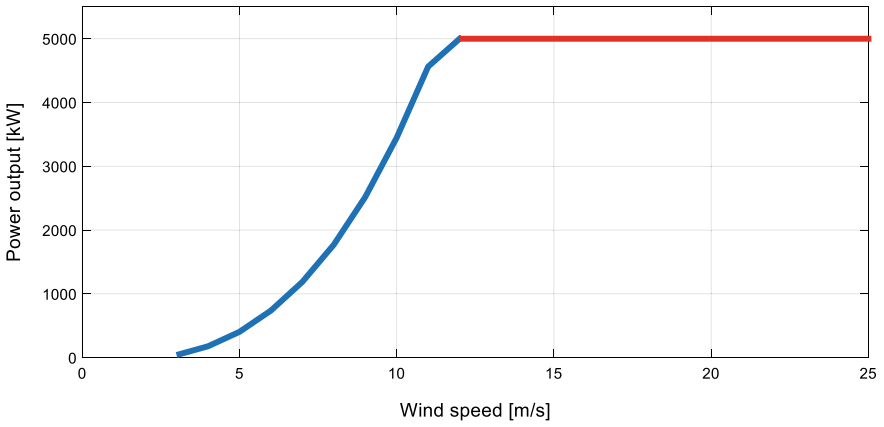

Fig. 1 Power curve of the NREL 5 MW wind turbine

$$P_{opt} = \frac{1}{2} \cdot \rho \cdot C_P \cdot A \cdot v^3 [W] \tag{1}$$

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 [m/s] 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. (2).

$$P = P_{opt} \cdot \cos^3\left(\theta_{yaw}\right)[W] \tag{2}$$

where θ_{yaw} [deg] is the yaw angle.

As it can be observed in Eq. (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 and Özder [14].

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 [15, 16]. Additionally, an analysis of the mechanical loads generated as a consequence of the yaw rotation is presented in the work of Shariatpanah et al. [17].

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.

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. 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. [18]. The most important AI techniques are the RL, ANNs, Fuzzy Logic, bio-inspired or metaheuristic optimization algorithms and Bayesian Networks. Each AI technique serves to a determined goal and could be used individually or in interrelation with other AI techniques.

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. [19]. 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 [19] 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 [19] 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 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 Sect. 3.1. Section 3.2 analyses the theory behind the ANNs. And, finally, the theoretical basis of the optimization algorithms is introduced in the Sect. 3.3.

3.1 Reinforcement Learning

RL [10, 20–22] 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 Jagodnik et al. [20], 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 Fig. 2. A defined agent which is in a determined environment receives information of

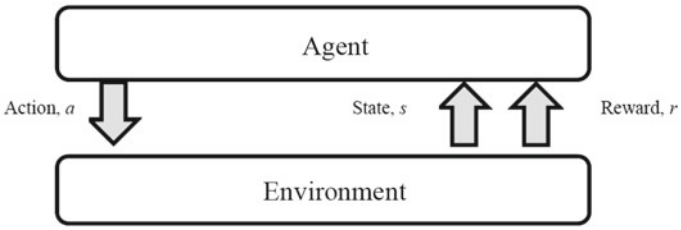

Fig. 2 Basic pipeline of a RL algorithm [5]

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. [23].

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. (3).

$$R_t = \sum_{k=t}^{t+T} \gamma_k \cdot r_{t+k+1} \tag{3}$$

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. (4).

$$E\left(\sum_{k=t}^{t+T} \gamma_k \cdot r_{t+k+1}\right) \tag{4}$$

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. (5) and Eq. (6), respectively.

$$p^a_{ss'} = p\left\{s_{t+1} = s' | s_t = s, a_t = a\right\} \tag{5}$$

$$R^a_{ss'} = E\left\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\right\} \tag{6}$$

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. (7) and Eq. (8), respectively.

$$V^\pi(s) = E_\pi\{R_t | s_t = s\} = E_\pi\left\{\sum_{k=t}^{t+T} \gamma_k \cdot r_{t+k+1} | s_t = s\right\} \tag{7}$$

$$Q^\pi(s, a) = E_\pi\{R_t | s_t = s, a_t = a\} = E_\pi\left\{\sum_{k=t}^{t+T} \gamma_k \cdot r_{t+k+1} | s_t = s, , a_t = a\right\} \tag{8}$$

The optimal values of both $V^\pi(s)$ and $Q^\pi(s, a)$ can be expressed as in Eqs. (9) and (10).

$$V(s) = \max(V^\pi(s)) \tag{9}$$

$$Q(s, a) = \max(Q^\pi(s, a)) \tag{10}$$

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. (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.

