Wind farm distributed PSO-based control for constrained power generation maximization

Nicolo Gionfra, Guillaume Sandou, Houria Siguerdidjane, Damien Faille, Philippe Loevenbruck

Wind Farm Distributed PSO-based Control for Constrained Power Generation Maximization

Nicolò Gionfra[a], Guillaume Sandou[a,*], Houria Siguerdidjane[a], Damien Faille[b], Philippe Loevenbruck[c]

[a] *Laboratoire des Signaux et Systèmes – Centrale Supélec, CNRS, Université Paris Sud, Université Paris-Saclay, 3 rue Joliot Curie - 91192, Gif-sur-Yvette, France*
[b] *EDF R&D, Department PRISME, 6 quai Watier - 78401, Chatou, France*
[c] *EDF R&D, Department EFESE, 7 boulevard Gaspard Monge - 91120, Palaiseau, France*

Abstract

A novel distributed approach to treat the wind farm (WF) power maximization problem accounting for the wake interaction among the wind turbines (WTs) is presented. Power constraints are also considered within the optimization problem. These are either the WTs nominal power or a maximum allowed power injection, typically imposed by the grid operator. The approach is model-based. Coupled with a distributed architecture it allows fast convergence to a solution, which makes it exploitable for real-time operations. The WF optimization problem is solved in a cooperative way among the WTs by introducing a new distributed particle swarm optimization algorithm, based on cooperative co-evolution techniques. The algorithm is first analyzed for the unconstrained case, where we show how the WF problem can be distributed by exploiting the knowledge of the aerodynamic couplings among the WTs. The algorithm is extended to the constrained case employing Deb's rule. Simulations are carried out on different WFs and wind conditions, showing good power gains and fast convergence of the algorithm.

Keywords: Wind farm, Wake effect, Distributed optimization, Particle

*Corresponding author
Email addresses: nicolo.gionfra@centralesupelec.fr (Nicolò Gionfra), guillaume.sandou@centralesupelec.fr (Guillaume Sandou), houria.siguerdidjane@centralesupelec.fr (Houria Siguerdidjane), damien.faille@edf.fr (Damien Faille), philippe.loevenbruck@edf.fr (Philippe Loevenbruck)

swarm optimization

1. Introduction

1.1. Context and Motivation

In recent years, we have witnessed a relevant research effort in the field of wind energy production due to the high increase of installed capacity concerning wind farms. Developments in the field of control and optimization, along with the mature technology level of variable-speed variable-pitch variable-yaw wind turbines, have thus pushed the WF control targets further ahead towards a better exploitation of the wind source. In particular, the great adaptability of modern WTs to a wide range of wind conditions for the maximum power capture as well as a better understanding of the aerodynamic phenomena involved in the WFs, suggest to take in consideration the aerodynamic interaction among the WTs, when the power maximization of large wind farms is concerned. Indeed, when extracting kinetic energy from the wind, a WT causes a reduction of the wind speed in the downstream wake. As a result a turbine, standing in the wake of an upstream one, experiences a reduction of available wind power. Intuitively as the number of wind turbines of a wind farm increases, such phenomenon becomes more important, so that considering it when optimizing the wind production proves potential gain with respect to classic individual turbine maximum power point tracking (MPPT) mode. As a matter of fact, in such situation, a *greedy* control, according to which each WT tracks its *own* maximum available power, no longer guarantees the maximization of the power extraction at the WF level. This mainly justifies a growing interest in *cooperative* methods to control wind turbines belonging to large wind farms, e.g. [1]. These rely on a WF controller in order to manage the cooperative power production sharing among the WTs. Such controller has to deal with a large quantity of information as it has to process and communicate a growing number of variables with the increasing number of WT units, while respecting more and more real-time operation constraints. As a result, the depicted situation suggests the employment of a *distributed* control architecture among the WTs, rather than its more classic centralized counterpart. Indeed, a large-size WF can be naturally defined as a multi-agent system (MAS), where its WTs represent the system agents. Even though treating the WF control problem in the MAS framework may lead to suboptimal solutions with respect to the centralized

approaches, these latter may be unfeasible if we aim at real-time operations. As a matter of fact, distributed systems allow the important feature of *reducing the computational and communication burden*. Moreover, they typically lead to a system that is scalable, modular, and resilient, which is among the reasons why the distributed approaches are rapidly gaining popularity within the more general framework of distributed energy generation, [2, 3].

1.2. Related Works

Modeling the aerodynamic coupling among the WTs of a WF, i.e. the wake interaction, also known as *wake effect*, with high fidelity degree is not a trivial task. This involves numerical solutions to systems of partial differential equations. In other words, in principle, computational fluid dynamics (CFD) tools should be employed. See for instance [4] for large eddy simulation (LES) tools for WF applications. However, for the sake of engineering applications, typically the use of simplified *static* models for the prediction of WT wakes is preferred as it enables reduced computational cost, [5]. This is also the case for off-line WF applications, such as the layout optimization problem. In this regard, one can cite for instance the works of [6, 7, 8, 9]. If the use of CFD tools is impractical for off-line applications, this reveals to be also true for real-time WF operations, where wake effect has to be recursively recalculated, [10, 11]. Thus LES is either employed for model parameters tuning in a grey-box approach as in [1, 6, 11], or for control validation as in [12, 13, 14].
When the WF layout is given, still optimal control of a WF can be seek via optimization and control methods. The research literature is vast in this regard, and it can be roughly divided in two main groups, namely *centralized model-based* and *distributed model-free* approaches. The former typically make use of the mentioned simplified wake models as they are suitable for real-time control algorithms, [14]. These in turns rely on a WF problem that is generally formulated as an optimization one. The concerning literature works differ according to the used model, optimization algorithm, and WT variables of control. These latter are typically the yaw angle, and either the axial induction factor, or a combination of pitch angle and tip speed ratio. Authors of [15] also propose methods based on individual pitch control to induce both yaw and tilt moment to redirect turbine wakes. An exception is presented by the works of [16] and [5], where the authors tackle the WF power maximization problem by simultaneously acting on the WT relative positions and on their control variables. Among the gradient-based tech-

niques used to solve the WF problem, one can cite the works of [1, 5, 17, 18]. Authors of [1] capitalize on sequential convex programming techniques, with yaw influence consideration, enabled by the choice of a continuous and differentiable wake model. In [18], authors aim at speeding up the optimization problem convergence, by using a sequential optimization method. In particular, with the knowledge of the upstream free wind direction, they exploit the problem structure by letting only the parameters of a WT to be active at a time. A similar approach is proposed by [5] where the employed dynamic programming technique reflects the structure of the problem. Since the WF optimization problem happens to be nonconvex, the aforementioned optimization approaches suffer from convergence to local optima, and their solution is affected by the initial guess. This is why *global optimization* methods have been proposed for the sake of maximizing the WF power production. In particular, centralized metaheuristic algorithms have been used in [19], where the well-known technique of particle swarm optimization (PSO) is employed, and in [10, 13, 16], where the authors capitalize on the genetic algorithm (GA).

It is worth mentioning that the addressed optimization problem in the aforementioned works is usually concerned with the *unconstrained* WF power maximization one. In other words, neither the system physical constraints such as the WTs nominal power, and the one imposed by the grid operator, are directly considered in the problem formulation. Thus, the provided solutions are confined to the case in which the total available WF power is lower than the maximum allowed one. Alternatively, one can *a posteriori* constrain the solution of the unconstrained problem. However, this generally leads to a suboptimal solution. As it will be explained in the following sections, adding the mentioned constraints to the optimization problem brings it to a higher level of complexity, because they generally define a nonconvex feasible set. Exceptions can be found for instance in the metaheuristic approaches of [10, 19], where the authors consider constraints within the optimization problem formulation. Unfortunately it is not mentioned how these are managed in the employed algorithms, which is known to be a nontrivial problem in metaheuristic optimization.

One of the main shortcomings of the model-based approach is the need for simplified assumptions in the description of the system physics, which can lead to discrepancies between model and actual aerodynamic phenomena. This fact paved the way to model-free approaches. Moreover, due to their intrinsic algorithm logic, they are usually well-suited for distributed imple-

mentations, which can bring the important features listed in Subsection 1.1. One can cite [20, 21] for methods based on gradient estimation, [22] as far as game-theoretic approaches are concerned, and [12] for a Bayesian approach. Similarly to the mentioned model-based methods of [5, 18], authors of [14] propose a model-free nested extremum seeking technique to exploit the WF problem structure. The main drawback of the mentioned data-driven distributed approaches mainly regards the speed of convergence. This is due to the fact that the WF control parameters have to be first tested on the real plant in order to evaluate their real effect.

1.3. Contribution of the Paper

In this paper we tackle the WF power maximization from a MAS perspective. In particular, our main contribution is the proposal of a *model-based distributed* approach to control a wind farm. The aim is to blend the advantages of both the centralized model-based and distributed model-free methods mentioned in Subsection 1.2. Among these, one of our scopes is to reach fast convergence for real-time operations that a simpler centralized model-based architecture may fail to achieve in the case of large-size WFs, [10].

The second contribution regards the introduction of a novel distributed PSO (DPSO) algorithm to let the WTs cooperate via only local interactions. The employment of metaheuristic algorithms in the framework of WF control under wake effect is largely justified by their proven good performance in the literature for both layout optimization, as in [9, 23, 8], and in control optimization applications, as in [10, 13, 16, 19]. As it will be treated more in details in the sequel, the proposal of a new DPSO algorithm is necessary as, to the authors' knowledge, there does not exist a ready-to-use distributed PSO solution capable of handling a multi-agent system whose agents share a common optimization variable.

Eventually the presented algorithm allows to take system constraints into account within the problem formulation. Specifically, these are handle by employing *Deb's rule*, [24].

The remainder of the paper is organized as follows. Firstly, the wake model is presented in Section 2. The WF optimization problem, together with the considered reduced formulation, is stated in Section 3. In Section 4 we recall and discuss some basic concepts concerning cooperative co-evolution techniques on which DPSO is based. DPSO algorithm for the WF problem

is detailed in Section 5. We carry out simulations to test the algorithm performance on different WFs and wind conditions in Section 6. The paper end with conclusions and further perspectives in Section 7.

2. Wind Farm Power Function and Wake Model

Even though in the literature there exist many simplified analytic wake model representations, many of them are variants of the pioneer work of [25], who first proposed the single wake model, i.e. describing the wake interaction between two WTs, and [26], who introduced the wake model accounting for multiple WTs interactions. This, also known as *Park model*, describes a piece-wise linear wind speed profile distribution within a WF, and it is based on the assumption that the wake behind the WTs expands linearly, according to a *wake constant* k_w, in the downstream direction, and that the wind inside the wake is constant at equal distance from the according WT. This provides a top-hat shape to the wake velocity deficit.
In this paper we make use of a wake model based on the work of [1, 11]. In particular, the authors of [11] propose a *parametric* model to be tuned on WF real data or via LES. Based on it, authors of [1] provide a continuous and differentiable analytic model, thanks to the choice of a Gaussian shape for the crosswind direction wind velocity profile, instead of the top-hat one of the classic Park model. Notice that a similar approach has been used e.g. in [13] for a Gaussian shape, and in [27] for a cosine one. Moreover, both [1, 11] allow to take into account the WT yaw angle influence in the wake direction.
The aim of this section is to provide the main modeling steps required to build the WF power function. The reader may refer to [1, 11] for further details.

