Energy 208 (2020) 118310

Contents lists available at ScienceDirect

Energy

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

A data-driven evolutionary algorithm for wind farm layout optimization

Huan Long [a, b, *], Peikun Li [a], Wei Gu [a]

[a] School of Electrical Engineering, Southeast University, Nanjing, China
[b] Jiangsu Key Laboratory of Smart Grid Technology and Equipment, Nanjing, China

ARTICLE INFO

Article history:
Received 22 October 2019
Received in revised form
30 June 2020
Accepted 2 July 2020
Available online 14 July 2020

Keywords:
Wind farm layout
Wake effect
Adaptive differential evolution
Data-driven model
Function approximation

ABSTRACT

The wind farm layout model is to optimize the location of wind turbines to maximize the power output of the wind farm. Due to the complexity of the wind farm layout problem, the computation of objective function costs lots of time. To reduce the high computational cost while maintaining the solution performance, a data-driven evolutionary algorithm is proposed. An adaptive differential evolution algorithm (ADE) is proposed as the solver of the wind farm layout model. The adaption mechanism of ADE benefits the automatic adjustment of parameters in the mutation and crossover operators to achieve the optimal solution. The general regression neural network (GRNN) algorithm builds the data-driven surrogate model. The data-driven surrogate model is trained and updated using the data generated by the evolutionary algorithm throughout the evolution process. Through the data-driven surrogate model, the objective function is fast approximated and the bad candidate solutions are identified. The algorithm efficiency is greatly improved by fast filtering the bad candidate solutions. The ADE-GRNN is compared to other three conventional optimization methods based on two different wind scenarios. The results show the super-performance of ADE-GRNN in complex situations in terms of power output and execution time.

1. Introduction

The wind industry has experienced remarkable growth in the past several decades. The installed capacity of wind turbines in the grid is continuously increasing due to the minimal environmental impact. The central and distributed wind turbines have widely been utilized in the micro-grids [1]. The latest statistics released by the Global Wind Energy Council show that from 2000 to 2018, the cumulative global wind power installed capacity increased from 16.9 GW to 591 GW [2].

Given certain wind farm layout and wind direction, wake effect is the main factor affecting wind farm productivity. Upstream wind turbine extracting energy from wind causes the wind velocity deficit behind its swept area, which impairs its downstream wind turbines' energy capture [3]. Thus, it is important to determine wind turbine locations to maximize the power output of the wind farm, which is usually referred to as layout optimization or micro-siting problem [4]. The wind farm layout problem is defined as a non-deterministic polynomial hard (NP-hard) problem and usually solved by the heuristic algorithm [5]. Currently, there are two major optimization models describing the wind farm layout problem: the grid model and the coordinate model.

In the grid model, the wind farm area is discretized with a grid and each cell center represents a candidate location for placing a wind turbine. Ref [5] firstly introduced a discrete grid-based model and applied a genetic algorithm to address the layout optimization problem. Ref [6] further investigated the same model and demonstrated the effectiveness of genetic algorithms based on Ref [5]. Some other advanced heuristic search algorithms were also modified to solve the grid-based optimization model such as binary particle swarm optimization [7] and binary differential evolution [8]. In the coordinate model, the wind turbine can be located anywhere in the wind farm and its location is described by a two-dimensional Cartesian coordinate (x, y). Ref [9] firstly proposed the coordinated wind farm layout model and used the evolutionary strategy algorithm to solve it. Applications of particle swarm optimization [10] and simulated annealing algorithm [11] were also reported to solve the coordinated wind farm layout model in

* Corresponding author. School of Electrical Engineering, Southeast University, Nanjing, China.
E-mail address: hlong@seu.edu.cn (H. Long).

https://doi.org/10.1016/j.energy.2020.118310
0360-5442/

Nomenclature

N	Number of wind turbines
i, j	Index of wind turbines
v	Wind speed
R	Radius of wind turbine rotor
v_{ci}, v_r, v_{co}	Cut-in, rated and cut-out wind speed of a wind turbine
θ	Wind direction
$p(\bullet)$	Wind speed probability density function
$c(\theta)$	Scale parameter conditioned on wind direction θ
$k(\theta)$	Shape parameter conditioned on wind direction θ
(x_i, y_i)	The coordinate of wind turbine i
$VD_{j,i}$	Velocity deficit of wind turbine i caused by the wake of the wind turbine j
VD_i	Total velocity deficit of wind turbine i
a	Axial induction factor
C_T	Fixed thrust coefficient
z	Wind turbine tower height
z_0	Ground surface roughness
κ	Entrainment constant
P	Wind turbine power output
P_r	Rated power of a wind turbine
α, β	Parameters of logistic function of wind power curve
$g(\theta)$	Probability density function of wind direction θ
ξ_l	Probability mass function of $(\theta_{l-1}+\theta_l)/2$
$x_{\min}$, $x_{\max}$	Lower and upper bounds of the wind farm in x-axis direction
$y_{\min}$, $y_{\max}$	Lower and upper bounds of the wind farm in y-axis direction

subsequent studies. In this study, the coordinate wind farm layout model is applied due to its flexibility and close to the real situation.

Due to the excessive computation time of the heuristic algorithm, many studies have made efforts to reduce the computational cost while maintaining the solution performance [12]. Ref [13] proposed a two-echelon layout planning model combining the grid model and coordinate model. The grid model was used to exclude the unfeasible solution zone and limit the optimal solution search zone for the coordinate model. Ref [14] combined the caching technique and the greedy method. The velocity deficit between unmoved wind turbines in the iterative process was cached to avoid repetitive computation. Ref [15] further applied the caching technique in a differential evolution (DE) algorithm with a new coding mechanism. All of the above methods [13–15] reduced the computational time by sacrificing the search space or search capabilities of the heuristic algorithms.

Through deeply analyzing the iterative solving process of the wind farm layout problem, the computation of the objective function costs the most computational time. To reduce the computational cost, the surrogate model has been used to solve complex problems, such as the computational fluid dynamics simulation [16]. It approximates the objective and constraint functions driven by data collected in simulations, physical experiments, production processes, or daily life [17]. In this study, the data-driven surrogate model is trained by the general regression neural network (GRNN) [18]. GRNN fits the exact objective function to achieve cheaper computational cost compared with the original objective functions.

In the aspect of solution performance, the DE algorithm with caching technique [15] showed the excellent performance in solving the wind farm layout problem. The typical mutation strategies of DE include “DE/rand/1”, “DE/best/1”, and “DE/current-to-pbest/1” [19,20]. In Ref. [19], the DE with the strategy “DE/rand/1” emphasized on the global search capability with slow convergence speed. The strategy “DE/best/1” featured fast convergence with the risk of falling into local optimum. Ref [20] introduced “DE/current-to-pbest/1” to combine the “DE/rand/1” and “DE/best/1”. With the increasing iterations, the diversity of candidate solutions weakens and the evolutionary success rate decreases. To cut down the invalid mutation and crossover operations, the adaption mechanism is introduced to improve the evolutionary success rate and the ADE is proposed.

In summary, the previous wind farm layout optimization studies have focused on reducing computational cost of solution evaluation and adapting the search strategy for the specific case. However, most improvements in algorithm efficiency impair the search space or search capabilities. Improvements also lack versatility since they are designed for specific algorithms and models. This study makes efforts in proposing a more flexible and general mechanism to improve the heuristic algorithm efficiency. The data-driven surrogate model learns from the historical data generated in the evolutionary process and directly pre-evaluates candidate layouts with cheap computational cost. Compared with previous research, this scheme avoids modifying specific steps in the objective function. Therefore, it has a strong adaptability to ensure the accuracy and completeness of the layout model when the model and algorithm are changed. Besides, the searching strategy incorporates the adaptation mechanism to avoid tuning the parameters. The main contributions can be concluded as follows:

(1) The structure and mechanism of the data-driven evolutionary algorithm is proposed and presented. The proposed ADE-GRNN makes good performance on the solution effect and computational efficiency in solving the wind farm layout problem;
(2) The exact objective function is approximated by the data-driven surrogate model to cut down the computational cost and the GRNN is utilized to train the surrogate model;
(3) The ADE algorithm with the adaptive mutation strategy and crossover strategy is proposed to improve the evolutionary success rate. The adaption mechanism also efficiently reduces the requirement for parameter selection in the heuristic algorithm.

A comparative analysis of ADE-GRNN with three other optimization algorithms is presented to validate its effectiveness in terms of power output and computational time through numerical studies. The rest paper is organized as follows: Section 2 formulates the wind farm layout optimization problem, Section 3 describes the ADE-GRNN algorithm, Section 4 discusses the experimental results based on two wind scenarios and Section 5 gives the conclusion and future work.

2. Wind farm layout problem formulation

This section formulates a wind farm layout model aiming at maximizing the wind farm's power output. Each wind turbine's power output is calculated based on the power curve model and numerical integration technology. To formulate a general wind farm layout model, several assumptions are considered in this study.