– **Dynamic Programming**

The DP method, explained in detail in the works of Bertsk et al. [24–26], is based on the knowledge of a model of the environment in which the agent is placed. That means that the state transitions $p^a_{ss'}$, see Eq. (5), and the reward transitions $R^a_{ss'}$, see Eq. (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 Eqs. (11) and (12).

$$V^\pi(s) = \sum_a \pi(s, a) \sum_{s_{t+1}} p^a_{ss'} \cdot [R^a_{ss'} + \gamma \cdot V^\pi(s_{t+1})] \tag{11}$$

$$Q^\pi(s, a) = \sum_a \pi(s, a) \sum_{s_{t+1}} p^a_{ss'} \cdot [R^a_{ss'} + \gamma \cdot Q^\pi(s_{t+1}, a_{t+1})] \tag{12}$$

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.

– **Monte Carlo method**

The MC method [27, 28] 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. (5), and the reward transitions $R^a_{ss'}$, see Eq. (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.

– **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. [29, 30], and SARSA, introduced in the work of Adam et al. [31].

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. (13).

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \cdot \left[r + \gamma \cdot Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right] \quad (13)$$

The calculation of the $Q(s, a)$ in Q-Learning algorithm can be expressed as shown in Eq. (14).

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \cdot \left[r + \gamma \cdot max_a Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right] \quad (14)$$

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 [32–34], 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 [35], 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 Fig. 3, is the smallest element of an ANN and presents the following structure:

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. (15).

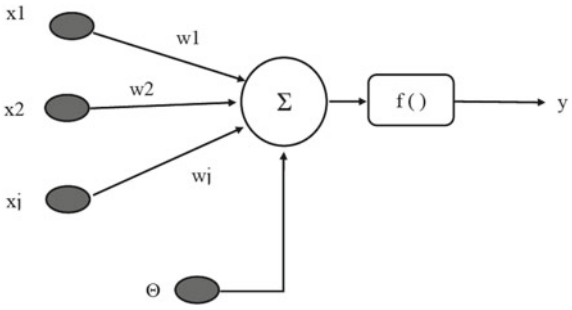

Fig. 3 Neuron of an ANN

$$h(x_1, \ldots, x_j, w_1, \ldots, w_j) = \sum_{j=1}^{n} w_j \cdot x_j - \theta \tag{15}$$

– 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. (16).

$$y = \begin{cases} 1 \quad if \sum_{j=1}^{n} w_j \cdot x_j \geq \theta \\ 0 \quad if \sum_{j=1}^{n} w_j \cdot x_j < \theta \end{cases} \tag{16}$$

However, when a continuous output of the neuron is desired, usually a sigmoid function [36] is used as the activation function, as shown in Eq. (17).

$$f(x) = \frac{1}{1 + e^{-\beta \cdot x}} \tag{17}$$

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., 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.

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. [32], 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. The most important application of the supervised learning is for regressions or modelling of systems and one of the most used examples of supervised learning is the BackPropagation algorithm. The unsupervised learning does 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 a hybrid learning strategy is also possible.

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 twentieth century. Nowadays, optimization algorithms are applied to a grand variety of applications. As it is explained in the work of Yang et al. [35], 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.

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 [37–39] have been widely studied in the literature. In the following lines the metaheuristic optimization algorithm used in the design process exposed in this chapter and the multivariable optimization is explained in detail.

– **Particle Swarm Optimization**

The PSO [40] 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 Khan and Singh [38], 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 and Singh [38], 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. (18) and Eq. (19), respectively.

$$v_{i,t+1} = H \cdot v_{i,t} + \varphi_1 \cdot \left(x\_opt_{i,t} - x_{i,t}\right) + \varphi_2 \cdot \left(x\_global\_opt_{i,t} - x_{i,t}\right) \quad (18)$$

$$x_{i,t+1} = x_{i,t} + v_{i,t+1} \cdot \Delta t \quad (19)$$

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. (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.

– **Multiobjective optimization. Pareto optimal Front**

A multiobjective optimization [41–43] 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 Pareto [44] 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. 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 Fig. 4.