2.1. Single Wake Model

The power P extracted by a WT from the upstream wind can be expressed as

$$P = \frac{1}{2}\varrho\pi R^2 v^3 4\alpha(\cos(\tau_1 o) - \alpha)^2 \tag{1}$$

where ϱ is the air density, R the rotor plane radius, v the upstream wind speed, α is the WT axial induction factor and defined as the ratio of the difference between the wind speed value in the axial direction and the wind

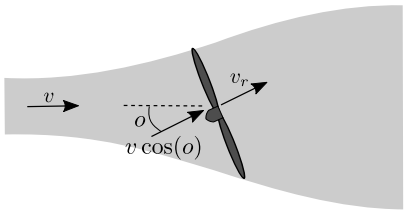


Figure 1: Wind flow and wind speed at rotor disc, $v_r = v(\cos o - \alpha)$, [1].

speed behind the rotor plane v_r, and the upstream value, o is the WT yaw offset with respect to the upstream wind direction, as shown in Fig. 1. $\tau_1 \in \mathbb{R}$ is the first introduced model parameter. The term $C_p(\alpha, o) \triangleq 4\alpha(\cos(\tau_1 o) - \alpha)^2$ is also known as the WT power coefficient, and it has a maximum theoretical value equal to $\frac{16}{27}$, corresponding to a zero yaw offset and $\alpha = \frac{1}{3}$. In such case the WT is operated in MPPT mode.
Park wake model describes the wind speed value in a downstream wake, by introducing a deficit factor $\delta_v(d, r, \alpha)$, i.e. function of the distance d behind the wake, the radial wake distance r, and the axial induction factor α of the WT causing the wake. Such wind speed value is computed according to

$$v(d, r, \alpha) = v(1 - \delta_v(d, r, \alpha))$$

In such model, it is assumed that the radius of the wake linearly increases with the downstream distance, according to $R(d) = R + \kappa_w d$, where the wake expansion rate κ_w is a constant depending on the surface roughness of the WF site. Deficit δ_v is then assumed to be null outside the wake, constant inside on equal distance values d, and proportional to the ratio between the area of the rotor plane, πR^2, and the wake plane area at d, $\pi R^2(d)$, having value $2\alpha \left(\frac{R}{R(d)}\right)^2$. This is where Park model introduces a discontinuity. However, it has been observed that for distance values $d > 10R$, the cross sectional wind speed profile resembles a Gaussian function [1]. Thus, δ_v can be modeled as

$$\delta_v(d, r, \alpha, o) = 2\alpha \cos(\tau_2 o) \left(\frac{R}{R(d)}\right)^2 e^{-\left(\frac{r}{R(d)}\right)^2} \tag{2}$$

where the exponential term defines the Gaussian shape, and where it has been further introduced the factor accounting for the yaw effect. $\tau_2 \in \mathbb{R}$ is the second model parameter. The aforementioned considerations on the

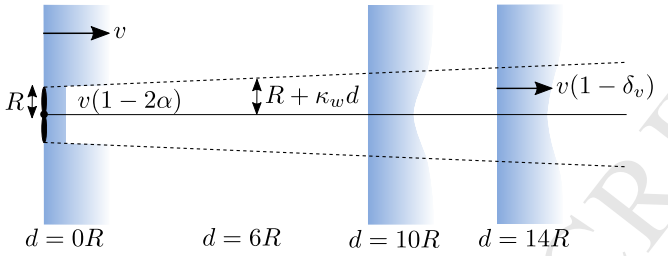


Figure 2: Gaussian wake expansion model, [1].

wake expansion, and shape behind the WT are depicted in Fig. 2. The wind speed reduction experienced by a downstream WT i, caused by an upstream WT j, intuitively depends on the wake position with respect to the downstream WT rotor plane. This can be modeled by introducing two parameters, namely d_{ij}, and r_{ij}. The former represents the axial distance according to the free stream wind direction θ^W, while the latter represents the radial distance of the wake centerline from WT i rotor plane center. Let us consider an *absolute* reference frame (x, y). We can thus define the position of each turbine i via its coordinates (x_i, y_i). The wind direction θ^W in front of the considered upstream WT j can be defined with respect to the mentioned reference frame. Then, we can consider a *rotated* reference (x', y') by an angle equal to θ^W, given by the coordinate transformation

$$\begin{cases} y' = x \sin\theta^W + y \cos\theta^W \\ x' = x \cos\theta^W - y \sin\theta^W \end{cases} \tag{3}$$

Both the absolute and rotated reference frame are shown in Fig. 3. On this basis, we can compute the distance d_{ij} as $d_{ij} = |y'_i - y'_j|$, while r_{ij} is given by three terms as follows

$$r_{ij} = r^l_{ij} + r^r_{ij} + r^o_{ij}$$

and where $r^l_{ij} = |x'_i - x'_j|$ represents the radial wake distance due to the relative locations of WT i, and j with respect to θ^W, i.e. in the rotated coordinates. The second term of r_{ij}, i.e. r^r_{ij}, allows to take into account the wake deviation caused by the rotating blades. In [11], it is shown to be linearly dependent on d_{ij} via a tunable parameter $\tau_3 \in \mathbb{R}^+$. Moreover, without loss of generality, we consider a positive deflection towards the right of the wake centerline, i.e.

$$r^r_{ij} = \tau_3 d_{ij} sign(x'_i - x'_j)$$

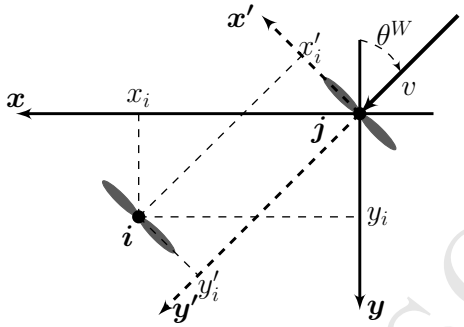


Figure 3: Absolute and rotated reference frame according to the free stream wind direction θ^W.

where $sign(x'_i - x'_j) = 1$ if $x'_i - x'_j \geq 0$, and -1 otherwise. The last term in r_{ij}, r^o_{ij}, is due to the yaw offset angle of WT j, o_j. The wake is deflected by the yaw, then, influenced by the free stream, it deviates in its direction. The result is a *curved* trajectory. Authors of [28] describe this curvature at a generic downstream distance d via the angle of the centerline of the wake $\xi(d)$, which assumes the following expression [11]

$$\xi(d) = \frac{\xi_{init}(\alpha_j, o_j)}{\left(1 + \frac{\tau_4 d}{R}\right)^2}$$

where $\tau_4 \in \mathbb{R}^+$ is left as an additional tunable parameter, and $\xi_{init}(\alpha_j, o_j) \triangleq \frac{1}{2}(\cos o_j)^2(\sin o_j)4\alpha_j(1-\alpha_j)$. Moreover, the yaw-induced lateral offset $\delta_\gamma(d_{ij})$ can be computed by solving the following integral via approximated Taylor expansion

$$\delta_\gamma(d_{ij}) = \int_0^{d_{ij}} \tan(\xi(x))dx$$

A positive yaw offset o_j increases the yaw-induced offset. However, this latter can increase or decrease the total value of r_{ij} depending on the relative locations of the wind turbines and the wind direction. According to the above considerations, r^o_{ij} can be computed as

$$r^o_{ij} = |\delta_\gamma(d_{ij})| sign(x'_i - x'_j)$$

The three cumulative effects composing r_{ij} are shown in Fig. 4. We are now

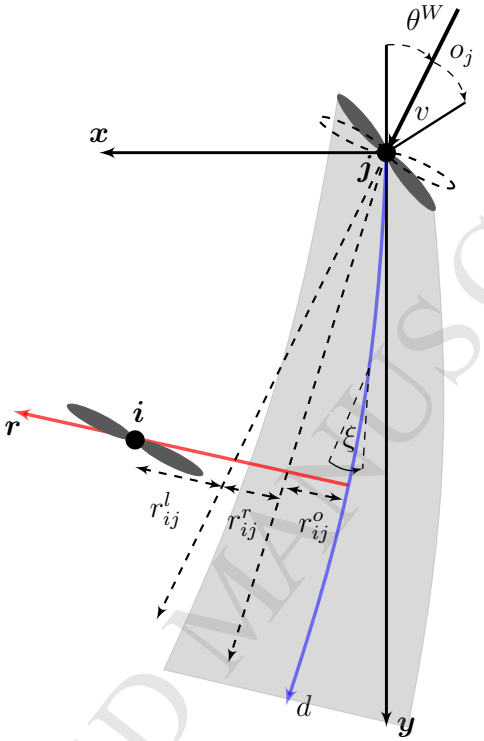


Figure 4: Radial distance of WT j wake centerline from WT i rotor plane, due to three effects: r^l_{ij}, r^r_{ij}, and r^o_{ij}, [1].

able to determine the wind speed deficit of a WT i caused by the *single wake* of WT j. This is done by using wind deficit δ_v expression of (2), and d_{ij}, r_{ij}, defined above. The wind speed value captured by WT i, v_{ij}, can be computed at any generic point on its rotor disc by defining the local polar coordinates (r', θ'), (see [1] for details). This is given by

$$v_{ij}(r', \theta', \alpha_j, o_j, v, \theta^W) = v(1 - \delta_v(d_{ij}, r, \alpha_j, o_j))$$

where δ_v is computed in $(d_{ij}, r, \alpha_j, o_j)$, being $r = \sqrt{(r_{ij} - r'\cos\theta')^2 + (r'\sin\theta')^2}$. Notice that d_{ij}, and r, on which δ_v depends, are function of (α_j, o_j), and θ^W, but not of v. For the purpose of simplification, the *average* wind speed $\bar{v}_{ij}$ on the rotor disc of the downstream WT i can be computed by applying the

mean value theorem for integrals. This is given by

$$\bar{v}_{ij}(\alpha_j, o_j, v, \theta^W) = \frac{1}{\pi R^2} \int_{\theta'=0}^{\theta'=2\pi} \int_{r'=0}^{r'=R} v_{ij}(r', \theta', \alpha_j, o_j, v, \theta^W) r' dr' d\theta'$$

From the above expression, the power extracted from the downstream WT is

$$P_i = \frac{1}{2} \varrho \pi R^2 \bar{v}_{ij}^3(\alpha_j, o_j, v, \theta^W) C_p(\alpha_i, o_i) \tag{4}$$

We are able to see how P_i is function of the according WT i operating conditions via C_p, as well as function of the upstream WT j ones via the wake model, defining $\bar{v}_{ij}(\alpha_j, o_j, v, \theta^W)$. An example of average wind deficit $\bar{\delta}_{v,ij} \triangleq 1 - \frac{\bar{v}_{ij}}{v}$ as a function of d_{ij}, and r_{ij}, caused by a WT j with $o_j = 0$, $\alpha_j = \frac{1}{3}$, is given in Fig. 5.

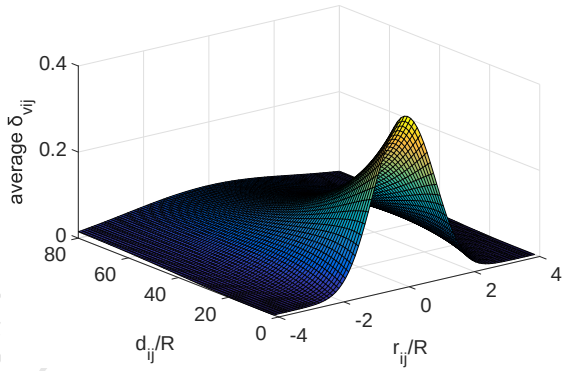


Figure 5: Wind deficit as a function of d_{ij} and r_{ij} caused by WT j on WT i, when $o_j = 0, \alpha_j = \frac{1}{3}$.