A1. The topology of the wind farm is relatively flat and wind turbines are located in a two-dimensional plane.

A2. The number of wind turbines, N, is known and fixed. All the wind turbines and their power curve functions are identical.

A3. The minimal distance between any two wind turbines is set to five times of the wind turbine rotor radius, R, to ensure safety.

A4. For wind turbine i, the wind speed v follows the Weibull distribution on a given wind direction θ, expressed as:

$$p(v,\theta,c(\theta),k(\theta)) = \frac{k(\theta)}{c(\theta)}\left(\frac{v}{c(\theta)}\right)^{k(\theta)-1} \times e^{-\left(\frac{v}{c(\theta)}\right)^{k(\theta)}} \tag{1}$$

Where $p(\bullet)$ is the wind speed Weibull probability density function, v is the wind speed, and $c(\theta)$ and $k(\theta)$ are the scale and shape parameters conditioned on the wind direction θ, $0^\circ \le \theta \le 360^\circ$, respectively. The values of $c(\cdot)$ and $k(\cdot)$ can be estimated from the historical wind data.

2.1. Wake effect model

Wake effect is the main factor affecting the power output of the wind farm. When the free stream wind passes through the rotating rotor, the kinetic energy of wind is reduced. It is assumed that the upstream turbine generates a linearly expanding wake behind it. This paper adopts the Jensen's wake model [3] to quantify the wake effect.

Suppose that the downstream wind turbine i located at (x_i, y_i) is affected by the upstream wind turbine j located at (x_j, y_j). The velocity deficit of wind turbine i caused by the wake of the wind turbine j is denoted as $VD_{j,i}$, and calculated by (2)-(4).

$$a = 0.5\left(1-\sqrt{1-C_T}\right), \kappa = 0.5/\ln(z/z_0) \tag{2}$$

$$d_{j,i} = \left|(x_j - x_i)\cos\theta + \left(y_j - y_i\right)\sin\theta\right| \tag{3}$$

$$VD_{j,i} = 1 - v_i/v_j = 2a/\left(1+\kappa d_{j,i}/R\right)^2 \tag{4}$$

where a is the axial induction factor, C_T is the fixed thrust coefficient [21], z is the wind turbine tower height, z_0 is the ground surface roughness, κ is the entrainment constant, v_i is the wind speed at downstream wind turbine i, and v_j is the wind speed at upstream wind turbine j.

Afterward, the total velocity deficit of wind turbine i affected by all other turbines is calculated as (5).

$$VD_i = \min\left\{\sqrt{\sum_{j=1\,j\neq i}^{N}\left(VD_{j,i}\right)^2}, 1\right\},\ i = 1,2,\ldots,N \tag{5}$$

It is worth noting that only the scale parameter $c(\theta)$ of the Weibull distribution is affected by wake effect [9], shown as (6).

$$c_i'(\theta) = c(\theta)\times(1-VD_i),\ i = 1,2,\ldots,N \tag{6}$$

2.2. Power curve model

A power curve function can be used to describe the relationship between the power output P and wind speed v. The logistic function is employed in this paper and the values of α and β are determined according to the actual data [13].

$$P = f(v) = \begin{cases} 0 & v \ge v_{co}, v < v_{ci} \\ \dfrac{e^v}{\alpha+\beta e^v} & v_{ci} \le v < v_r \\ P_r & v_r \le v < v_{co} \end{cases} \tag{7}$$

In (7), when $v \ge v_{co}$ or $v < v_{ci}$, the wind turbine does not work. When v ranges from v_r to v_{co}, the wind turbine maintains the rated power output P_r.

2.3. Numerical integration of wind power output

The expected power output of wind turbine i is presented as (8), which is integrated of the product of (1) and (7). The detailed derivation of the numerical integration of (8) is shown in the Appendix.

$$E(P_i) = \int_{0^\circ}^{360^\circ} g(\theta)\int_0^\infty f(v)\frac{k(\theta)v^{k(\theta)-1}}{c_i'(\theta)^{k(\theta)}} \times e^{-\left(\frac{v}{c_i'(\theta)}\right)^{k(\theta)}} dv d\theta \tag{8}$$

where $g(\theta)$ is the probability density function of θ. Since $f(v)$ is a piecewise function in (7), (8) can be divided into four intervals: $[0, v_{ci})$, $[v_{ci}, v_r)$, $[v_r, v_{co})$ and $[v_{co}, +\infty)$.

- In $[0, v_{ci})$ and $[v_{co}, +\infty)$, $f(v) = 0$ and $P_i = 0$;
- In $[v_r, v_{co})$, (8) can be simplified to (9);

$$E(P_i) = \int_{0^\circ}^{360^\circ} g(\theta)\times P_r \times \left(e^{-(v_r/c_i'(\theta))^{k(\theta)}} - e^{-(v_{co}/c_i'(\theta))^{k(\theta)}}\right) d\theta \tag{9}$$

- In the interval $[v_{ci}, v_r)$, it is challenging to obtain the integration analytically and the numerical integration technique is adopted [22]. In this case, wind speed v is quantized into s intervals with the same width: $[v_{ci}, v_1)$, $[v_1, v_2)$, …,$[v_{s\text{-}1}, v_r)$. Similarly, wind direction θ is quantized into h intervals with the same width: $[0^\circ, \theta_1)$, $[\theta_1, \theta_2)$,…,$[\theta_{h\text{-}1}, 360^\circ)$. By replacing θ with $(\theta_{l-1}+\theta_l)/2$ and $\int_{\theta_{l-1}}^{\theta_l} g(\theta)d\theta$ with ξ_n, (8) is approximated by (10).

$$E(P_i)=\sum_{l=1}^{h}\xi_l\Bigg\{P_r\times\left(e^{-(v_r/c_i'((\theta_{l-1}+\theta_l)/2))^{k((\theta_{l-1}+\theta_l)/2)}}-e^{-(v_{co}/c_i'((\theta_{l-1}+\theta_l)/2))^{k((\theta_{l-1}+\theta_l)/2)}}\right)+\sum_{j=1}^{s}\left(e^{-(v_{j-1}/c_i'((\theta_{l-1}+\theta_l)/2))^{k((\theta_{l-1}+\theta_l)/2)}}-e^{-(v_j/c_i'((\theta_{l-1}+\theta_l)/2))^{k((\theta_{l-1}+\theta_l)/2)}}\right)\frac{e^{(v_{j-1}+v_j)/2}}{\alpha+\beta e^{(v_{j-1}+v_j)/2}}\Bigg\} \tag{10}$$

2.4. Wind farm layout model

The layout of wind turbines is optimized to maximize the total power output of a wind farm. The general wind farm layout optimization model is formulated as (11). The first two constraints make the wind turbine located within the rectangular restricted wind farm area. Note that the rectangular boundary of a wind farm can be easily modified to a circular or irregular polygonal boundary by revising the constraints or adding penalty functions in restricted zones. The third constraint ensures that the minimal distance between any adjacent two wind turbines.

$$\begin{aligned}&\underset{x_i,y_i}{\operatorname{argmax}}\sum_{i=1}^{N}E(P_i)\\&\text{s.t. } x_{\min}+R\le x_i\le x_{\max}-R\\&\quad y_{\min}+R\le y_i\le y_{\max}-R\\&\quad\sqrt{(x_i-x_j)^2+\left(y_i-y_j\right)^2}\ge 5R,\ i,j=1,2,\ldots,N\ ,\ i\neq j\end{aligned} \tag{11}$$

3. The structure of data-driven evolutionary algorithm

The structure of the proposed ADE-GRNN can be divided into two stages, shown as Fig. 1. In stage I, ADE runs independently and collects training samples for initializing GRNN, presented in the left part of Fig. 1. ADE is firstly initialized with the random parent population and then follows the steps in each iteration.

- The offspring population is generated via variation operators, including the mutation operator and crossover operation;
- The candidate solutions in the offspring population assess their fitness values by the exact objective function;
- The selection operator selects the new parent population for the next iteration based on the fitness value;
- The controlling parameters of the mutation and crossover operations are adapted.

All the candidate solutions are collected after function evaluation. When iterations reach the pre-set number, the ADE-GRNN enters stage II and is presented in the right part of Fig. 1. In stage II, the GRNN is firstly trained as the surrogate model based on the data collected in stage I. The surrogate model approximates the fitness values of the candidate solutions generated by variation operators and filter the bad candidate solutions. Only the candidate solutions with high approximated fitness values enter the exact function evaluation. The rest ones are weeded out to shrink the exact fitness evaluation time. In each iteration, the surrogate model is updated by incorporating the latest sample information.

The whole algorithm ends until satisfying the expected number of iteration. This filtering scheme ensures the algorithm efficiency and enhances the approximation accuracy of the surrogate model throughout the evolution process.