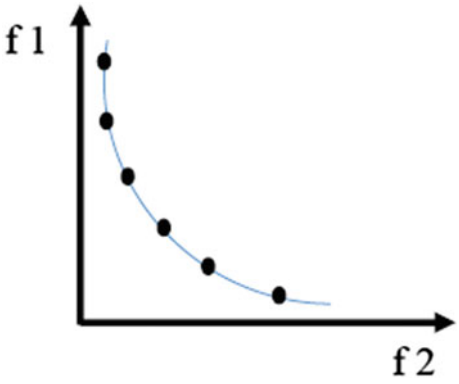

Fig. 4 PoF of a double-objective optimization problem

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 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.

Usually, classical control structures based on PIDs have been used for the design of the yaw control strategy of the wind turbine [17, 45]. 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 [5, 8, 46–48].

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 Fig. 5.

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 Sect. 4.1. Section 4.2 presents the structure and design process of the MLP-BP neural network. The design of the

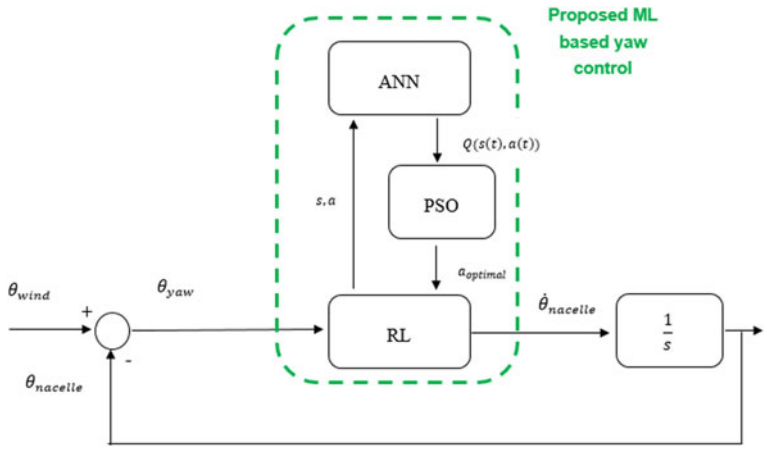

Fig. 5 Pipeline of the proposed ML based yaw control

PSO and PoF based algorithm is explained in Sect. 4.3. Finally, the Decision Making (DM) algorithm is exposed in Sect. 4.4.

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 is shown in Eq. (20).

$$\theta_{yaw} = \theta_{wind} - \theta_{nacelle} \tag{20}$$

– **StateWindS** [m/s]: This state defines the wind speed value incident to the rotor.

The actions a are:

– **ActionYawK** [-/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. (21).

$$\Omega_{yaw} = ActionYawK \cdot \theta_{yaw} \tag{21}$$

– **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. (22).

$$\Delta\theta_{yaw} \in [-\text{ActionYaw} \ , \ \text{ActionYaw} \ ] \tag{22}$$

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. (23).

$$RewardP = \frac{P\_control - P\_no\_control}{P\_no\_deviation} \cdot 100 \tag{23}$$

As it can be observed in Eq. (23), 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.

– **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 Sect. 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. (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. (24).

$$Q(s, a) = \sum_{i=0}^{i=T} r_{t+i} \cdot \gamma^i \tag{24}$$

The expression in Eq. (24) 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 Eqs. (25) and (26). 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.

$$Q\_P(s,a) = \frac{\frac{1}{T}\int_t^{t+T}(P\_control - P\_no\_control)\cdot dt}{\frac{1}{T}\int_t^{t+T} P\_no\_deviation \cdot dt} \cdot 100[\%] \tag{25}$$

$$Q\_M(s,a) = \int_t^{t+T} RewardM(t)\cdot dt[N\cdot m] \tag{26}$$

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 [49] and the wind turbine model NREL 5 MW, presented in the work of Jonkman et al. [13], 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 [-/s] 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 Eqs. (25) and (26).

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. 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 Fig. 6. Internally, the MLP-BP presents a structure

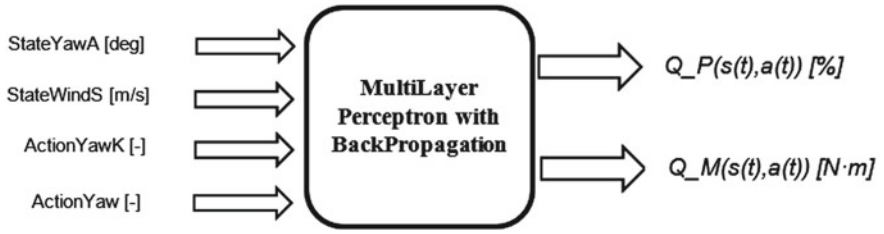

Fig. 6 Input and outputs of the MLP-BP designed for the ML based yaw control strategy

with one input layer with 4 neurons, two hidden layers with 75 neurons and 25 neurons respectively and one output layer with 2 neurons.