2.2. Multiple Wake Model and WF Power Function

In a wind farm, a wind turbine i is likely to experience a wind speed deficit caused by *multiple* wakes, i.e. from all the upstream WTs. In order to take into account the wake interference among multiple WTs, the kinetic energy conservation method proposed by [26] in Park model, is by far the most employed one. In this model it is assumed that *the kinetic energy deficit by the mixed wake is equal to the sum of the kinetic energy deficits by individual*

wakes, [1]. Before providing the expression of the wind deficit in the multi-wake case, it is useful to provide the following notation as it will be employed in the algorithm discussion in Section 5. According to the wake model, each WT i is physically coupled to any upstream WT j, since the average captured wind speed $\bar{v}_{ij}$, affecting turbine i power function via (4), is function of WT j operating points, i.e. α_j, and o_j. We say that WT j is a *physical neighbor* of WT i. Notice that the vice versa is not true. If we name $\mathcal{V} \triangleq \{1, \dots, N\}$ the set of N WTs composing a WF, for each WT i we are thus able to define an associated *physical neighborhood* as $\mathcal{N}_i^p \triangleq \{j \in \mathcal{V} : y_i' - y_j' > 0\}$, i.e. if WT j is an upstream turbine to i. We can also define the vector of variables associated to the WTs in the neighborhood as $\boldsymbol{\alpha}_{ij} \triangleq \{\alpha_j : j \in \mathcal{N}_i^p\}$, and $\boldsymbol{o}_{ij} \triangleq \{o_j : j \in \mathcal{N}_i^p\}$. Notice that $\mathcal{N}_i^p(\theta^W)$, i.e. it is a function of the wind farm free stream wind direction θ^W via (3). Thus, if θ^W is a function of time, then $\mathcal{N}_i^p$ is time-varying too. According to the mentioned method of conservation of kinetic energy, we can thus describe the total average wind speed deficit $\bar{\delta}_{v,i}$ experienced by WT i in a WF as

$$\bar{\delta}_{v,i} = \sqrt{\sum_{j \in \mathcal{N}_i^p} \bar{\delta}_{v,ij}^2} \tag{5}$$

If we now name v_∞ the free stream wind speed blowing towards the wind farm, we have that the average wind speed captured by WT i is

$$\bar{v}_i(\boldsymbol{\alpha}_{ij}, \boldsymbol{o}_{ij}, v_\infty, \theta^W) = v_\infty(1 - \bar{\delta}_{v,i}(\boldsymbol{\alpha}_{ij}, \boldsymbol{o}_{ij}, \theta^W))$$

As a result, its available power is

$$P_i(\alpha_i, o_i, \boldsymbol{\alpha}_{ij}, \boldsymbol{o}_{ij}, v_\infty, \theta^W) = \frac{1}{2}\varrho\pi R^2 \bar{v}_i^3(\boldsymbol{\alpha}_{ij}, \boldsymbol{o}_{ij}, v_\infty, \theta^W) C_p(\alpha_i, o_i) \tag{6}$$

Eventually, the total wind farm available power P_{wf}, given the free stream wind parameters (v_∞, θ^W), and the WTs operating conditions α_i, o_i, $\forall i \in \mathcal{V}$, can be simply computed by summing the single power productions, yielding

$$P_{wf} \triangleq \sum_{i=1}^{N} P_i(\alpha_i, o_i, \boldsymbol{\alpha}_{ij}, \boldsymbol{o}_{ij}, v_\infty, \theta^W) \tag{7}$$

Remark 1. *According to the considered wake model, the wind deficit experienced by each turbine is function of the free stream wind direction θ^W, but* not *of its speed value v_∞. This influences the power production, but it intervenes as a* factor *in (7). This fact has important consequences in the optimization problem formulation, as it will be discussed in the next sections.*

Remark 2. *Wake model parameters k_w, τ_1, τ_2, τ_3, and τ_4 can be tuned via real-data identification as suggested by [11], or via CFD simulation. In this work we make use of the parameter values provided in [1], where these are calibrated using SOWFA, a CFD tool introduced by [4]. The analytic model proves to match well with CFD simulation data for yaw offset angles in the range of $o = \pm 30°$.*

3. Wind Farm Problem Formulation

3.1. Problem Statement

The main wind farm problem of interest is the one of maximizing the WF power production. In the literature this usually takes the form of an *unconstrained* optimization problem, which can be formulated as follows

$$\min_{(\boldsymbol{\alpha},\boldsymbol{o})\triangleq\{(\alpha_i,o_i),i\in\mathcal{V}\}} -P_{wf}(\boldsymbol{\alpha},\boldsymbol{o},v_\infty,\theta^W) \tag{8}$$

Some considerations need to be done concerning problem (8). First of all, notice that according to Remark 1, it follows that its argument $(\boldsymbol{\alpha}^\star,\boldsymbol{o}^\star)$, i.e. the optimal optimization variables, *does not* change if only the wind speed value v_∞ does. This is due to the fact that the cost function in (8) has invariant minima with respect to v_∞, as it appears as a factor in it. The important practical implication is that, in this case, the optimization needs to be performed only when the wind direction θ^W changes. In a real world implementation though, the solution to (8) may not be feasible due to system constraints. We identify two main constraint sources, which will be considered in this work, namely

a) Physical constraints, i.e. WT nominal power P_n.
b) Grid constraints, expressed in the form of maximum allowed power injection in the grid, P_{wf}^{max}.

The interest in considering power constraints in point b, also known as *power curtailment*, comes from recent needs of letting a WF participate to grid operations, (see e.g. [29]). Furthermore, authors of [30] show the economic advantage of power curtailment as a possible alternative solution to grid reinforcement for the integration of renewable energies.
From a practical point of view, the optimal solution to (8) can be applied to the system only if, in the absence of a P_{wf}^{max} constraint, the free stream wind speed value v_∞ is such that $P_i(\boldsymbol{\alpha}^\star,\boldsymbol{o}^\star) \leq P_n$, $\forall i \in \mathcal{V}$. Another approach that

could be considered is to constrain the solution of (8) *a posteriori*. However, in this case, being the optimization problem nonconvex, and because the considered power constraints generally define a nonconvex feasible set, such solution is likely to be suboptimal. For this reason, we rather consider such constraints actively in the optimization problem, by modifying (8) as follows

$$\begin{aligned} &\min_{(\boldsymbol{\alpha},\boldsymbol{o})} -P_{wf}(\boldsymbol{\alpha},\boldsymbol{o},v_\infty,\theta^W) \\ &\text{subject to} \\ &P_i(\boldsymbol{\alpha},\boldsymbol{o},v_\infty,\theta^W) \leq P_n, \quad \forall i \in \mathcal{V} \\ &P_{wf}(\boldsymbol{\alpha},\boldsymbol{o},v_\infty,\theta^W) \leq P_{wf}^{max} \end{aligned} \tag{9}$$

It is important to notice that, differently from (8), problem (9) has to be solved each time that either v_∞, or θ^W changes, as v_∞ modifies the feasible region via the problem constraints. In this case it appears even clearer the importance of algorithm fast convergence to solve (9) with real-time performance. However, for real implementations it seems reasonable to consider an optimization step with respect to *averaged* values of wind speed and direction signals, in order to grasp their main trends affecting the wind farm power function, and to sufficiently filter out the turbulence components. For this reason, for now on, we will refer to (v_∞, θ^W) as the filtered values of the original wind speed and direction signals. For instance, from wind measurements, this can be achieved via a moving average filter. Eventually we consider the additional

Assumption 1. *The couple* (v_∞, θ^W) *is* uniform *along the wind farm length.*

3.2. Yaw Angle Presetting

As shown in the wake model of Section 2, the yaw angle influences the wake shape trajectory, and for this reason it should be considered as an optimization variable, as done in the optimization problem formulation in (9). However, since the yaw offset can be controlled according to dynamics which are slow when compared to the axial induction factor ones, and being interested in real-time control capabilities, in this work we make the choice to preset the WTs yaw angle, and to leave the WTs axial induction factor as the only WF optimization variable. For instance, the yaw values can be thought as being set according to a higher optimization step which only takes into account slow wind variations. In this paper, for the sake of the proposed

algorithm analysis, we make the simple choice to set $o_i = 0$, $\forall i \in \mathcal{V}$, i.e. the yaw angle γ_i of each WT is $\gamma_i = \theta^W$. In this case the WTs are always oriented such that their rotor plane is perpendicular to the wind direction. This choice is generally suboptimal as the yaw angle has an active role in providing the optimal solution when constraints are not active and in the presence of the wake effect. In the light of the above considerations, the WF optimization problem considered in the sequel reduces to

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} -P_{wf}(\boldsymbol{\alpha}, v_\infty, \theta^W) \\ &\text{subject to} \\ &P_i(\boldsymbol{\alpha}, v_\infty, \theta^W) \leq P_n, \quad \forall i \in \mathcal{V} \\ &P_{wf}(\boldsymbol{\alpha}, v_\infty, \theta^W) \leq P_{wf}^{max} \end{aligned} \tag{10}$$

where we removed the explicit dependence upon $\boldsymbol{o}$ as it is set equal to $\mathbf{0}$.

Remark 3. *Since the yaw effect is considered in the wake model, the results of Section 5 can still be applied if $\boldsymbol{o}$ is pre-assigned with a value which is different from $\mathbf{0}$.*

4. Preliminaries on PSO and Cooperative Co-evolution

Before providing the *ad hoc* distributed PSO algorithm to solve WF problem (10) we need to introduce some basic concepts concerning a particular implementation of the classic PSO algorithm. This is based on a technique known as *cooperative co-evolution* (CC) on which our distributed PSO version is founded. The CC technique was first introduced in [31], while its first application to centralized PSO can be found in [32].

4.1. CC-PSO Algorithm

Let us consider the following unconstrained optimization problem

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} F(\boldsymbol{x}) \tag{11}$$

where $F : \mathbb{R}^n \to \mathbb{R}$. In classic PSO, a set of N_p particles, $p = 1, \ldots, N_p$, is associated to problem (11), each of which represents a *position* $\boldsymbol{x}_p$ in the search space. During the run of the algorithm, these move according to a prescribed law of evolution, and they update and store in memory their latest *personal best* visited position $\boldsymbol{b}_p$ in respect of the cost function F to be

minimized. Moreover, they have access to the other particles personal best and the according cost function value. On the basis of this knowledge then each particle is able to elect the best particle among the swarm particle bests, i.e. the global best $\boldsymbol{g}$.

In cooperative co-evolution PSO (CC-PSO) the optimization problem is simplified by dividing it in smaller sub-problems. The main idea, known also as *divide and conquer* strategy, can be summarized in three important steps, namely *problem decomposition*, *subcomponent optimization*, and *cooperative combination*, [33]. In the first step, the n-dimensional problem is divided in N sub-problems of lower dimension, i.e. the vector $\boldsymbol{x}$ is decomposed in N subcomponents $x_i \in \mathbb{R}^{n_i}$, $i = 1, \ldots, N$, each of which represents a non-overlapping subset of the dimension of the search space. A swarm of N_p particles is associated to each subcomponent. In order to evaluate each subcomponent, i.e. how each of its particles fits the cost function F, a *context vector* has to be constructed, [32]. This is usually done by concatenating each subcomponent particle with the global best particles of the other subcomponents swarms, [34]. This practice will be detailed in the algorithm description.

In the original CC-PSO formulation, the swarms belonging to each subcomponent are processed in *sequential* order, i.e. at any time step of the algorithm, only one population is active. In this case we say that the swarm *update timing* is sequential, [35]. This strategy guarantees the objective function F to be strictly nonincreasing in the best solutions found during the run of the algorithm, [32]. Nonetheless, in view of distributing the optimization among the agents, where the goal is to reduce the run time, we are interested in a *parallel* update timing, i.e. each agent swarm is active at any algorithm time step, without the need for waiting its own turn. In this work we focus on this latter case, where each swarm is updated according to the following classic PSO equations

$$\begin{cases} s_{i,p}(k+1) = \omega s_{i,p}(k) + \phi_{1_{i,p}}(k)(g_i(k) - x_{i,p}(k)) \\ \qquad\qquad + \phi_{2_{i,p}}(k)(b_{i,p}(k) - x_{i,p}(k)) \\ x_{i,p}(k+1) = x_{i,p}(k) + s_{i,p}(k+1) \end{cases} \tag{12}$$

where k is the current algorithm step, $x_{i,p}$, $s_{i,p}$, and $b_{i,p}$ are respectively the position, the speed and the personal best position values associated to the p-th particle of the i-th subcomponent, g_i is the global best of the i-th swarm, $\phi_{1_{i,p}}$ and $\phi_{2_{i,p}}$ are two aleatory variables with uniform distribution of

probability in the respective intervals $[0, c_1]$, and $[0, c_2]$, where $c_1, c_2 \in \mathbb{R}^+$, and ω is the inertia factor. See for instance [36] for the tuning of these parameters. The overall CC-PSO algorithm is described in Algorithm 1 for the generic i-th swarm.