4. The details of proposed ADE-GRNN algorithm

The structure of the ADE-GRNN is described in section 3. The

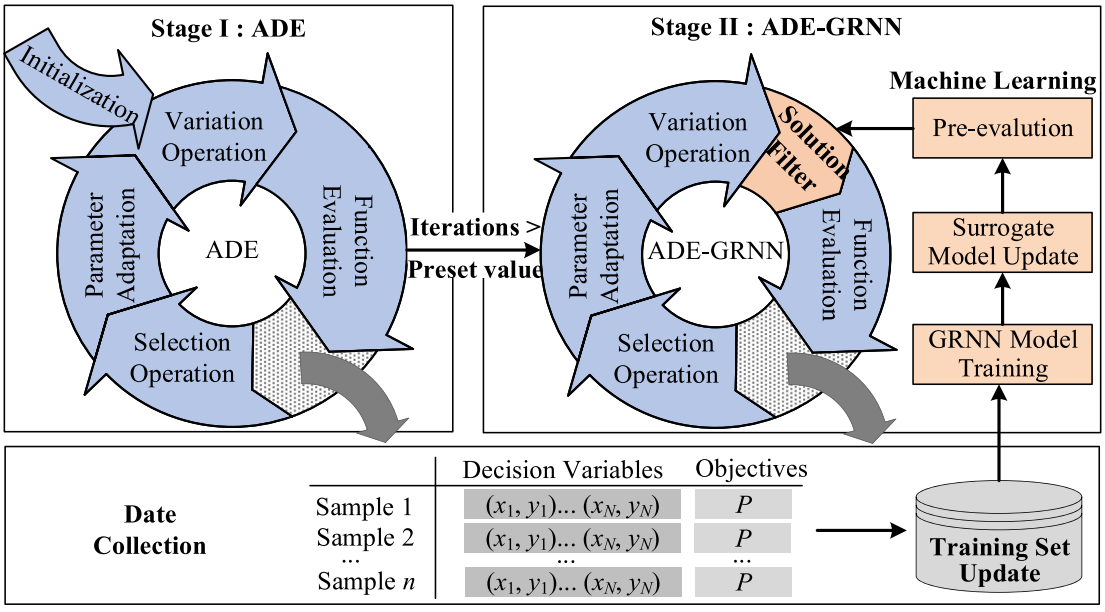

Fig. 1. The structure and flowchart of ADE-GRNN algorithm.

details of ADE and GRNN are presented in this section.

4.1. Adaptive differential evolutionary algorithm

The ADE is composed of mutation operation, crossover operation, selection operation and parameter adaptation. At the g-th iteration, the parent population is composed of NP individuals $\{s_{i,g} = (s_{1,i,g}, s_{2,i,g}, \ldots, s_{2N,i,g}) | i = 1, 2, \ldots, NP\}$. Each individual $s_{i,g}$ represents a candidate solution and is encoded by the coordinates of all wind turbines $\{(x_{1,i,g}, y_{1,i,g}), (x_{2,i,g}, y_{2,i,g}), \ldots, (x_{N,i,g}, y_{N,i,g}) | i = 1, 2, \ldots, NP\}$. The ADE first randomly initialized the parent population and then enters the evolution process, shown in the left part of Fig. 1.

4.1.1. Mutation operation

At the g-th iteration, mutation vectors $v_{i,g}$ are generated based on the current parent population $\{s_{i,g} | i = 1,2, \ldots, NP\}$ by using different mutation strategies. Some typical mutation strategies [19,20] are shown as (12)–(14).

$$\text{"DE/rand/1"}: \mathbf{v}_{i,g} = \mathbf{s}_{r_0,g} + F_i \times (\mathbf{s}_{r_1,g} - \mathbf{s}_{r_2,g}) \tag{12}$$

$$\text{"DE/ best/1"}: \mathbf{v}_{i,g} = \mathbf{s}_{best,g} + F_i \times (\mathbf{s}_{r_1,g} - \mathbf{s}_{r_2,g}) \tag{13}$$

$$\text{"DE/current} - \text{to} - \text{pbest/1"}: \mathbf{v}_{i,g} = \mathbf{s}_{i,g} + F_i \times \left(\mathbf{s}^{p}_{best,g} - \mathbf{s}_{i,g}\right) + F_i \times (\mathbf{s}_{r_1,g} - \tilde{\mathbf{s}}_{r_2,g}) \tag{14}$$

where the indexes r_0, r_1, r_2 are different integers randomly generated from [1, NP], which should differ from the index i. $s_{best,g}$ is the target vector with the best fitness value in the parent population at generation g. $\tilde{s}_{r\ 2,g}$ is likely to be chosen from an optional archive containing historical failed parent solutions. s p $best,g$ is randomly chosen as one of the top 100p% individuals in the current population. F_i is the mutation factor used to scale difference vectors which usually ranges in (0, 2] [23].

In order to achieve a strong global search capability while maintaining the convergence speed, a new mutation strategy, named as ADE/best/1, is proposed as (15). The ADE/best/1 is originally from DE/best/1. The $s_{best,g}$ guarantees the fast convergence speed and the difference vector D adds the solution diversity.

$$\begin{cases} \mathbf{v}_{i,g} = \mathbf{s}_{best,g} + \mathbf{D} \\ \mathbf{D} = \mathrm{sgn}(x)\left(F_{1i} \times \mathbf{s}_{r_1,g} - F_{2i} \times \mathbf{s}_{r_2,g}\right),\ x \in U[-1,1] \end{cases} \tag{15}$$

For each $s_{i,g}$, the new mutation factors, F_{1i} and F_{2i}, are randomly generated by $N(\mu_{F1}, \sigma_1)$ and $N(\mu_{F2}, \sigma_2)$ respectively. μ_{F1} and μ_{F2} are adaptive controlling parameters updated at the end of each iteration. $s_{r1,g}$ is a randomly picked individual from the parent population. $s_{r2,g}$ is generated by disrupting the sequence of the wind turbines in $s_{r1,g}$. Fig. 2 shows how the mutation strategy (15) operates.

4.1.2. Crossover operation

After mutation, a binomial crossover operation is applied to mix mutation vector $v_{i,g}$ with its corresponding target/parent vector $s_{i,g}$ to form the trail/offspring vector $u_{i,g} = (u_{1,i,g}, u_{2,i,g}, \ldots, u_{N,i,g})$.

$$u_{j,i,g} = \begin{cases} v_{j,i,g}, & \text{if rand}(0,1) \le CR_i \text{ or } j = j_{rand} \\ s_{j,i,g}, & \text{otherwise} \end{cases}, j = 1, 2, \ldots, 2N \tag{16}$$

where rand(0, 1) is a uniform random number in the range (0, 1). The crossover probability $CR_i \in (0, 1)$ determines the amount of information inherited from target vector $s_{i,g}$ and mutation vector $v_{i,g}$ respectively. The j_{rand} is a randomly chosen integer to ensure that $u_{i,g}$ differs from its corresponding $s_{i,g}$.

Considering the physical meaning of individual $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ in the wind farm layout problem, the replaced elements in $s_{i,g}$ means the positions of some wind turbines are changed. In order to intuitively reflect the number of wind turbines moved in each iteration, the crossover factor CR_i is deprecated. The modified crossover strategy is presented as (17).

$$[u_{2*j-1,i,g}, u_{2*j,i,g}] = \begin{cases} [v_{2*j-1,i,g}, v_{2*j,i,g}] & \text{if } j \in \mathbf{j}_{rand} \\ [s_{2*j-1,i,g}, s_{2*j,i,g}] & \text{otherwise} \end{cases}, j = 1, 2, \ldots, N \tag{17}$$

In (17), the set j_{rand} contains all randomly selected wind turbines and the size of j_{rand} is denoted as $n_i\_j_{rand}$. The $n_i\_j_{rand}$ is randomly generated by $N(n\_j_{rand}, \sigma_3)$ and then rounded to the closest integer belonging to [1, N]. The n_j_{rand} is an adaptive controlling parameter and updated at the end of each iteration. After crossover, the $u_{i,g}$ is checked whether satisfying the constraints in (11). If violated, the difference vector D will be reduced by half and the crossover operation is operated again.