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.

4.3 Yaw Control PSO and PoF

As it was stated in Sect. 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.

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

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 [-/s] 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.

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.

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.

Acknowledgements 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.

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

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

References

1. Rosales-Asensio E, Borge-Diez D, Blanes-Peiro J, Perez-Hoyos A, Comenar-Santos A (2019) Review of wind energy technology and associated market and economic conditions in Spain. Renew Sustain Energy Rev 101:415–427
2. WindEurope (2019) Wind energy in Europe in 2018. Trends and Statistics
3. Nyanteh Y, Schneider N, Netter D, Wei B, Masson PJ (2015) Optimization of a 10 MW direct drive HTS generator for minimum levelized cost of energy. IEEE Trans Appl Supercond 25(3):1–4
4. Kim Y (2016) Robust data driven H-infinity control for wind turbine. J Franklin Inst 353(13):3104–3117
5. Saenz-Aguirre A, Zulueta E, Fernandez-Gamiz U, Lozano J, Lopez-Guede JM (2019) Artificial neural network based reinforcement learning for wind turbine Yaw control. Energies
6. Saenz-Aguirre A, Fernandez-Gamiz U, Zulueta E, Ulazia A, Martinez-Rico J (2019) Optimal wind turbine operation by artificial neural network-based active gurney flap flow control. Sustainability 11:2809
7. Aramendia I, Fernandez-Gamiz U, Zulueta E, Saenz-Aguirre A, Teso D (2019) Parametric study of a gurney flap implementation in a DU91W(2)250 airfoil. Energies
8. Saenz-Aguirre A, Zulueta E, Fernandez-Gamiz U, Ulazia A, Teso D (2019) Performance enhancement of the artificial neural network based reinforcement learning for wind turbine Yaw control. Wind Energy
9. Fernandez-Gamiz U, Zulueta E, Boyano A, Ansoategui I, Uriarte I (2017) Five megawatt wind turbine power output improvements by passive flow control devices. Energies 10(6):742
10. Zhang D, Han X, Deng C (2018) Review on the research and practice of deep learning and reinforcement learning in smart grids. CSEE J Power Energy Syst 4(3):362–370
11. Yang Z, Merrick K, Jin L, Abbass HA (2018) Hierarchical deep reinforcement learning for continuous action control. IEEE Trans Neural Netw Learn Syst 29(11):5174–5184
12. Bivona S, Bonanno G, Burlon R, Gurrera D, Leone C (2010) Stochastic models for wind speed forecasting. Stochas Models Wind Speed Forecast 52(2):1157–1165
13. Jonkman JM, Butterfield S, Musial W, Scott G (2009) Definition of a 5MW reference wind turbine for offshore system development. National Renewable Energy Laboratory (NREL)
14. Yücel M, Özder S, 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):74–87
15. Ahrens M, Kucera L, Larsonneur R (1996) Performance of a magnetically suspended flywheel energy storage device. IEEE Trans Control Syst Technol 4(5):495–502
16. Zheng S, Yang J, Song X, Ma C (2018) Tracking compensation control for nutation mode of high-speed rotors with strong gyroscopic effects. IEEE Trans Ind Electron 65(5):4156–4165
17. Shariatpanah H, Fadaeinedjad R, Rashidinejad M (2013) A new model for PMSG-based wind turbine with Yaw control. IEEE Trans Energy Convers 28(4):929–937
18. Wang X, Li X, Leung VCM (2015) Artificial intelligence-based techniques for emerging heterogeneous network: state of the arts, opportunities, and challenges. IEEE Access 3:1379–1391