Algorithm 1 CC-PSO with parallel update timing

Output: global best: g_i

Initialization :

randomly initialize $x_{i,p}$, $s_{i,p}$, $p = 1, \dots, N_p$

$b_{i,p} = x_{i,p}$, $p = 1, \dots, N_p$

randomly initialize g_i

LOOP Process

for $k = 1$ to `max_iter` **do**

compose the context vectors associated to the particles $x_{i,p}$: $\boldsymbol{x}_{i,p}^g \triangleq (g_1, \cdots, x_{i,p}, \dots, g_N)$, $p = 1, \dots, N_p$

compose the context vectors associated to the personal bests $b_{i,p}$: $\boldsymbol{b}_{i,p}^g \triangleq (g_1, \cdots, b_{i,p}, \dots, g_N)$, $p = 1, \dots, N_p$

evaluate the particle context vectors in F: $F_{i,p}^x \triangleq F(\boldsymbol{x}_{i,p}^g)$, $p = 1, \dots, N_p$

evaluate the personal best context vectors in F: $F_{i,p}^b \triangleq F(\boldsymbol{b}_{i,p}^g)$, $p = 1, \dots, N_p$

update personal bests for $p = 1, \dots, N_p$

$$\left(b_{i,p}^{new}, F_{i,p}^{b,new}\right) = \begin{cases} \left(x_{i,p}, F_{i,p}^x\right) & \text{if } F_{i,p}^x < F_{i,p}^b \\ \left(b_{i,p}, F_{i,p}^b\right) & \text{otherwise} \end{cases}$$

$b_{i,p} = b_{i,p}^{new}$; $F_{i,p}^b = F_{i,p}^{b,new}$

update global bests $g_i^{new} = \arg\min_{\{b_{i,p}\}} \left\{F_{i,p}^b\right\}$

$g_i = g_i^{new}$

perform (12)

end for

return g_i

Remark 4. *Since at each iteration the global bests of each swarm are likely to change, the cost function value associated to the personal best $b_{i,p}$ context vector may provide a* false *comparison reference for the corresponding particle $x_{i,p}$ context vector evaluation because they may refer to different values of global bests. This is why we propose a first algorithm modification by requiring to evaluate the personal best at each algorithm step, by composing the*

context vectors associated to them too. This is performed by Steps 6*, and* 8 *of Algorithm 1.*

4.2. Motivating Example

It is well known that the key point affecting Algorithm 1 convergence is the problem decomposition step. By recalling the definition of *separable* function:

Definition 1. *$F(x_1, \cdots, x_N)$ is separable if and only if*

$$\arg \min_{x_1,\ldots,x_N} F(x_1, \ldots, x_N) = \left\{ \arg\min_{x_1} F(x_1, \ldots), \cdots, \arg\min_{x_N} F(\ldots, x_N) \right\}$$

then, in order to assure proper convergence of the algorithm, the cost function F should be separable in the given decomposition. In particular the given subcomponents affect the convergence properties as well as the dynamic behavior of the algorithm because they define the so-called *best-response curves*, [35]. These are defined as

$$\begin{aligned} bestResponseX_i(x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_N) \triangleq \\ \arg\min_{x_i} F(\cdots, x_i, \cdots), \quad i = 1, \cdots, N \end{aligned} \tag{13}$$

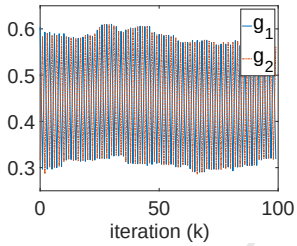
which clearly depend on the given problem decomposition. Motivated by the following example, we provide a new dynamic behavior analysis as well as the synthesis of a *modified* CC-PSO algorithm with parallel update timing, which enhances the algorithm convergence when the problem decomposition does not fit the problem separability.

Let us consider the cost function $F(x_1, x_2) = (x_1 - x_2)^2$ to be minimized, where $x_1, x_2 \in \mathbb{R}$. Note that $F(x_1, x_2)$ is not separable. However let us consider the decomposition given by the two components of the optimization variable, x_1, and x_2. Using (13), as in [35], we analyze the deterministic system associated to the CC-PSO algorithm, which, for this example, is given by

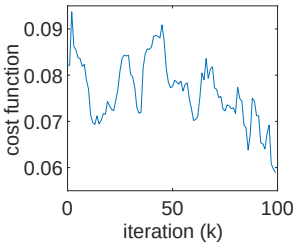
$$\begin{cases} x_1(k+1) = bestResponseX_1(x_2(k)) = x_2(k) \\ x_2(k+1) = bestResponseX_2(x_1(k)) = x_1(k) \end{cases} \tag{14}$$

Such system reproduces the parallel update timing of the CC-PSO algorithm as if the swarms associated to the two components were able to find

the global best at each time step. Even though $F(x_1, x_2)$ is a simple convex function, having its minima in $\Omega_{ex} \triangleq \{(x_1, x_2) \in \mathbb{R}^2 : x_1 = x_2\}$, system (14) shows an *oscillatory* behavior, given by the fact of having its eigenvalues in -1, and 1. Thus, it does not converge to Ω_{ex} for any initial condition $\{(x_1(0), x_2(0)) \in \mathbb{R}^2 : x_1(0) \neq x_2(0)\}$. This fact has an important implication on the dynamics of the CC-PSO algorithm. Although system (14) does not exactly reproduce Algorithm 1 behavior, this is likely to have similar oscillatory dynamics as the number of particles of each swarm increases. Fig. 6a shows the swarms global best trajectories during the run of CC-PSO algorithm for such example, where we set $N_p = 100$. Inspired by classic results



(a) Global bests trajectories.



(b) Cost function F evaluated in the global bests trajectories.

Figure 6: CC-PSO with parallel update timing dynamic behavior.

in control theory, where typically an oscillatory system can be stabilized by introducing *additional damping*, we modify system (14) equations, according to

$$\begin{cases} x_1(k+1) = x_1(k) + \beta\left(bestResponseX_1(x_2(k)) - x_1(k)\right) \\ \qquad\qquad = x_1(k) + \beta(x_2(k) - x_1(k)) \\ x_2(k+1) = x_2(k) + \beta(bestResponseX_2(x_1(k)) - x_2(k)) \\ \qquad\qquad = x_2(k) + \beta(x_1(k) - x_2(k)) \end{cases} \tag{15}$$

Such system converges to Ω_{ex} for any initial condition, and $\beta \in]0, 1[$. Indeed, for the considered values of β, system (15) has one eigenvalue inside the unit circle, and one eigenvalue in 1 with corresponding eigenvector equal to $\mathbf{1}_2 \triangleq [1 \quad 1]^\top$. Thus, system (15) trajectories converge to the subspace given by $span(\mathbf{1}_2)$, i.e. $\{(x_1, x_2) \in \mathbb{R}^2 : x_1 = x_2\}$, which is exactly Ω_{ex}. This motivates the CC-PSO algorithm modification shown in next subsection.

Remark 5. *As previously mentioned, the original CC-PSO is conceived for a* sequential *update timing parameter. This guarantees F to be* strictly non-increasing *in the* best *context vector trajectory, [32]. Such property is no longer satisfied in the case of parallel update timing. For instance, Fig. 6b shows the cost function of the aforementioned example evaluated in the best context vector trajectory, i.e. the one show in Fig. 6a.*

4.3. Damped CC-PSO Algorithm

The given example allows us to deduce an important heuristic. The main idea is to add the damping factor β in Algorithm 1 in order to reproduce a similar behavior to the one seen in the example of the deterministic system (15). In particular, we propose to replace Step 11 of Algorithm 1 with

$$g_i^{new} = g_i + \beta \left(\arg\min_{\{b_{i,p}\}} \left\{ F_{i,p}^b \right\} - g_i \right) \tag{16}$$

This has the effect of damping the update of the global bests, and possibly reducing unwanted oscillations. When applied in the algorithm, it is difficult to compute β using mathematical tools. Thus, it is left as an additional parameter to be tuned by selecting a value in $]0, 1]$. Generally, such value should be chosen to reduce possible oscillations while letting proper convergence of the algorithm, which could be unnecessarily worsened by excessive damping. We name the modified algorithm as the *damped CC-PSO with parallel update timing*. Let us now apply this algorithm to the example of the previous subsection for different values of β. The results are shown in Fig. 7. We remark how the damping reduces the oscillations seen in Fig. 6a. Moreover, as expected, the convergence time increases as the damping increases, i.e. as $\beta \to 0$. Notice also that even though the introduction of a damping factor does not in general guarantee the cost function to be strictly non-increasing in the global best trajectories, it helps mitigating the effect mentioned in Remark 5. This fact is shown for the above example in Fig. 8 for two different choices of β value.

5. Distributed Wind Farm Optimization

5.1. Unconstrained WF Distributed Optimization Problem

We first tackle the WF distributed optimization problem in its *unconstrained* formulation. In other words the following results are valid whenever

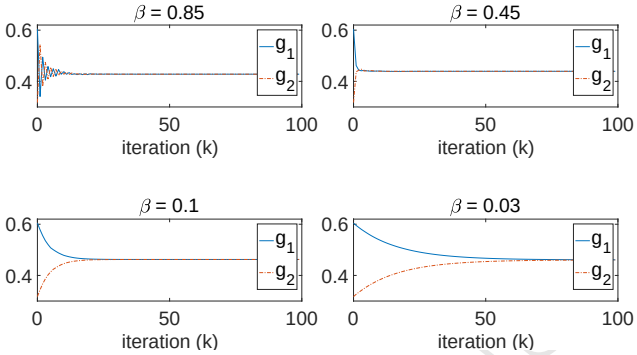


Figure 7: Global bests trajectories for different values of damping factor β.

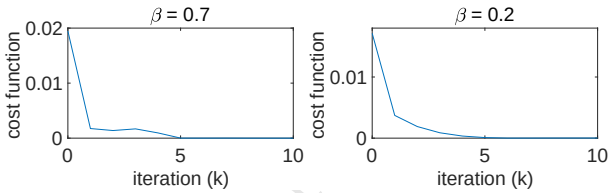


Figure 8: Cost function during the run of damped CC-PSO. The introduction of a damping factor helps giving a global decreasing pace to the cost function.

the free stream wind and P_{wf}^{max} are such that the constraints in (10) are not active. By recalling the definition of physical neighborhood and the associated variables introduced in Section 2, each WT i is endowed with a private optimization variable, i.e. its own axial induction factor α_i, and cost function, i.e. $-P_i(\alpha_i, \boldsymbol{\alpha_{ij}})$, where we removed the explicit dependence upon the free stream wind parameters (v_∞, θ^W) for ease of reading. The overall optimization problem is

$$\min_{\boldsymbol{\alpha}} -P_{wf}(\boldsymbol{\alpha}) = \min_{\{\alpha_i, i=1,\dots,N\}} -\sum_{i=1}^{N} P_i(\alpha_i, \boldsymbol{\alpha_{ij}}) \tag{17}$$

Thus, the WTs have to *cooperatively* minimize a *common* cost function in a distributed way, while *sharing* the common optimization variable $\boldsymbol{\alpha}$. Notice that (17) is not separable in the given problem physical decomposition $(\alpha_1, \dots, \alpha_N)$ as it does not match the separability condition of Definition 1. If this was met then it would imply that each WT i could optimize its own α_i independently from the other WTs choice.

To the authors' knowledge, in the literature, there does not exist DPSO algorithms readily applicable to the mentioned optimization problem. These usually combine PSO, and consensus techniques. In [37] each agent has knowledge of its own cost function that depends on only its own optimization variable, i.e. they do not share a common variable. Coupling among the agents is then given by a common objective function known by them all. A modified consensus technique is employed to estimate the sum of local cost functions at each step of PSO. Unfortunately such estimation may fail to be sufficiently accurate to guarantee proper convergence of the algorithm when the agents do share a common variable, [38]. Authors of [39] propose a distributed primal-dual optimization method, where the primal variable update, usually provided by sub-gradient methods, is replaced by the PSO algorithm. As in [37], agents do not share common variables, and the only coupling among them is given by the requirement of satisfying a common inequality constraint obtained by the sum of local private constraint functions. Examples of DPSO in which the agents share a common optimization variable can be found for instance in [40, 41]. However, they are both specific to the problem they address, and they are not readily extendable to the WF optimization problem.
In this section we aim at providing a DPSO algorithm for the WF optimization problem which is founded on the previously introduced damped CC-PSO, and it exploits the problem structure to distribute it among the WTs.