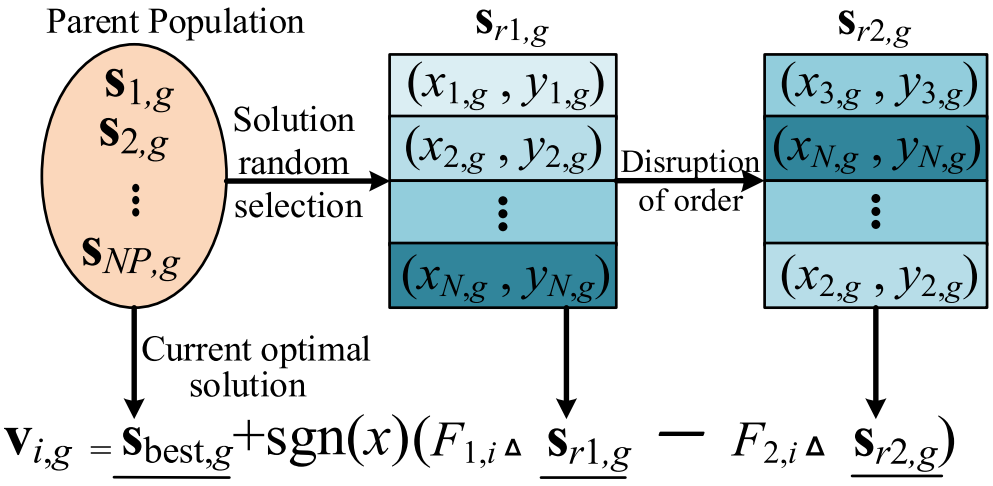

Fig. 2. Mutation operation "ADE/best/1"

4.1.3. Selection operation

After the mutation operation and crossover operation, the exact fitness values of individuals in $s_{i,g}$ and $u_{i,g}$ are evaluated. The selection operation selects the better individual from $s_{i,g}$ and $u_{i,g}$ based on their exact fitness values to generate $s_{i,g+1}$. The operation is called a successful update if $u_{i,g}$ is better than $s_{i,g}$.

$$\mathbf{s}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g}, & \text{if } f(\mathbf{u}_{i,g}) > f(\mathbf{s}_{i,g}) \\ \mathbf{s}_{i,g}, & \text{otherwise} \end{cases} \tag{18}$$

4.1.4. Parameter adaptation in ADE

The controlling parameters, μ_{F1}, μ_{F2} and n_j_{rand} display the adaptation of ADE in the mutation and crossover operations. The corresponding mutation and crossover factors, $F_{1,i}$, $F_{2,i}$ and $n_i\_j_{rand}$, are called successful factors, which are beneficial for generating good $u_{i,g}$ and improve evolutionary success rate. The recent successful factors are recorded in sets S_{F1}, S_{F2} and S_{jrand}. Their information is propagated to the next generation to adapt the mutation and crossover operations. The old successful factors in the set are gradually replaced to prevent that the new ones are just accidental successful parameters.

Thus, $F_{1,i}$, $F_{2,i}$ and $n_i\_j_{rand}$ are firstly recorded in sets S_{F1}, S_{F2} and S_{jrand}, respectively. At the end of the current generation, several oldest factors removed in S_{F1}, S_{F2} and S_{jrand} to keep the set size unchanged. The μ_{F1}, μ_{F2} and n_j_{rand} are then updated through mean(S_{F1}), mean(S_{F2}) and mean(S_{jrand}).

4.2. General regression neural network for data-driven surrogate model

Ref [18] introduces the basic GRNN in 1991. The foundation of GRNN is the radial basis function network and the theory of kernel regression. GRNN has good nonlinear approximation ability as well as excellent performance of anti-interference [24]. When the sample size is small, the prediction results can still be good [25]. Moreover, The GRNN speed is much faster in model building compared with back propagation neural networks because it does not need an iterative training [26]. Based on these features, GRNN is combined with ADE to improve the algorithm efficiency.

4.2.1. General regression neural network

The topology of GRNN consists of four layers as presented in Fig. 3. The input neurons are in the first layer which receive and distribute information to the second layer of pattern neurons. The number of pattern neurons is equal to the number of samples in the training set. A pattern neuron computes its output f_j ($s_{i,g}$) through the transfer function (19).

$$f_j(\mathbf{s}_{i,g}) = \exp\left(-\frac{(\mathbf{s}_{i,g} - \mathbf{l}_j)^T (\mathbf{s}_{i,g} - \mathbf{l}_j)}{2\sigma^2} \right) \tag{19}$$

where $s_{i,g}$ is the input vector, l_j is the input of the j-th sample in the training set corresponding to the j-th pattern neuron. The outputs of pattern neurons are then passed on to the summation neurons in the third layer. There are two types of summation neurons that perform arithmetic summation and weighted summation according to (20), respectively.

$$S_1 = \sum_{j=1} f_j(\mathbf{s}_{i,g}) p_j, \quad S_2 = \sum_{j=1} f_j(\mathbf{s}_{i,g}) \tag{20}$$

where p_j is the output of j-th sample in the training set. The final layer covers the output neuron which is decided by S_1/S_2.

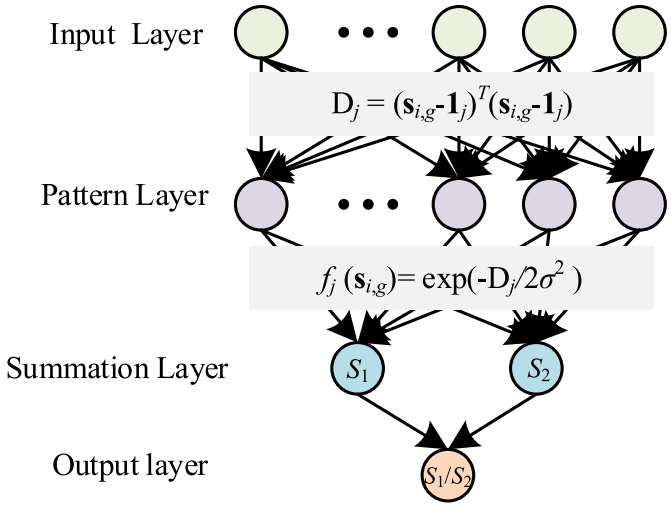

Fig. 3. Schematic diagram of GRNN architecture.

In this paper, the input of GRNN is the wind farm layout and the output of GRNN is the scalar of the total power output of the wind farm. Fig. 3 denotes the architecture of GRNN, where $s_{i,g}$ is the candidate layout solution to be evaluated, l_j is the j-th layout in the training set and p_j is the corresponding total power output with l_j. The number of neurons in the pattern layer is decided by trial and error to balance memory usage, calculation speed and prediction performance.

4.2.2. Data processing

As discussed in section 4.1, each individual in ADE is encoded as $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ containing N wind turbines' coordinates. If such individuals are directly used as the input of GRNN, the information transmitted to GRNN is only a set of coordinates, rather than the layout. For example, if any two wind turbines exchange their positions, the layout remains the same which may mislead the learning of GRNN.

Thus, data processing is conducted for each individual. Firstly, all the wind turbines are sorted by x coordinate. The smaller the x coordinate is, the smaller the serial number is. If the wind turbines with the same x coordinate, they are further sorted by y coordinate. After data processing, the wind turbines' coordinates are ordered sequentially to form the individual. Fig. 4 shows how the data processing incorporates the layout information into the individual where T_i denotes the ith wind turbine.

4.2.3. Training data collection and update

In stage I, ADE runs independently. In each iteration, all the candidate solutions with their corresponding fitness values are collected in the training set for initializing GRNN. In each iteration of stage II, some candidate solutions with low approximated values are filtered before function evaluation. Therefore, only the candidate solutions which are evaluated by the exact objection function are added into the training set to replace the oldest samples. Since the candidate solutions always develop towards an optimal direction in the evolutionary algorithm, too old samples have a negative effect on the training effect. This scheme incorporates the latest information into GRNN and prevents the stale data from misleading the GRNN learning.

4.2.4. Hyperparameter selection of GRNN

GRNN is a one-pass learning algorithm, which is different from the traditional backpropagation neural network. Once the input of

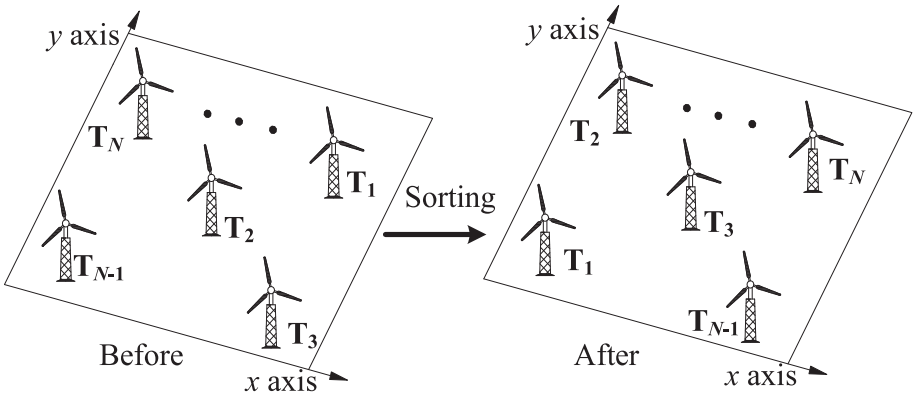

Fig. 4. Data processing: sequence the wind turbines.

the GRNN is given, its architecture and weights are determined accordingly. Therefore, tuning GRNN is essentially to optimize the smoothing parameter σ which represents the width of the calculated Gaussian curve for the probability density function [27]. The optimal value of σ can be determined by using the validation data when the GRNN model gives rise to the minimum validation error [28].