19. Fadlullah ZM, Tang F, Mao B et al (2017) State-of-the-art deep learning: evolving machine intelligence toward tomorrow's intelligent network traffic control systems. IEEE Commun Surv Tutor 19(4):2432–2455
20. Jagodnik KM, Thomas PS, Bogert AJvd, Branicky MS, Kirsch RF (2017) Training an actor-critic reinforcement learning controller for arm movement using human-generated rewards. IEEE Trans Neural Syst Rehabilitat Eng 25(10):1892–1905
21. Mongillo G, Shteingart H, Loewenstein Y (2014) The misbehavior of reinforcement learning. Proc IEEE 102(4):528–541
22. Sutton RS, Barto AG (1998) Reinforcement learning: an introduction. MIT Press, Cambridge, MA, USA
23. Kaelbling LP, Littman ML, Moore AW (1996) Reinforcement learning: a survey. J Artif Intell Res 4(1):237–285
24. Bertsekas DP (2013) Abstract dynamic programming. Athena Scientific, Belmont, MA, USA
25. Bertsekas DP (2012) Dynamic programming and optimal control: approximate dynamic programming, no 2. Athena Scientific, Belmont, MA, USA
26. Bertsekas DP (2017) Value and policy iterations in optimal control and adaptive dynamic programming. IEEE Trans Neural Netw Learn Syst 28(3):500–509
27. Kao K, Wu I, Yen S, Shan Y (2013) Incentive learning in Monte Carlo tree search. IEEE Trans Comput Intell AI Games 5(4):346–352
28. Coulom R (2006) Efficient selectivity and backup operators in Monte-Carlo tree search. In: Proceedings of 5th international conference computer games, pp 72–83
29. Watkins CJCH (1989) Learning from delayed rewards. PhD thesis, King's College, Cambridge, UK
30. Watkins CJCH, Dayan P (1992) Q-learning. Mach Learn 8(3):279–292
31. Adam S, Busoniu L, Babuska R (2012) Experience replay for real-time reinforcement learning control. IEEE Trans Syst Man Cybernet Part C (Appl Rev) 42(2):201–212
32. Chen SH, Jakeman AJ, Norton JP (2008) Artificial Intelligence techniques: an introduction to their use for modelling environmental systems. Mathemat Comput Simul 78:379–400
33. Yao X (1999) Evolving artificial neural networks. Proc IEEE 87(9):1423–1447
34. Oong TH, Isa NAM (2011) Adaptive evolutionary artificial neural networks for pattern classification. IEEE Trans Neural Netw 22(11):1823–1836
35. Yang XS (2013) Optimization and metaheuristic algorithms in engineering. In: Yang XS, Gandomi AH, Talatahari S, Alavi AH (eds) Metaheursitics in water, geotechnical and transport engineering. Elsevier, pp 1–23
36. Jain AK, Mao J, Mohiuddin KM (1996) Artificial neural networks: a tutorial. Computer 29(3):31–44
37. Wang L, Shen J (2017) A systematic review of bio-inspired service concretization. IEEE Trans Serv Comput 10(4):493–505
38. Khan B, Singh P (2017) Selecting a meta-heuristic technique for smart micro-grid optimization problem: a comprehensive analysis. IEEE Access 5:13951–13977
39. Bala A, Ismail I, Ibrahim R, Sait SM (2018) Applications of metaheuristics in reservoir computing techniques: a review. IEEE Access 6:58012–58029
40. Kennedy J, Eberhar RC (1995) Particle swarm optimization. In: Proceedings IEEE international conference on neural network. Perth, WA, Australia, pp 1942–1948
41. Ehrgott M, Gandibleux X (2000) A survey and annotated bibliography of multiobjective combinatorial optimization. OR-Spektrum 22(4):425–460
42. Durillo JJ, Nebro AJ, García-Nieto J, Alba E (2010) On the velocity update in multi-objective particle swarm optimizers. In: Coello CA, Dhaenens C, Jourdan L (eds) Advances in multi-objective nature inspired computing. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 45–62
43. Coello Coello CA, Dhaenens C, Jourdan L (2010) Multi-objective combinatorial optimization: problematic and context. In: Coello CA, Dhaenens C, Jourdan L (eds) Advances in multi-objective nature inspired computing. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 1–21
44. Pareto V (1896) Cours D'Economie Politique, F. Rouge, Lausanne, I(II)

45. Karakasis N, Mesemanolis A, Nalmpantis T, Mademlis C (2016) Active yaw control in a horizontal axis wind system without requiring wind direction measurement. IET Renew Power Gener 10(9):1441–1449
46. Song D, Fan X, Yang J, Liu A, Chen S, Joo YH (2018) Power extraction efficiency optimization of horizontal-axis wind turbines through optimizing control parameters of yaw control systems using an intelligent method. Appl Energy 224:267–279
47. Song D, Yang J, Fan X et al (2018) Maximum power extraction for wind turbines through a novel yaw control solution using predicted wind directions. Energy Convers Manage 157:587–599
48. Bharani R, Jayasankar KC (2015) Yaw control of wind turbine using fuzzy logic controller. Power Electron Renew Energy Syst 326:997–1006
49. NREL NWTC FAST version 7. Available online: https://nwtc.nrel.gov/FAST7/. Accessed 21 Oct 2018