Remark 6. *It is important to stress that, in the literature, DPSO, and distributed evolutionary algorithms more in general, are often related to* parallel *computation in order to speed up the convergence to a solution. In this case the originally centralized optimization problem is split among several computing units. The available works are then mainly concerned with finding smart ways to define a problem* decomposition *which can resemble the problem separability structure as much as possible. In other words, how the algorithm is distributed among the agents is a* choice*, [42]. In the WF problem, each WT i optimization variable α_i acts as a subcomponent of problem (17) optimization variable $\boldsymbol{\alpha}$. Reflecting the physics of the problem, such decomposition is* imposed*, and it cannot be altered to perform particular CC algorithms.*

5.2. Exploiting the Problem Structure

To distribute CC-PSO the idea is that each WT i only needs to evaluate its contribution, i.e. its context vectors, in the WTs private cost functions

in which its own private control variable appears. In other words, WT i has to evaluate its context vectors in all P_j such that $j : i \in \mathcal{N}_j^p$. This is due to the fact that for $i = 1, \ldots, N$

$$\begin{aligned} &\arg \min_{\left\{\boldsymbol{x}_{i,p}^g, p=1,\ldots,N_p\right\}} \left\{-P_{wf}(\boldsymbol{x}_{i,p}^g), p = 1, \ldots, N_p\right\} = \\ &\arg \min_{\left\{\boldsymbol{x}_{i,p}^g p=1,\ldots,N_p\right\}} \left\{ - \sum_{\{j:i\in\mathcal{N}_j^p\}\cup\{i\}} P_j(\boldsymbol{x}_{i,p}^g), p = 1, \ldots, N_p \right\} \end{aligned} \tag{18}$$

where we remind that $\boldsymbol{x}_{i,p}^g = (g_1, \ldots, x_{i,p}, \ldots, g_N)$ is the context vector associated to the particle $x_{i,p}$, defined in Step 5 of Algorithm 1. Clearly the same conclusions hold true for the personal best context vectors. It is important to stress that (18) is made possible thanks to the use of the context vector strategy. Moreover, notice that, being applied to the WF optimization problem, $x_{i,p}$ represents a value of WT i axial induction factor α_i. Relationship (18) is precisely what lets a reduced computational burden since, for $i = 1, \ldots, N$, $\{j : i \in \mathcal{N}_j^p\} \cup \{i\} \subseteq \mathcal{V}$. Moreover, it is naturally defined upon the physical relationships among the WTs. This can be formalized by defining a graph $\mathcal{G}_p$ that keeps track of the wake couplings among the WTs. Thus $\mathcal{G}_p \triangleq (\mathcal{V}, \mathcal{E}_p)$ where $\mathcal{E}_p \subseteq \mathcal{V} \times \mathcal{V}$, the set of edges, is such that $(i, j) \in \mathcal{E}_p$ if WT $i \in \mathcal{N}_j^p$, i.e. if WT i is an upstream turbine with respect to WT j. Edge (i, j) is graphically indicated as an arrow $i \to j$. $\mathcal{G}_p$ generally defines a digraph. Since every WT power function is only known locally, each WT i that has to evaluate its context vectors in P_j, $j \neq i$, it firstly sends them to WT j, which then sends P_j valued in these context vectors back to WT i. Thus the required *communication* graph can be defined via the following

Assumption 2. *It exists a communication graph $\mathcal{G}_c = (\mathcal{V}, \mathcal{E}_c)$, where $\mathcal{E}_c \subseteq \mathcal{V} \times \mathcal{V}$ is such that if $(i, j) \in \mathcal{E}_p$, then both (i, j) and $(j, i) \in \mathcal{E}_c$.*

$\mathcal{G}_c$ defines an *undirected* graph, whose connections depend on the WTs physical neighborhoods. An example of $\mathcal{G}_p$ and $\mathcal{G}_c$ is shown in Fig. 9.

5.3. Role of Local Bests

Instead of letting each particle compare its personal best with the ones of all the other N_p-1 particles for the global best computation, usually a smaller subset is considered. This practice is well-known to reduce the possibility of *premature convergence*, [43]. In particular, each particle $x_{i,p}$ has access to the

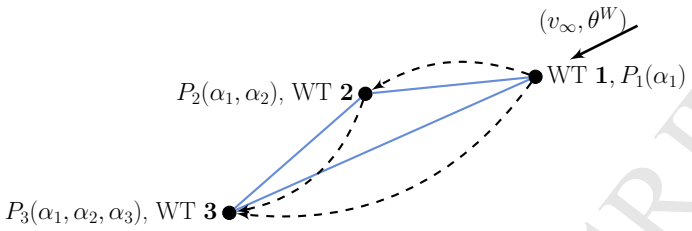


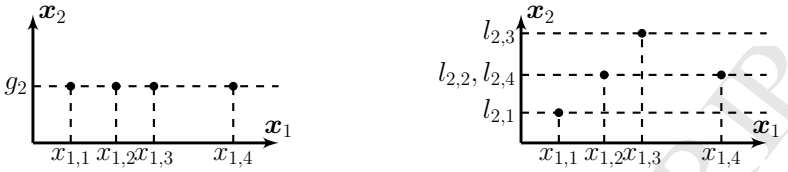
Figure 9: Physical graph $\mathcal{G}_p$ (*dash line*), and communication graph $\mathcal{G}_c$ (*solid line*). WT 1 evaluates its context vectors in $-P_1 - P_2 - P_3$, WT 2 in $-P_2 - P_3$, and WT 3 in $-P_3$.

personal bests of the particles belonging to a defined subset $S_{i,p}$. Thus, $x_{i,p}$ has *its own* knowledge of global best, which is *local* because it is restricted to the mentioned subset. This is why in this case the global best is called *local best*, and we indicate it with $l_{i,p}$. PSO equations (12), and the damped global best update (16) are then modified by simply replacing g_i with $l_{i,p}$. We additionally require the subset $S_{i,p}$ to be the same for each particle among the agents having same p index. Thus we can drop its i index: $S_{i,p} = S_p$, $\forall i \in \mathcal{V}$. In the context considered in this paper, the local best strategy has an important additional role in the convergence of the algorithm. CC-PSO with sequential update timing is known to suffer from a stagnation problem that is caused by the restriction that only one swarm is updated at a time, [32]. The parallel update timing version can present a similar problem. This is mainly due to the fact that each swarm puts together its context vectors by employing the *single best collaboration* technique, i.e. using the other swarm *global* bests. Thus the particles, and the personal bests, are restricted to *one* hyperplane of the search space, and this limits its exploration. The algorithm can be remarkably improved by employing the local bests when composing the context vectors, i.e.

$$\boldsymbol{x}^l_{i,p} \triangleq (l_{1,p}, \cdots, x_{i,p}, \cdots, l_{N,p})$$
$$\boldsymbol{b}^l_{i,p} \triangleq (l_{1,p}, \cdots, b_{i,p}, \cdots, l_{N,p})$$

where we renamed $\boldsymbol{x}^l_{i,p}$, and $\boldsymbol{b}^l_{i,p}$, respectively the context vector, and the personal best context vector associated to the particle $x_{i,p}$. This strategy is depicted in the example of Fig. 10, where we show a search space of dimension 2, $N_p = 4$ particles, and the corresponding context vectors for the swarm associated to the x_1 component.

In this paper we make use of the *singly-linked ring* structure for S_p, described



(a) Single best strategy. (b) Local best strategy.

Figure 10: Local best based context vectors (*dots*) for the x_1 component.

in [43], where we can select the number N_v of particles belonging to each subset S_p. In order to show the improvement brought by the described strategy, let us provide the following example. Consider the cost function

$$F(x_1, x_2) = 3(1 - x_1)^2 e^{-x_1^2 - (x_2+1)^2} + \\ - 10(x_1/5 - x_1^3 - x_2^5) e^{-x_1^2 - x_2^2} - 1/3 e^{-(x_1+1)^2 - x_2^2} \quad (19)$$

In the region defined by $[-2.5, 2.5] \times [-2.5, 2.5]$, among its minima, function (19) has a global minimum in $A = (0.2282; -1.626)$ where its value is -6.5511, and a higher local minimum in $B = (-1.347; 0.2045)$ where its value is -3.0498. We run 1000 times the damped CC-PSO algorithm with parallel update timing for both the case of single best, and local best strategy. As far as the latter is concerned we select $N_v = 2$, while for both strategies we choose $N_p = 40$, and $\beta = 0.4$. While for the local best strategy the algorithm converges to the global minimum for the totality of the trials, the single best one attains A the 70% of them while for the remaining 30% it converges to B. This fact is representative of the benefit gained by using the local bests to compose the context vectors.

5.4. Wake Model Approximation for Reduced Communication

Since according to the considered wake model each upstream WT influences *every* downstream one, and because of communication Assumption 2, it would be generally required to implement a *complete* communication graph, i.e. there should be a direct communication between any two WTs in a WF. However, from a practical point of view, the physical graph $\mathcal{G}_p$, and the according $\mathcal{G}_c$, can be highly reduced because the wake interaction between two enough distant WTs is negligible, especially if they are not aligned in the wind direction. The idea is thus to reduce the WTs physical neighborhoods, leading to an *approximated* wake interaction model, which would in turns

allow the proposed DPSO to run on a reduced communication graph. In this work we propose to define the WT i communication neighborhood by acting on two newly introduced parameters, namely $\psi_{d,i}$, and $\psi_{t,i}$. These define respectively a distance in the wind direction, and a distance in the cross direction of the wind. Such parameters are in fact function of θ^W: $\psi_{d,i}(\theta^W)$, $\psi_{t,i}(\theta^W)$. The neighborhood of WT i is obtained as shown in Fig. 11. For a

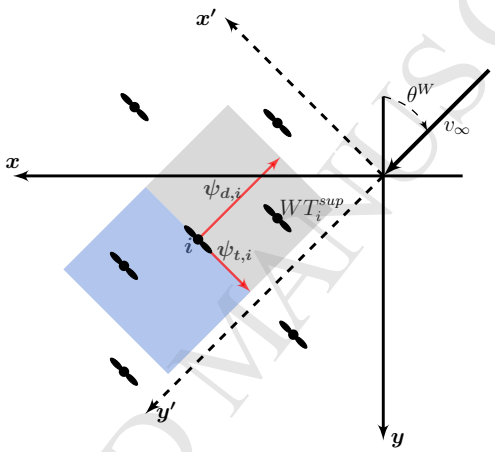


Figure 11: Rectangular neighborhood via $\psi_{d,i}$ and $\psi_{t,i}$: WT_i^{up} in light grey area, WT_i^{down} in light blue area.

given wind direction θ^W, by selecting a value for the aforementioned parameters, a *rectangle* centered in WT i is obtained. Then, all the WTs laying within it are part of the communication neighborhood of WT i. The rectangular form is justified to allow flexibility to the degree of wake interaction approximation. Typically $\psi_{d,i} > \psi_{t,i}$, as the wake coupling of WTs aligned in the wind direction is stronger. The neighborhood defined via the mentioned parameters can be further divided in the upstream turbines subset, which we name WT_i^{up}, i.e. WT i physical neighbors, and the downstream turbines subset, which we name WT_i^{down}, i.e. those turbines for which WT i is a physical neighbor. WT_i^{sup}, defined as the most upstream WT in the approximated neighborhood of WT i, belongs to WT_i^{up}. Values for $\psi_{d,i}$, and $\psi_{t,i}$ for each wind direction of interest can be found in such a way that they guarantee a satisfactory solution to the optimization problem. This can be done off-line and via simulation. The obtained functions $\psi_{d,i}(\theta^W)$, $\psi_{t,i}(\theta^W)$ can be thus stored in the corresponding WT i, for instance, via a lookup table. Based on this knowledge and on the following assumptions, each WT

can compute its own neighborhood.