ADE without GRNN is performed once for collecting a set of learning samples as the validation data. The k-fold cross validation technique is adopted to optimize the smoothing parameter [29]. The learning samples are split randomly into k equal-sized groups firstly. One of the groups is used as the test set and the rest are used as the training set. The model is trained on the training set and tested on the test set by the Mean Absolute Error (MAE). The process is repeated k times with each unique group used exactly once as the test set and the k results then are averaged to produce a single estimation. Finally, the optimum smoothing parameter is determined by the one corresponding to the minimum value of MAE.

5. Computational studies

In order to validate the effectiveness of the proposed methodology, ADE, Comprehensive Learning Particle Swarm Optimizer (CLPSO) [30] and J-Adaptive Differential Evolution (JADE) [20] are introduced to compare with it. ADE is the simplified version of the proposed algorithm without GRNN. In CLPSO, all other particles' historical best information is used to update the velocity of any one particle. To compare and analyze the impact of the mutation operation well, we revise the JADE and name it as JADE_m, which keeps the same mutation operation, DE/current-to-pbest/1, and adopts the crossover operation of ADE. The CLPSO is regarded as the benchmark algorithm.

To compare the algorithms' performance, numerical examples and industrial case studies are presented. To reduce the impact of the randomness of the heuristic algorithms, 5 independent runs are performed for each case. The experiments are conducted with the Matlab in a PC laptop with a 2.7-GHz Intel Core i5 processor and 8 GB of memory.

5.1. Parameter settings

The specification of the wind turbine, GE1.5-77, is shown in Table 1. For each algorithm, the maximum fitness evaluation times is set to 150000, the population size is set to 40 and the maximum number of iterations is set to 3750. For GRNN, the size of training set, i.e., the number of neurons in the pattern layer is set to 5000. The 10-fold cross validation is adopted to tune the smoothing parameter σ. Experiments are conducted in the range from 0.001 to 1 and σ is determined as 0.01 finally. Table 2 lists the detailed settings of algorithm parameters.

The shape of the wind farm is set to a square area with varying side lengths as shown in Table 3. Table 4 and Table 5 display the details of two wind scenarios. Wind scenario 1 (WS1) is fit based on the historical wind resource data collected from the industrial wind farm located in Iowa, USA. The values of the $k(\theta)$ and $c(\theta)$ are estimated based on the historical data. Wind scenario 2 (WS2) is based on the setting in Ref. [11]. The wind direction from west to east is defined as 0° and the wind direction from south to north is defined as 90°. It is clear that wind scenario 1 has wider prevailing wind and is more complex than wind scenario 2.

5.2. Computational results and discussion

In this section, the power output, runtime and typical layout

Table 1
The specification of wind turbine.

Parameter	Explanation	Value
z	Hub height of wind turbine (m)	80
R	Rotor radius of wind turbine (m)	40
C_T	Thrust coefficient of wind turbine	0.8
P_r	Rated power output of wind turbine (kW)	1500
κ	Environment constant	0.01
v_{ci}	Cut-in wind speed of wind turbine (m/s)	3.5
v_r	Rated wind speed of wind turbine (m/s)	14
v_{co}	Cut-out wind speed of wind turbine (m/s)	25
α	The parameter of wind power output function	6.0268
β	The parameter of wind power output function	0.0007

Table 2
Parameter settings of algorithms.

Algorithm	Parameter Settings
ADE-GRNN	$\mu_{F1} = \mu_{F2} = 1$, $n\_j_{rand} = 5$, $\sigma_1 = \sigma_2 = \sigma_3 = 1$, $\sigma = 0.01$
ADE	$\mu_{F1} = \mu_{F2} = 1$, $n\_j_{rand} = 5$, $\sigma_1 = \sigma_2 = \sigma_3 = 1$
CLPSO	ω: linear decreasing from 0.9 to 0.4, $c = 1.49445$
JADE_m	$\mu_F = 0.5$, $n\_j_{rand} = 5$, $\sigma_1 = 0.2$, $\sigma_2 = 1$

Table 3
Side lengths (m) of the wind farm with $N = 15$–100.

N	15–25	30	35	40	60	80	100
Side lengths	2000	2200	2400	2600	3100	3600	4000

Table 4
Wind scenario 1.

l	$[\theta_{l-1},\theta_l)$	$k(\theta)$	$c(\theta)$	ξ_l	l	$[\theta_{l-1},\theta_l)$	$k(\theta)$	$c(\theta)$	ξ_l
1	[0°,15°)	2	7	0.0003	13	[180°,195°)	2	10	0.1909
2	[15°,30°)	2	5	0.0072	14	[195°,210°)	2	8.5	0.1162
3	[30°,45°)	2	5	0.0237	15	[210°,225°)	2	8.5	0.0793
4	[45°,60°)	2	5	0.0242	16	[225°,240°)	2	6.5	0.0082
5	[60°,75°)	2	5	0.0222	17	[240°,255°)	2	4.6	0.0041
6	[75°,90°)	2	4	0.0301	18	[255°,270°)	2	2.6	0.0008
7	[90°,105°)	2	5	0.0397	19	[270°,285°)	2	8	0.001
8	[105°,120°)	2	6	0.0268	20	[285°,300°)	2	5	0.0005
9	[120°,135°)	2	7	0.0626	21	[330°,315°)	2	6.4	0.0013
10	[135°,150°)	2	7	0.0801	22	[315°,330°)	2	5.2	0.0031
11	[150°,165°)	2	8	0.1025	23	[330°,345°)	2	4.5	0.0085
12	[165°,180°)	2	9.5	0.1445	24	[345°,360°)	2	3.9	0.0222

Table 5
Wind scenario 2.

l	$[\theta_{l-1},\theta_l)$	$k(\theta)$	$c(\theta)$	ξ_l	l	$[\theta_{l-1},\theta_l)$	$k(\theta)$	$c(\theta)$	ξ_l
1	[0°,15°)	2	13	0	13	[180°,195°)	2	13	0.01
2	[15°,30°)	2	13	0.01	14	[195°,210°)	2	13	0.01
3	[30°,45°)	2	13	0.01	15	[210°,225°)	2	13	0.01
4	[45°,60°)	2	13	0.01	16	[225°,240°)	2	13	0.01
5	[60°,75°)	2	13	0.01	17	[240°,255°)	2	13	0.01
6	[75°,90°)	2	13	0.20	18	[255°,270°)	2	13	0.01
7	[90°,105°)	2	13	0.60	19	[270°,285°)	2	13	0.01
8	[105°,120°)	2	13	0.01	20	[285°,300°)	2	13	0.01
9	[120°,135°)	2	13	0.01	21	[330°,315°)	2	13	0.01
10	[135°,150°)	2	13	0.01	22	[315°,330°)	2	13	0.01
11	[150°,165°)	2	13	0.01	23	[330°,345°)	2	13	0.01
12	[165°,180°)	2	13	0.01	24	[345°,360°)	2	13	0

of four compared algorithms under WS1 and WS2 are displayed. The impact of GRNN and the benefit of the adaption mechanism is further discussed based on the experiment results.

5.2.1. Computational results

Table 6 presents the average and standard deviation of the power output derived from four compared algorithms under WS1. It is obvious that ADE and ADE-GRNN perform the best and CLPSO performs the worst. This advantage performs more and more obvious with the increasing number of wind turbines. JADE_m performs worse than ADE, which proves the effectiveness of the proposed mutation strategy ADE/best/1. The average power output of ADE and ADE-GRNN are very close under different numbers of wind turbines. The standard deviation of power output of ADE-GRNN is smaller than ADE in most cases. It is implied that the solutions filtered by GRNN have seldom impact on the optimal solution search. Besides, it also benefits for achieving the stable solution due to the small standard deviation compared with ADE.

Table 7 summarizes the average and standard deviation of the power output derived from four compared algorithms under WS2. The results and conclusion keep consistent with WS1. The power output of WS2 is much larger than WS1 due to its concentrated distribution of wind resources. In addition, the differences among four compared algorithms under WS2 are smaller than WS1, especially the small number of wind turbine. The average power output of four algorithms under $N = 15$ and WS2 is close. It is concluded that the concentrated distribution of wind resources and large wind turbines add the difficulty of problem solving and has higher requirement of the algorithm.

Fig. 5 presents the evolution of the average power output achieved by the four compared algorithms on $N = 25$ under WS1 and WS2. The average power output of ADE and ADE-GRNN reaches a large improvement rapidly at the early stage of the evolution and maintains the highest throughout the evolutionary process. It is implied the proposed adaption mechanism accelerates the convergence speed and improves the evolution success rate. Besides, all the algorithms achieve faster convergence under WS2 than WS1, which also proves the impact of wind resources on algorithms.

Fig. 6 displays the runtime of four compared algorithms versus the number of wind turbines. It is observed that ADE, JADE_m and CLPSO have a similar runtime. It is because their fitness evaluation times are the same under the experiment setting. The runtime of ADE-GRNN is considerably lower than other algorithms with an increase in the number of wind turbines. In the case of $N = 100$,

Table 6
Experimental results of four compared algorithms under WS1 (kW).