Assumption 3. *Each WT has knowledge of the number N of WTs in the wind farm, and their position (x_i, y_i), $i = 1, \ldots, N$ with respect to a given reference frame.*

Assumption 4. *Each WT measures (v_i, θ_i), i.e. the speed and direction of the wind blowing in the front of their rotor plane.*

Because of Assumption 1, then $\theta_i \simeq \theta^W$, $\forall i \in \mathcal{V}$. Thus, given θ^W, the subroutine that the generic WT i has to perform to compute its neighborhood is shown in Algorithm 2, where (x'_i, y'_i) values are computed via (3).

Algorithm 2 Wind Turbine i Neighborhood Computation

Input: (x'_i, y'_i), $i = 1, \ldots, N$
Output: WT_i^{sup}, WT_i^{up}, WT_i^{down}

$\texttt{distance} = y'_i$
$WT_i^{sup} = \{\}$, $WT_i^{up} = \{\}$, $WT_i^{down} = \{\}$
LOOP Process
for ($j = 1$ to N) **and** ($j \neq i$) **do**
  if ($|x'_i - x'_j| < \psi_{t,i}$) **and** ($|y'_i - y'_j| < \psi_{d,i}$) **then**
    Establish communication between WT i and WT j
    if $y'_j < y'_i$ **then**
      Add WT j to WT_i^{up}
    else
      Add WT j to WT_i^{down}
    end if
    if $y'_j < \texttt{distance}$ **then**
      $WT_i^{sup} = j$
      $\texttt{distance} = y'_j$
    end if
  end if
end for
return WT_i^{sup}, WT_i^{up}, WT_i^{down}

5.5. Fundamental Wind Farm DPSO algorithm

We are now ready to provide the proposed DPSO algorithm to solve (17). This is shown in Algorithm 3, which is written for the generic WT i.

Algorithm 3 DPSO for the Wind Farm Optimization Problem

Output: Local bests: $l_{i,p}$, $p = 1, \ldots, N_p$

Initialization :

Randomly initialize $x_{i,p} \in [\underline{x}, \bar{x}]$, $s_{i,p} \in [-\bar{s}, \bar{s}]$, $p = 1, \ldots, N_p$

$b_{i,p} = x_{i,p}$, $p = 1, \ldots, N_p$

Randomly initialize $l_{i,p} \in [\underline{x}, \bar{x}]$, $p = 1, \ldots, N_p$

Initialize $v_{i,p} = v_i$, $p = 1, \ldots, N_p$

LOOP Process

for $k = 1$ to `max_iter` **do**

  Send $x_{i,p}$, $b_{i,p}$, $l_{i,p}$, $v_{i,p}$ $p = 1, \ldots, N_p$ *to* WT $k \in WT_i^{down}$, via $\mathcal{G}_c$

  Wait to receive $x_{j,p}$, $b_{j,p}$, $l_{j,p}$, $v_{j,p}$ $p = 1, \ldots, N_p$ *from* WT $j \in WT_i^{up}$, via $\mathcal{G}_c$

  Compose the context vectors associated to the its own particles $x_{i,p}$: $\boldsymbol{x}_{i,p}^l \triangleq (x_{i,p}, l_{j,p} : j \in WT_i^{up})$, $p = 1, \ldots, N_p$

  Compose the context vectors associated to its own personal bests $b_{i,p}$: $\boldsymbol{b}_{i,p}^l \triangleq (b_{i,p}, l_{j,p} : j \in WT_i^{up})$, $p = 1, \ldots, N_p$

  Compose the context vectors associated to the particles $x_{j,p}$ of its neighbors $j \in WT_i^{up}$: $\boldsymbol{x}_{j,p}^l \triangleq (x_{j,p}, l_{i,p}, l_{k,p} : k \in WT_i^{up} \wedge k \neq j)$, $p = 1, \ldots, N_p$

  Compose the context vectors associated to the personal bests $b_{j,p}$ of its neighbors $j \in WT_i^{up}$: $\boldsymbol{b}_{j,p}^l \triangleq (b_{j,p}, l_{i,p}, l_{k,p} : k \in WT_i^{up} \wedge k \neq j)$, $p = 1, \ldots, N_p$

  Among the received $v_{j,p}$, select $v_{k,p}$ where $k = WT_i^{sup}$, for $p = 1, \ldots, N_p$

  Compute $v_{i,p}$ according to the wake model, using $v_{k,p}$ as upstream wind value, and $\boldsymbol{x}_{i,p}^l$ as operating points, for $p = 1, \ldots, N_p$

  Evaluate particle context vectors $\boldsymbol{x}_{i,p}^l$ in P_i: $P_{i,p}^x \triangleq P_i(\boldsymbol{x}_{i,p}^l)$, $p = 1, \ldots, N_p$

  Evaluate personal best context vectors $\boldsymbol{b}_{i,p}^l$ in P_i: $P_{i,p}^b \triangleq P_i(\boldsymbol{b}_{i,p}^l)$, $p = 1, \ldots, N_p$

  Evaluate particle context vectors $\boldsymbol{x}_{j,p}^l$ $\forall j \in WT_i^{up}$ in P_i: $P_{i,p}^{x,j} \triangleq P_i(\boldsymbol{x}_{j,p}^l)$, $p = 1, \ldots, N_p$

  Evaluate personal best context vectors $\boldsymbol{b}_{j,p}^l$ $\forall j \in WT_i^{up}$ in P_i: $P_{i,p}^{b,j} \triangleq P_i(\boldsymbol{b}_{j,p}^l)$, $p = 1, \ldots, N_p$

  Send $P_{i,p}^{x,j}$, $P_{i,p}^{b,j}$, $p = 1, \ldots, N_p$ *to* the corresponding WT $j \in WT_i^{up}$, via $\mathcal{G}_c$

  Wait to receive $P_{k,p}^{x,i}$, $P_{k,p}^{b,i}$, $p = 1, \ldots, N_p$ *from* the corresponding WT $k \in WT_i^{down}$, via $\mathcal{G}_c$

  Compute *fitness function* values $F_{i,p}^x$, $p = 1, \ldots, N_p$

  Compute *fitness function* values $F_{i,p}^b$, $p = 1, \ldots, N_p$

  Update personal bests for $p = 1, \ldots, N_p$

$$\left(b_{i,p}^{new}, F_{i,p}^{b,new}\right) = \begin{cases} \left(x_{i,p}, F_{i,p}^x\right) & \text{if } F_{i,p}^x < F_{i,p}^b \\ \left(b_{i,p}, F_{i,p}^b\right) & \text{otherwise} \end{cases}$$

  $b_{i,p} = b_{i,p}^{new}$; $F_{i,p}^b = F_{i,p}^{b,new}$

  Update local bests for $p = 1, \ldots, N_p$

$$l_{i,p}^{new} = l_{i,p} - \beta \left(\arg \min_{\{b_{i,p} \in S_p\}} \left\{ F_{i,p}^b \right\} - l_{i,p} \right)$$

  $l_{i,p} = l_{i,p}^{new}$, $p = 1, \ldots, N_p$

  Perform PSO update (12), with box constraints handled via (20), (21)

end for

return $l_{i,p}$, $p = 1, \ldots, N_p$

First of all notice that the box constraints on the axial induction factor are handled via

$$s_{i,p}(k+1) \triangleq \max\{\min\{s_{i,p}(k+1), \bar{s}\}, -\bar{s}\} \tag{20}$$

$$x_{i,p}(k+1) \triangleq \max\{\min\{x_{i,p}(k+1), \bar{x}\}, \underline{x}\} \tag{21}$$

where $\underline{x} = 0$, $\bar{x} = 1/3$, $\bar{s} \triangleq 1/2(\bar{x} - \underline{x})$, and where (20), (21) are added respectively after the first and the second equation in (12).
Secondly, we introduce a wind speed variable that allows a proper power function computation. This is explained in the following. Recall that in order to compute its own power function P_i via (6), each WT i needs to compute the wind speed value $\bar{v}_i$. According to the original wake model of Section 2, this depends on the upstream WTs axial induction factors, and on v_∞, the free stream wind speed captured by *the most upstream* WTs in the WF. Because of the approximated neighborhood introduced in the previous subsection, each WT i now has its *own* most upstream WT, i.e. WT_i^{sup}. WT_i^{sup} wind speed is thus the value that WT i uses to compute its own power function. Since WT_i^{sup} wind speed value $\forall i \in \mathcal{V}$ is in turns function of the upstream WT axial induction factors, it changes during the run of the algorithm, and it needs to be updated at each iteration step. The above considerations are implemented in Algorithm 3 by associating a wind speed value $v_{i,p}$ to each particle in WT i. These values are initialized in the WT i measured one, v_i. At each iteration, each WT i sends these values in Step 6 to WT_i^{down}, and it receives the wind speed variables $v_{j,p}$ from WTs j in its physical neighborhood WT_i^{up}, for $p = 1, \ldots, N_p$, in Step 7. Then, in order to update its own wind speed values $v_{i,p}$, $p = 1, \ldots, N_p$, in Step 12, WT i selects the wind values $v_{k,p}$, $p = 1, \ldots, N_p$, belonging to the most upstream WT in its neighborhood, i.e. $k = WT_i^{sup}$. In Step 13, $v_{i,p}$ $p = 1, \ldots, N_p$ are updated.
Once the power functions have been evaluated in the required context vectors, and sent back to the according WTs, each WT i can compute the *fitness function* values associated to its particles context vectors, $F_{i,p}^x$, and to its personal bests context vectors, $F_{i,p}^b$. This is done in Steps 17, and 18 respectively. The computation formula for $F_{i,p}^x$ is

$$F_{i,p}^x \triangleq -P_{i,p}^x - \sum_{k \in WT_i^{down}} P_{k,p}^{x,i} \tag{22}$$

Similar results hold for $F_{i,p}^b$. All in all, each iteration of the algorithm requires

each WT to exchange information with its neighbors in WT_i^{up} and WT_i^{down} twice. Moreover, at each iteration WT i evaluates $2N_p(|WT_i^{up}|+1)$ times its private cost function P_i, where $|WT_i^{up}|$ indicates the cardinality of the set WT_i^{up}.

5.6. Extension to the Constrained Case

Once set the basics to solve (17), it is fairly easy to extend them to treat the power constraints. In this case we add the following inequality constraints to (17)

$$P_i(\alpha_i, \boldsymbol{\alpha}_{ij}) \leq \min\left\{P_n, \frac{P_{wf}^{max}}{N}\right\} \quad i = 1, \ldots, N \tag{23}$$

First of all this implies that the maximum allowed WF power is equally distributed among the WTs. Other distribution strategies are beyond the scope of this paper and they will not be treated further. Secondly, it requires each WT to have knowledge of P_{wf}^{max}. This piece of information can be forwarded to the WTs of the WF via $\mathcal{G}_c$. Constraints (23) can be integrated in Algorithm 3 via Deb's rule. This technique exhibits some interesting properties which make it eligible for the application of plenty constrained optimization problems. Indeed, even if belonging to the penalty function approaches, it does not require any penalty parameter. Moreover, it allows avoiding any cost function distortion that may occur when incorporating the constraints in the problem via penalty functions. Deb's rule consists of a tournament selection in which, when comparing two solutions of (17), (23), the following criteria is adopted

a) Any feasible solution is preferred to any infeasible solution.
b) Among two feasible solutions, the one having better objective function value is preferred.
c) Among two infeasible solutions, the one having smaller constraint violation is preferred.

This can be easily done by replacing fitness function (22) with

$$\tilde{F}_{i,p}^x \triangleq -P_{i,p}^x - \sum_{k \in WT_i^{down}} P_{k,p}^{x,i}$$

$$G_{i,p}^x \triangleq P_{i,p}^x - \min\left\{P_n, \frac{P_{wf}^{max}}{N}\right\}$$

$$F_{i,p}^x \triangleq \begin{cases} \tilde{F}_{i,p}^x & \text{if } G_{i,p}^x < 0 \\ G_{i,p}^x & \text{otherwise} \end{cases}$$

6. Simulations

In order to test DPSO performance we provide simulations for both the inactive and active constraints case in the WF optimization problem. For all the simulations we consider WFs having the layout structure shown in Fig. 12, whose WTs have $R = 63\ m$, and nominal power $P_n = 5\ MW$. Moreover we consider parameters $\psi_{d,i}$, and $\psi_{t,i}$ to be the same for each WT.