N	CLPSO (AVG ± std dev)	JADE_m (AVG ± std dev)	ADE (AVG ± std dev)	ADE-GRNN (AVG ± std dev)
15	5923.60 ± 35.84	6107.9 ± 33.28	6205.06 ± 49.51	6233.49 ± 30.46
20	7125.12 ± 75.66	7434.44 ± 12.89	7763.93 ± 92.35	7752.22 ± 80.98
25	7892.72 ± 71.44	8434.97 ± 109.6	8787.33 ± 109.29	8825.88 ± 54.79
30	8902.46 ± 73.32	9420.87 ± 56.34	10203.68 ± 118.34	10106.08 ± 126.14
35	9977.47 ± 67.45	10511.06 ± 32.04	11487.60 ± 147.57	11529.33 ± 113.32
40	11001.07 ± 77.25	11698.84 ± 30.26	12715.59 ± 215.27	12832.81 ± 79.32
60	14024.53 ± 145.60	15049.17 ± 170.34	17009.42 ± 270.61	16861.76 ± 187.79
80	16935.41 ± 138.92	18121.56 ± 311.50	20867.56 ± 133.23	20584.03 ± 197.19
100	19206.77 ± 126.08	20861.99 ± 195.75	24201.86 ± 328.48	23874.00 ± 122.53

Table 7
Experimental results of four compared algorithms under WS2 (kW).

N	CLPSO (AVG ± std dev)	JADE_m (AVG ± std dev)	ADE (AVG ± std dev)	ADE-GRNN (AVG ± std dev)
15	12874.64 ± 27.85	12984.75 ± 15.76	13024.86 ± 20.97	13039.15 ± 15.36
20	16304.34 ± 105.90	16668.42 ± 43.46	16963.78 ± 79.25	17041.32 ± 57.32
25	18824.44 ± 76.65	19497.92 ± 57.16	20054.21 ± 148.61	20073.4 ± 166.06
30	21747.00 ± 138.87	22747.57 ± 48.03	23726.28 ± 140.3	23662.49 ± 155.76
35	24737.83 ± 113.95	25988.99 ± 149.62	27211.34 ± 149.53	27215.36 ± 178.91
40	27731.31 ± 203.63	29024.64 ± 91.28	30686.17 ± 149.48	30615.07 ± 178.95
60	36894.43 ± 171.90	39161.64 ± 250.24	42625.23 ± 150.09	42625.68 ± 153.11
80	45930.86 ± 238.22	48633.58 ± 175.08	53968.50 ± 191.88	53796.22 ± 273.52
100	53432.41 ± 310.76	56873.12 ± 191.66	64510.85 ± 386.18	64319.46 ± 222.36

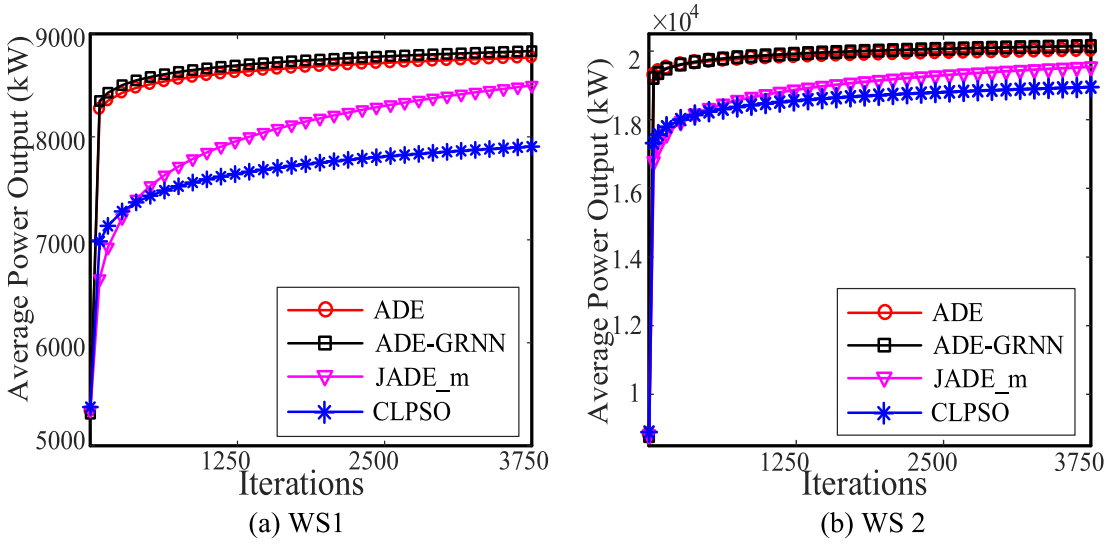

Fig. 5. The evolution of the average power output provided by four algorithms on $N = 25$.

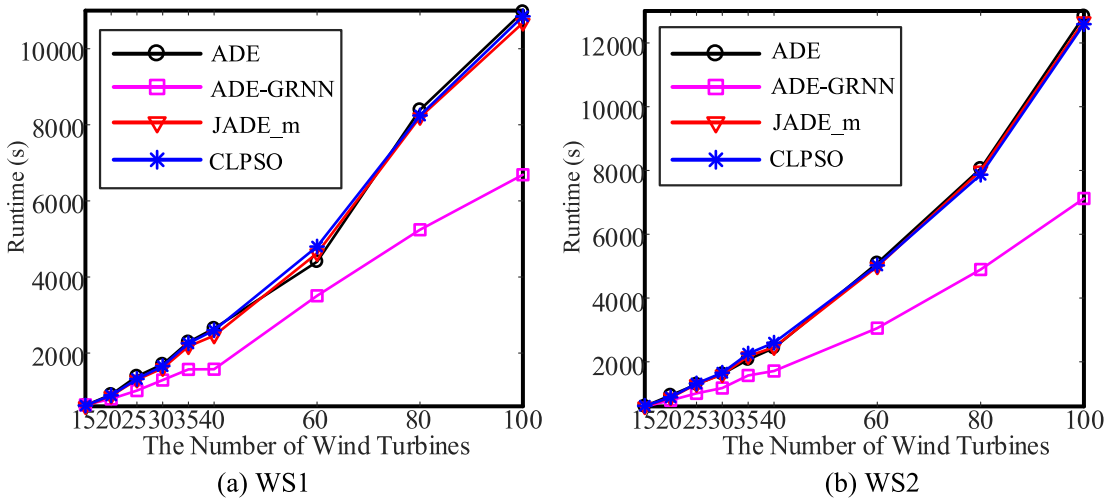

Fig. 6. Runtime of four compared algorithms.

ADE-GRNN is almost twice as fast as ADE, JADE_m and CLPSO.

The best layouts of four compared algorithms on $N = 25$, 60 and 100 under WS1 and WS2 are shown in Fig. 7 and Fig. 8. It is obvious that the layouts of ADE-GRNN and ADE are relatively orderly compared to those of JADE_m and CLPSO, especially in the case of a small number of wind turbines. It is also implied the solution of ADE-CRNN and ADE is better than JADE_m and CLPSO.

The wake effect decreases with the distance increasing. In Figs. 7 and 8, ADE-GRNN and ADE enlarges the distances among the wind turbines along the predominant wind direction (i.e., from 120° to 225° in WS1 and from 75° to 105° in WS2). As a result, many wind turbines in the layout are located close to the boundaries of the wind farm to decrease the wake effect. At the boundaries, the wind turbines present the "W" arrangement to further reduce the wake loss. On the contrary, the layouts of JADE_m and CLPSO are relatively disordered. In the case of $N = 60$ and $N = 100$, the wind turbines almost evenly located in the wind farm.

In summary, two typical wind scenarios with different features are employed to compare the four algorithms. WS1 and WS2 have mutually perpendicular predominant wind directions and different degrees of concentrations, which can avoid the coincident computational results. The solution quality, runtime, and layout effect of the four compared algorithms share the same trend under the two wind scenarios. The power output of ADE, ADE-GRNN and the runtime of ADE-GRNN take a great advantage over other algorithms, especially with large wind turbine numbers. The typical layouts of ADE and ADE-GRNN are more orderly than CLPSO and JADE_m. Apart from the similarities, higher concentrations of the wind scenario bring higher power output and accelerate the convergence speed of the four algorithms. The gap between the solving effects of the algorithms is reduced to some extent with a more concentrated prevailing wind.