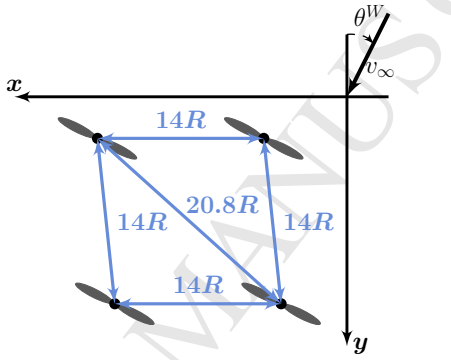


Figure 12: Horns Rev 1 wind farm layout.

We can thus drop index i. This choice simplifies the WT communication neighborhood computation and it can be generally applied for WFs having a regular layout structure. As far as the DPSO parameters are concerned we select $N_p = 10$ particles associated to each WT, a particle neighborhood dimension equal to $N_v = 2$, and a damping factor $\beta = 0.8$.

6.1. Unconstrained WF Optimization

The test is carried out for a free wind speed value $v_\infty = 7\ m/s$, and wind directions $\theta^W = 0°,\ 90°$. Moreover to test the algorithm scalability we consider two WFs having $N = 81$, and $N = 196$ WTs respectively, and same number of WTs along the x and y direction. For both WFs and wind directions, parameters $\psi_d(\theta^W)$, and $\psi_t(\theta^W)$ are such that the minimum number of direct communications required by at least one WT in the farm, i.e. $\underline{C} \triangleq \min_{i=1,\dots,N} \left\{|WT_i^{up}| + |WT_i^{down}|\right\}$, is $\underline{C} = 2$, and its maximum value, i.e. $\bar{C} \triangleq \max_{i=1,\dots,N} \left\{|WT_i^{up}| + |WT_i^{down}|\right\}$, is $\bar{C} = 4$. Notice also that the above parameters values directly influence the algorithm required computational time as they set the physical neighborhood dimension $|WT_i^{up}|$ for each WT i. Since DPSO is a synchronous algorithm, the bottleneck concerning

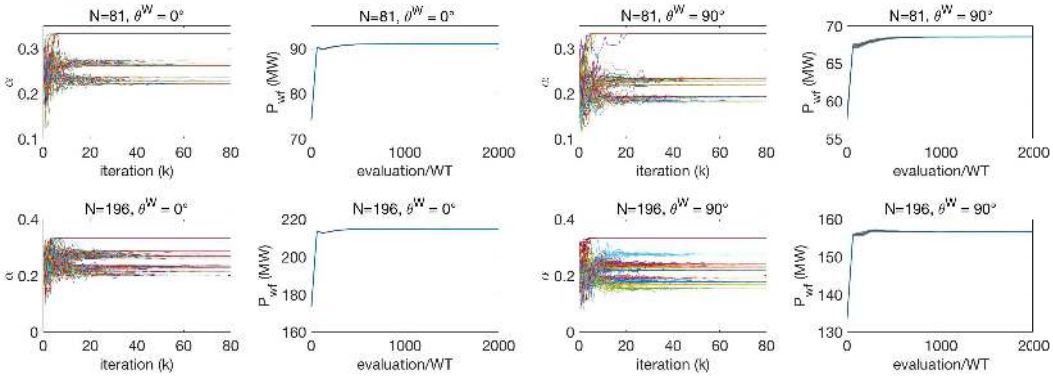


Figure 13: Simulations for the unconstrained case, i.e. $v_\infty = 7\ m/s$, and no P_{wf}^{max} constraint. For each couple of chosen WF size N and wind direction θ^W, we provide *a)* an example of the global best vector of α values trajectory, converging to a solution, against the number of iterations, *b)* the mean value of the wind farm power function $\bar{P}_{wf}$, in blue solid line, and the interval defined by the standard deviation σ_P, $\bar{P}_{wf} \pm \sigma_P$, shown in the grey area, against the number of cost function evaluations per WT, and where $\bar{P}_{wf}$ and σ_P are computed out of 20 DPSO algorithm trials.

its computational time is represented by those WTs having highest $|WT_i^{up}|$ value. First of all this is due to the fact that $|WT_i^{up}|$ influences the complexity of WT i power function P_i. Basically WT i has to compute the wake effect caused by each WT belonging to WT_i^{up}, via (5). Secondly, recall that each WT i has to evaluate its power function $2N_p(|WT_i^{up}|+1)$ times per iteration. For the considered wake model, DPSO parameters, and communication parameters, the computational time required at each iteration is $C_T \simeq 0.1\ ms$[1]. DPSO convergence performance is illustrated in Fig. 13. Here, for each considered couple of WF size N and wind direction θ^W, we first show an example of the global best vector trajectory during the run of the DPSO algorithm. In particular, for each algorithm iteration we are able to see the N global best axial induction factors α, entries of the global best vector, found by DPSO. As expected, according to the wind direction, a certain number of α values converges to $\frac{1}{3}$, meaning that the corresponding WTs are operated in MPPT mode. These are the WTs that, because of their position with respect to the wind direction, do not cause any wind speed reduction to any downstream

[1]Computations are carried on an Intel®Core™i5, CPU @ 2.30 GHz, RAM of 8 GB, with MATLAB®R2017b.

WT. The remaining axial induction factors all converge to lower values than $\frac{1}{3}$, meaning that the corresponding WTs are required to reduced their own wind power extraction in order to maximize the WF one.
Secondly, in Fig. 13, we show the mean and standard deviation of the WF power, evaluated in the global best α values, respectively $\bar{P}_{wf}$ and σ_P, computed for each algorithm iteration, and out of 20 algorithm trials. In particular, $\bar{P}_{wf} \pm \sigma_P$ values are shown against the number of local power function evaluations required by the WTs with highest $|WT_i^{up}|$. From this, first of all we are able to conclude on the algorithm robustness, as the grey area representing $\bar{P}_{wf} \pm \sigma_P$ is almost indistinguishable from $\bar{P}_{wf}$, reported in blue solid line. This means that, despite the DPSO aleatory feature, the algorithm always converges to the same value and in almost the same way. Secondly, we are able to see that the algorithm reaches a stable solution in a relatively small number of iterations and according number of function evaluations per WT. Because of the aforementioned computational time value C_T, if the time delay due to the communication steps is small, then DPSO proves to be eligible for real-time operations, as typically the WT dynamics has a response time on the order of seconds. Finally, in Table 1 we report, for each considered case, the obtained WF power value P_{wf}, the power gain G with respect to the greedy WF operation, and the loss L, due to the wake model approximation, with respect to the case of complete communication.

Table 1: WF power gains when constraints are inactive.

$\boldsymbol{N}$	$\boldsymbol{\theta^W}$ (°)	$\boldsymbol{P_{wf}}$ (MW)	$\boldsymbol{G}$ (%)	$\boldsymbol{L}$ (%)
81	0	91.02	6.4	0.3
81	90	68.58	20.7	1.7
196	0	214.63	6.8	0.6
196	90	156.65	23.5	2.7

6.2. Constrained WF Optimization

For the active constraints case we first propose a similar simulation to the one shown in the previous subsection. In particular we consider the same WF examples and same wind directions, but we select a free wind stream speed value equal to 14 *m/s*, so that the WTs are likely to have their nominal power constraint active. This is the only power constraint considered in this first

simulation. Similarly to Fig. 13, in Fig. 14, for each case we report an example of global best axial induction factors convergence, and $\bar{P}_{wf} \pm \sigma_P$ computed out of 20 algorithm trials, where we also show the WF nominal power in black dash-dot line. Because of the chosen v_∞ value, the WF operates at its

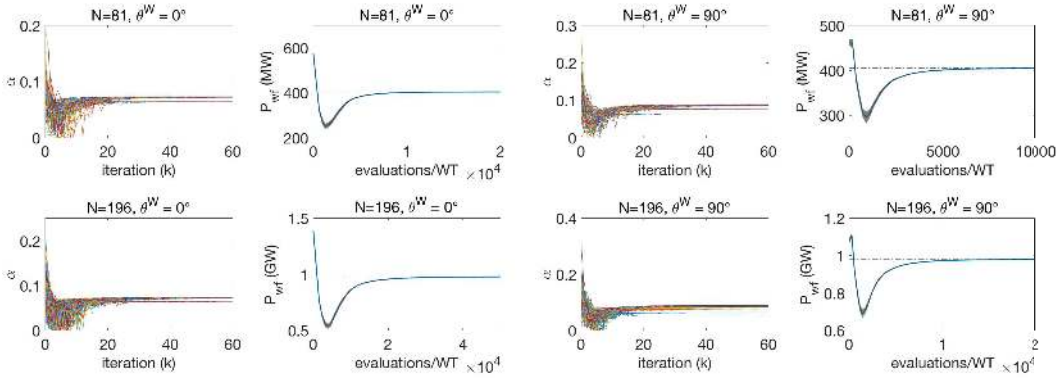


Figure 14: Simulations for the constrained case, i.e. $v_\infty = 14\ m/s$, and no P_{wf}^{max} constraint. For each couple of chosen WF size N and wind direction θ^W, we provide *a)* an example of the global best vector of α values trajectory, converging to a solution, against the number of iterations, *b)* the mean value of the wind farm power function $\bar{P}_{wf}$, in blue solid line, and the interval defined by the standard deviation σ_P, $\bar{P}_{wf} \pm \sigma_P$, shown in the grey area, against the number of cost function evaluations per WT, and where $\bar{P}_{wf}$ and σ_P are computed out of 20 DPSO algorithm trials. The black dash-dot line represents the WF nominal power.

nominal value for all the considered examples. As expected, this results in a choice of optimal axial induction factors having values lower than $\frac{1}{3}$, as every WT hits its nominal power constraint. Moreover this allows us to conclude on the good DPSO robustness in finding the optimal solution in the constrained case too. Indeed, for every considered case, the algorithm converges to a combination of axial induction factors yielding the WF nominal power, which is the maximum allowed value respecting the WF constraints. It could be argued that in this case one could simply saturate each WT at its nominal power value, without the need for performing any optimization. However, whether the WF operates at its nominal value or not is generally not known a priori on the only basis of (v_∞, θ^W) values. Such piece of information is issued from the optimization step itself. Moreover, the situation gets even more complex if one thinks about all the possible wind conditions combined with any possible P_{wf}^{max} value. In Table 2 we report the required minimum

and maximum number of direct communications among the WTs for each case, and the according C_T per iteration imposed by the WTs with highest $|WT_i^{up}|$. From this, as well as from the evaluations per WT shown in Fig. 14,

Table 2: WF required communication and C_T when constraints are active.

$\boldsymbol{N}$	$\boldsymbol{\theta^W}$ (°)	$\underline{\boldsymbol{C}}$	$\bar{\boldsymbol{C}}$	$\boldsymbol{C_T}$/iteration (ms)
81	0	8	15	~ 1.2
81	90	8	8	~ 0.4
196	0	18	40	~ 7.6
196	90	13	13	~ 1.2

it is clear that when constraints are active the convergence time is higher. However, since the required number of iterations to a solution is low, if we neglect the communication time delays, still the algorithm shows real-time performance. This is directly due to the need for a higher number of direct communications among the WTs, in turns due to the fact that in the constrained case the wake approximation has to be highly reduced to let DPSO find a feasible solution.
Eventually we show an example in which not all the constraints are active. For this last simulation we consider WF Horn Rev 1 layout, where there are 10 WTs along the x direction, and 8 along the y direction. We additionally take advantage of this case to give an example of grid power constraint, by setting P_{wf}^{max} equal to 63% of the WF nominal power. v_∞ and θ^W are set equal to 10 m/s and 45° respectively. In the above conditions, DPSO succeeds in finding the WF axial induction factors combination yielding $P_{wf} = P_{wf}^{max}$. In Fig. 15, we show the WTs whose constraints are active, i.e. operating at $P_{wf}^{max}/_N$.