5.2.2. *The impact of GRNN*

As depicted in Tables 6 and 7, ADE-GRNN and ADE have almost the same performance in power output when N is small. When N grows up, especially in the cases of 80 and 100, ADE is slightly better than ADE-GRNN. It is because the solution space increases dramatically with the increase of N. More candidate solutions in the training set are needed to support the estimation accuracy of the

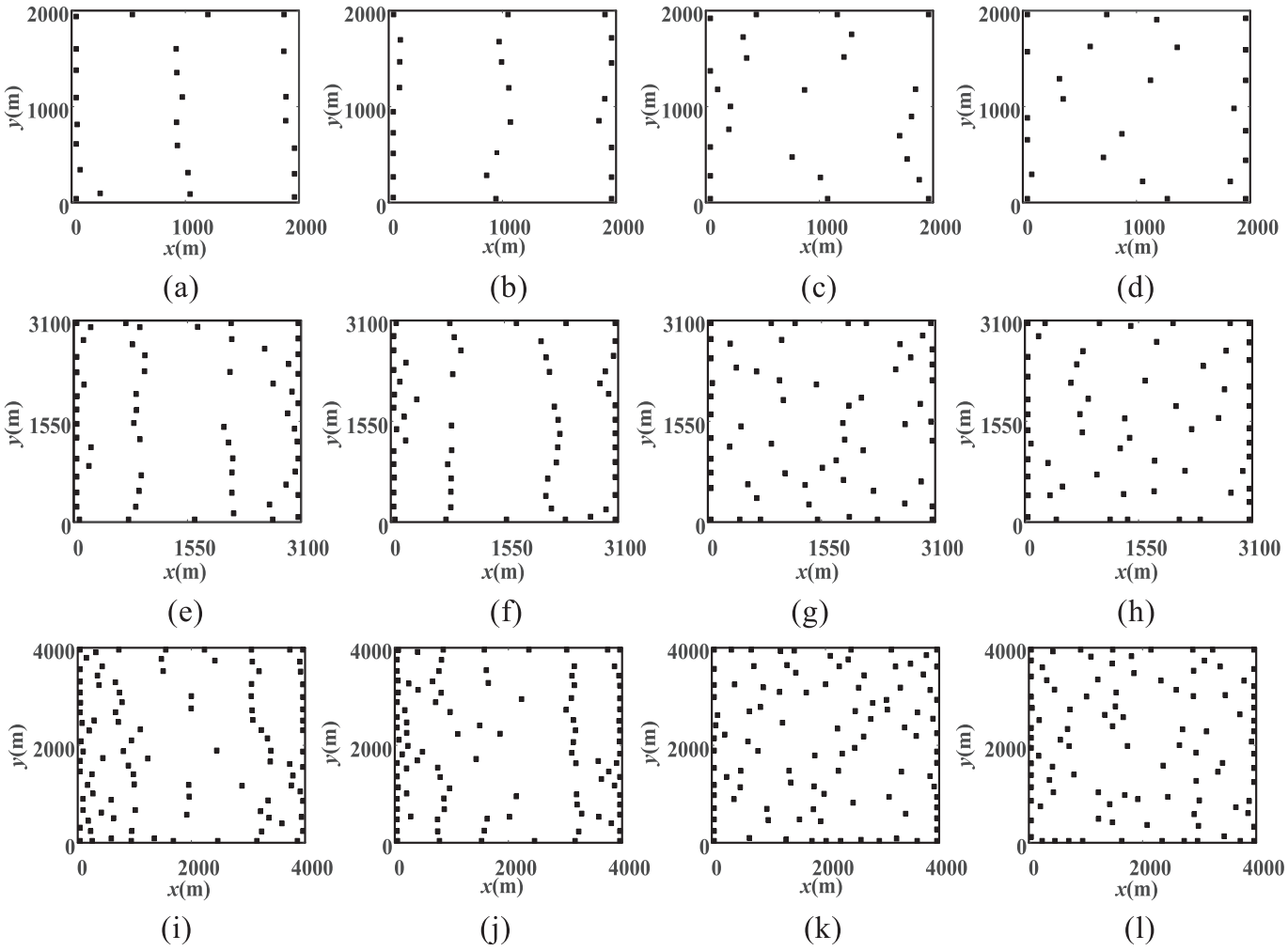

Fig. 7. The best layouts of four compared algorithms with different N in WS1. (a) ADE-GRNN, $N = 25$, (b) ADE, $N = 25$, (c) JADE_m, $N = 25$, (d) CLPSO, $N = 25$, (e) ADE-GRNN, $N = 60$, (f) ADE, $N = 60$, (g) JADE_m, $N = 60$, (h) CLPSO, $N = 60$, (i) ADE-GRNN, $N = 100$, (j) ADE, $N = 100$, (k) JADE_m, $N = 100$, (l) CLPSO, $N = 100$.

surrogate model.

In each iteration of stage II, all the candidate solutions are divided into two halves based on their approximated fitness values by GRNN. To verify the effectiveness of GRNN as a surrogate model filtering the bad candidate solutions, we introduce a new indicator, shown as (21). This indicator denotes the degree of similarity between the filtered half with higher approximated fitness values, denoted as $S1$ and the real half with higher exact fitness values, denoted as $S2$.

$$S = \frac{1}{N_i M} \sum_{n=1}^{N_i} m_n \tag{21}$$

where m_n is number of coincident layouts between the $S1$ and $S2$ in the n-th iteration, M is the total number of the layouts in $S1$, and N_i is the number of iterations. Table 8 summarizes the average of the similarity degree derived from five independent runs under each case of WS1. The similarity degree remains higher than 85% when N is smaller than 40 and higher than 79% under all the cases. This high degree of resemblance ensures most better candidate solutions are filtered to drive the evolutionary algorithm. The decrease in the similarity degree also accounts for the slight decrease in power output when N is large.

Despite a slight decrease in power output under complex conditions, ADE-GRNN displays a great advantage over ADE in algorithm efficiency. Table 9 summarizes the average and standard deviation of the runtime of ADE and ADE-GRNN in WS1 and WS2. The symbols "+" and "−" denote the runtime of ADE-GRNN is slower than and faster than ADE, respectively. When N is large, ADE-GRNN can save 40% of runtime compared with ADE. In theory, calculating actual fitness values for half the trial individuals can save 50% of the time. Considering that the first 125 iterations used ADE to accumulate the original samples and the data processing for all individuals both take the extra time, the 40% speed-up in complex situations is very ideal. It is proved that GRNN is a fast learning neural network and reflects the speed advantage of combining GRNN as the surrogate model especially when the objective function has a high computational cost.

5.2.3. The benefit of adaption mechanism of ADE

In order to further illustrate the benefit of adaption mechanism in ADE, the mutation operation and crossover operation are discussed.

(1) Mutation operation

In heuristic algorithms, the difference between individuals is gradually decreasing in the later stage of evolution. The