7. Conclusion

We presented a novel approach allowing the WF optimization problem to be treated in the MAS framework. The considered optimization problem is the one of maximizing the WF power production by taking into account the wake interaction among the WTs as well as the system constraints. By combining the knowledge of the wake model and the use of a distributed architecture, we are able to achieve fast convergence to a solution of the optimization problem, which makes it eligible for real-time operations, such

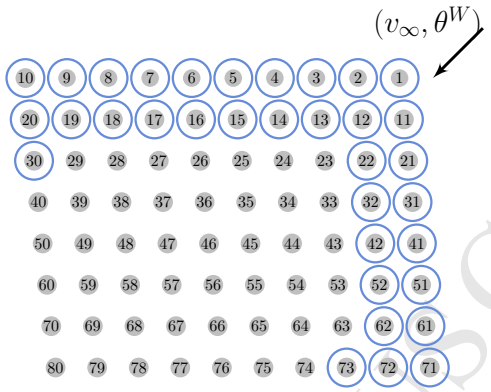


Figure 15: Horn Rev 1 wind farm for active P_{wf}^{max} constraint, $v_\infty = 10$ m/s, and $\theta^W = 45°$. The WTs with a blue circle are the one operating at P_{wf}^{max}/N.

as satisfying the power constraints while tracking the wind speed variability. The WF optimization is based on a novel distributed PSO algorithm. The choice of such metaheuristic technique is justified as it enables treating a nonconvex problem with nonlinear constraints. This in turns has its roots in the CC technique which we modified by mainly introducing a damping factor which enhances the algorithm convergence performance for those optimization problems, such as the WF one, whose distribution among the agents, reflecting the physics of the problem itself, does not match the problem separability structure. Distribution among the WTs is made possible by exploiting the knowledge of their physical inter-dependencies as well as an approximated wake model which lets the implementation of a reduced communication graph. Performance shown in simulation is satisfactory, as the algorithm proves to provide robust solutions in a relatively small number of iterations, even for the case of large WFs.

The main drawback of the proposed approach is the need for an increasing number of direct communications among the WTs with the growth of the aerodynamic couplings when constraints are active. In the near future it would be interesting to consider algorithm modifications allowing a further reduced communication in the constrained case. Eventually it is worth mentioning that the presented DPSO algorithm is poorly dependent on the employed wake model, since its main requirements are a model description based on the WTs yaw angle and axial induction factor, and the capability to evaluate its particles in the WF power function. This makes it applicable to a great variety of existing wake models in the literature. For instance, a sim-

ple model modification could be introduced by considering a wake expansion accounting for the wind turbulence as shown in [27].

Acknowledgment

This study has been carried out in the RISEGrid Institute (www.centralesupelec.fr/en/risegrid-institute-research-institute-smarter-electric-grids), joint program between CentraleSupélec and EDF ('Electricité de France') on smarter electric grids.

[1] J. Park, K. H. Law, Cooperative wind turbine control for maximizing wind farm power using sequential convex programming, Energy Conversion and Management 101 (2015) 295–316.

[2] S. Howell, Y. Rezgui, J.-L. Hippolyte, B. Jayan, H. Li, Towards the next generation of smart grids: Semantic and holonic multi-agent management of distributed energy resources, Renewable and Sustainable Energy Reviews 77 (2017) 193–214.

[3] M. W. Khan, J. Wang, The research on multi-agent system for microgrid control and optimization, Renewable and Sustainable Energy Reviews 80 (2017) 1399–1411.

[4] P. Fleming, P. Gebraad, J.-W. van Wingerden, S. Lee, M. Churchfield, A. Scholbrock, J. Michalakes, K. Johnson, P. Moriarty, Sowfa supercontroller: A high-fidelity tool for evaluating wind plant control approaches, Tech. rep., National Renewable Energy Laboratory (NREL), Golden, CO. (2013).

[5] V. Santhanagopalan, M. Rotea, G. Iungo, Performance optimization of a wind turbine column for different incoming wind turbulence, Renewable Energy 116 (2018) 232–243.

[6] J. Park, K. H. Law, Layout optimization for maximizing wind farm power production using sequential convex programming, Applied Energy 151 (2015) 320–334.

[7] J. Feng, W. Z. Shen, Solving the wind farm layout optimization problem using random search algorithm, Renewable Energy 78 (2015) 182–192.

[8] W. Li, E. Özcan, R. John, Multi-objective evolutionary algorithms and hyper-heuristics for wind farm layout optimisation, Renewable Energy 105 (2017) 473–482.

[9] H.-S. Huang, Distributed genetic algorithm for optimization of wind farm annual profits, in: Intelligent Systems Applications to Power Systems, 2007. ISAP 2007. International Conference on, IEEE, 2007, pp. 1–6.

[10] J. S. González, M. B. Payán, J. R. Santos, Á. G. G. Rodríguez, Maximizing the overall production of wind farms by setting the individual operating point of wind turbines, Renewable Energy 80 (2015) 219–229.

[11] P. Gebraad, F. Teeuwisse, J. Wingerden, P. A. Fleming, S. Ruben, J. Marden, L. Pao, Wind plant power optimization through yaw control using a parametric model for wake effects – a cfd simulation study, Wind Energy 19 (1) (2016) 95–114.

[12] J. Park, K. H. Law, A data-driven, cooperative wind farm control to maximize the total power production, Applied Energy 165 (2016) 151–165.

[13] J. Lee, E. Son, B. Hwang, S. Lee, Blade pitch angle control for aerodynamic performance optimization of a wind farm, Renewable energy 54 (2013) 124–130.

[14] U. Ciri, M. A. Rotea, S. Leonardi, Model-free control of wind farms: A comparative study between individual and coordinated extremum seeking, Renewable Energy 113 (2017) 1033–1045.

[15] P. A. Fleming, P. M. Gebraad, S. Lee, J.-W. van Wingerden, K. Johnson, M. Churchfield, J. Michalakes, P. Spalart, P. Moriarty, Evaluating techniques for redirecting turbine wakes using sowfa, Renewable Energy 70 (2014) 211–218.

[16] L. Wang, A. Tan, Y. Gu, A novel control strategy approach to optimally design a wind farm layout, Renewable Energy 95 (2016) 10–21.

[17] F. Heer, P. M. Esfahani, M. Kamgarpour, J. Lygeros, Model based power optimisation of wind farms, in: Control Conference (ECC), 2014 European, IEEE, 2014, pp. 1145–1150.

[18] J. Herp, U. V. Poulsen, M. Greiner, Wind farm power optimization including flow variability, Renewable Energy 81 (2015) 173–181.

[19] J. Tian, C. Su, M. Soltani, Z. Chen, Active power dispatch method for a wind farm central controller considering wake effect, in: Industrial Electronics Society, IECON 2014-40th Annual Conference of the IEEE, IEEE, 2014, pp. 5450–5456.

[20] P. M. Gebraad, F. C. van Dam, J.-W. van Wingerden, A model-free distributed approach for wind plant control, in: American Control Conference (ACC), 2013, IEEE, 2013, pp. 628–633.

[21] J. Barreiro-Gomez, C. Ocampo-Martinez, F. Bianchi, N. Quijano, Model-free control for wind farms using a gradient estimation-based algorithm, in: Control Conference (ECC), 2015 European, IEEE, 2015, pp. 1516–1521.

[22] J. R. Marden, S. D. Ruben, L. Y. Pao, A model-free approach to wind farm control using game theoretic methods, IEEE Transactions on Control Systems Technology 21 (4) (2013) 1207–1214.

[23] S. Behera, S. Sahoo, B. Pati, A review on optimization algorithms and application to wind energy integration to grid, Renewable and Sustainable Energy Reviews 48 (2015) 214–227.

[24] K. Deb, An efficient constraint handling method for genetic algorithms, Computer methods in applied mechanics and engineering 186 (2) (2000) 311–338.

[25] N. O. Jensen, A note on wind generator interaction, 1983.

[26] I. Katic, J. Højstrup, N. O. Jensen, A simple model for cluster efficiency, in: European wind energy association conference and exhibition, 1986, pp. 407–410.

[27] L. Tian, W. Zhu, W. Shen, Y. Song, N. Zhao, Prediction of multi-wake problems using an improved jensen wake model, Renewable Energy 102 (2017) 457–469.

[28] Á. Jiménez, A. Crespo, E. Migoya, Application of a les technique to characterize the wake deflection of a wind turbine in yaw, Wind energy 13 (6) (2010) 559–572.

[29] F. Ebrahimi, A. Khayatiyan, E. Farjah, A novel optimizing power control strategy for centralized wind farm control system, Renewable Energy 86 (2016) 399–408.

[30] G. Delille, G. Malarange, C. Gaudin, Analysis of the options to reduce the integration costs of renewable generation in the distribution networks. part 2: A step towards advanced connection studies taking into account the alternatives to grid reinforcement, 22nd International Conference and Exhibition on Electricity Distribution (CIRED) (2013) 1356.

[31] M. A. Potter, K. A. De Jong, A cooperative coevolutionary approach to function optimization, in: International Conference on Parallel Problem Solving from Nature, Springer, 1994, pp. 249–257.

[32] F. Van den Bergh, A. P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE transactions on evolutionary computation 8 (3) (2004) 225–239.

[33] Z. Yang, K. Tang, X. Yao, Large scale evolutionary optimization using cooperative coevolution, Information Sciences 178 (15) (2008) 2985–2999.

[34] R. P. Wiegand, An analysis of cooperative coevolutionary algorithms, Ph.D. thesis, George Mason University Virginia (2003).

[35] E. Popovici, K. De Jong, Sequential versus parallel cooperative coevolutionary algorithms for optimization, in: Evolutionary Computation, 2006. CEC 2006. IEEE Congress on, IEEE, 2006, pp. 1610–1617.

[36] J. Kenndy, M. Clerc, Standard pso (2006).
URL `http://www.particleswarm.info/Standard_PSO_2006.c`

[37] Y. Wakasa, S. Nakaya, Distributed particle swarm optimization using an average consensus algorithm, in: Decision and Control (CDC), 2015 IEEE 54th Annual Conference on, IEEE, 2015, pp. 2661–2666.

[38] N. Gionfra, G. Sandou, H. Siguerdidjane, P. Loevenbruck, D. Faille, A novel distributed particle swarm optimization algorithm for the optimal power flow problem, in: Control Technology and Applications (CCTA), 2017 IEEE Conference on, IEEE, 2017, pp. 656–661.

[39] Y. Wakasa, S. Yamasaki, Distributed particle swarm optimization based on primal-dual decomposition architectures, in: Proceedings of the ISCIE International Symposium on Stochastic Systems Theory and its Applications, Vol. 2015, The ISCIE Symposium on Stochastic Systems Theory and Its Applications, 2015, pp. 97–101.

[40] V. Gazi, R. Ordonez, Particle swarm optimization based distributed agreement in multi-agent dynamic systems, in: Swarm Intelligence (SIS), 2014 IEEE Symposium on, IEEE, 2014, pp. 1–7.

[41] I. Navarro, E. Di Mario, A. Martinoli, Distributed particle swarm optimization-particle allocation and neighborhood topologies for the learning of cooperative robotic behaviors, in: Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, IEEE, 2015, pp. 2958–2965.

[42] W. Rivera, Scalable parallel genetic algorithms, Artificial intelligence review 16 (2) (2001) 153–168.

[43] A. H. Aguirre, A. M. Zavala, E. V. Diharce, S. B. Rionda, Copso: Constrained optimization via pso algorithm, Center for Research in Mathematics (CIMAT). Technical report No. I-07-04/22-02-2007.

Metaheuristic optimization efficiently solves the wind farm power maximization problem

Real-time performance is achieved via model-based distributed optimization

Cooperative co-evolution allows the problem distribution among the wind turbines

Power constraints are simply handled in the optimization problem via Deb's rule