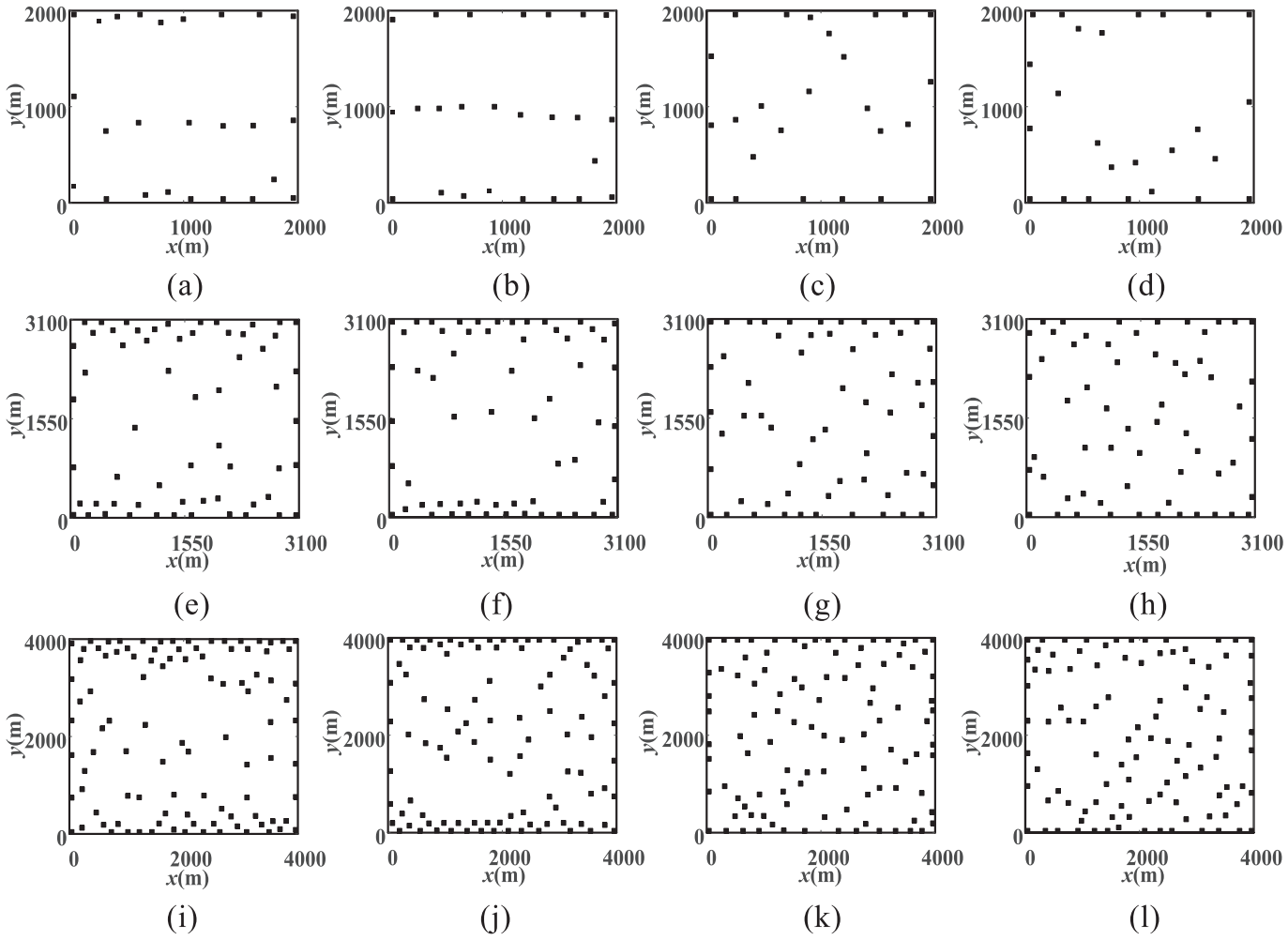

Fig. 8. The best layouts of four compared algorithms with different N in WS2. (a) ADE-GRNN, $N = 25$, (b) ADE, $N = 25$, (c) JADE_m, $N = 25$, (d) CLPSO, $N = 25$, (e) ADE-GRNN, $N = 60$, (f) ADE, $N = 60$, (g) JADE_m, $N = 60$, (h) CLPSO, $N = 60$, (i) ADE-GRNN, $N = 100$, (j) ADE, $N = 100$, (k) JADE_m, $N = 100$, (l) CLPSO, $N = 100$.

Table 8
Degree of similarity between two sets under WS1.

N	15	20	25	30	35	40	60	80	100
Similarity Degree	89.8%	88.8%	88.1%	88.8%	86.9%	85.8%	80.6%	80.2%	79.5%

corresponding searching region of the difference vector of the DE/current-to-pbest/1 strategy becomes small. ADE adopts the idea of making difference vector a movement guidance covering the entire wind farm area. The difference vector in ADE comes from the difference between the individual itself and one of its random disordered individuals in (12). It guarantees the difference vector will never converge to 0 and contains more direction information with an increasing N in ADE.

An example of the evolution progress of μ_{F1} and μ_{F2} in ADE is plotted in Fig. 9. It is obvious that μ_{F1} and μ_{F2} oscillate around 1 in the entire evolution progress. Constrained by the ability of the mutant to shrink, μ_{F1} and μ_{F2} will not always decrease, which allows ADE to balance the local and global search abilities.

Fig. 10 shows an example of evolution of n_j_{rand} in ADE with a blue curve. In Fig. 10, n_j_{rand} converges to 1 quickly and maintains $n\_j_{rand} \approx 1$ throughout the process of evolution. It is indicated that moving only one wind turbine in crossover operation is conducive to the generation of high-quality trial vectors especially in the later evolution.

The $n\_j_{rand} \approx 1$ also reveals the connection between ADE and the algorithm in Ref. [15]. In Ref. [15], only one wind farm layout is encoded as the population and only one wind turbine is replaced in each iteration. Therefore, the proposed algorithm in Refs. [15] can be considered as a special case of ADE.

The mutation strategy of JADE_m is DE/current-to-pbest/1, which emphasizes the local search ability. The experiment result of JADE_m also verifies the trend of n_j_{rand}. In Fig. 10, the n_j_{rand} of JADE_m maintains the same trend.

Based on all the results, it is believed that the wind farm layout optimization is a strong constraint problem. Moving multiple wind turbines simultaneously causes great changes in the total wake loss, which is adverse to individual optimization. This is also the underlying reason why greedy methods which moves only one wind turbine's position perform better than evolutionary algorithms which change all the wind turbines' position in each iteration. It must be noted that such greedy methods are not necessarily

Table 9
Runtime(s) of ADE and ADE-GRNN under WS1 and WS2.

N	Wind Scenario 1		Wind Scenario 2	
	ADE (AVG ± std dev)	ADE-GRNN (AVG ± std dev)	ADE (AVG ± std dev)	ADE-GRNN (AVG ± std dev)
15	625.83 ± 30.56	638.61 ± 36.48 + (2.04%)	606.07 ± 43.53	597.55 ± 17.5 − (1.41%)
20	908.53 ± 9.79	794.53 ± 12.99 − (12.54%)	949.04 ± 24.58	791.57 ± 40.9 − (16.59%)
25	1389.26 ± 49.08	1014.96 ± 15.81 − (26.94%)	1295.51 ± 65.04	1011.5 ± 69.4 − (21.92%)
30	1703.48 ± 33.05	1290.06 ± 26.21 − (24.26%)	1620.25 ± 23.93	1178.35 ± 63.4 − (27.27%)
35	2289.04 ± 163.17	1572.42 ± 17.04 − (31.31%)	2072.04 ± 53.43	1572.49 ± 35.45 − (24.11%)
40	2647.89 ± 234.38	1576.49 ± 25.62 − (40.46%)	2433.98 ± 195.66	1711.48 ± 80.79 − (29.68%)
60	4401.18 ± 28.37	3506.51 ± 385.5 − (20.33%)	5094.15 ± 85.02	3056.82 ± 315.8 − (39.99%)
80	8393.62 ± 900.76	5240.81 ± 505.9 − (37.56%)	8058.72 ± 590.4	4893.35 ± 452.5 − (39.28%)
100	10969.33 ± 402.79	6687.97 ± 439.2 − (39.03%)	12838.85 ± 352.81	7120.05 ± 526.6 − (44.54%)

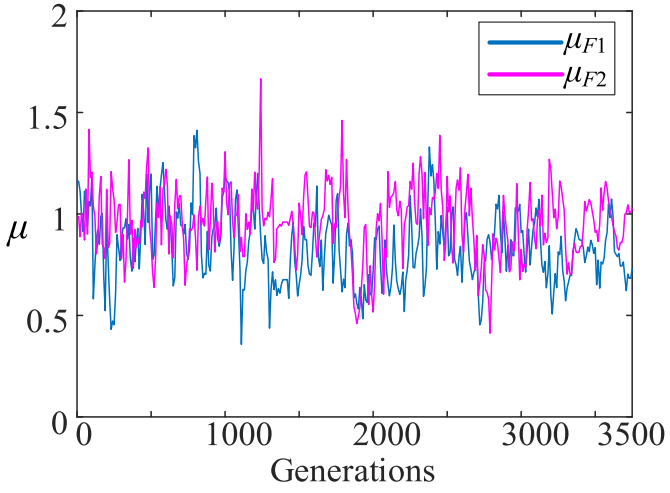

Fig. 9. The evolution example of controlling parameters μ_{F1} and μ_{F2} in ADE.

(2) Crossover mutation

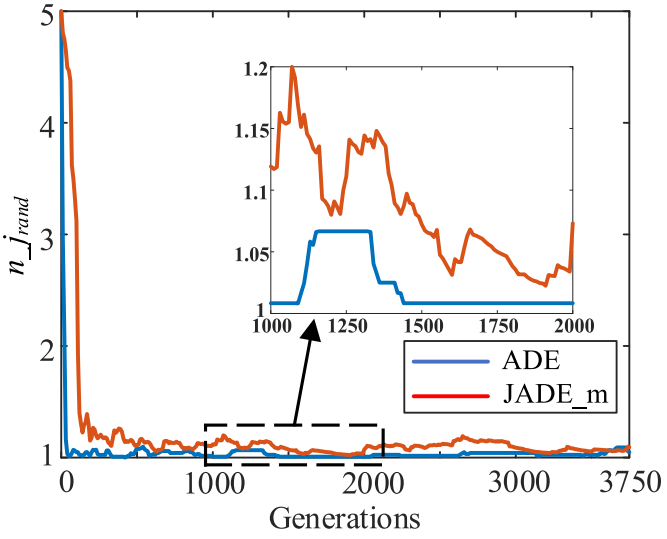

Fig. 10. The evolution example of controlling parameters n_j_{rand} in ADE and JADE_m.

applicable to other optimization problems. Under different constraints, the crossover parameter may converge in other directions. Therefore, the adaption mechanism efficiently reduces the requirement for parameter selection experience.

6. Conclusion

In this paper, a data-driven evolution algorithm, ADE-GRNN, was proposed for optimizing the wind farm layout. The data-driven surrogate model was utilized to approximate the objective function. The surrogate model was trained and updated by GRNN based on the data generated by ADE throughout the whole process. The computational cost was cut down through the bad solutions filtered by the data-driven surrogate model. ADE adopted an adaption mechanism on mutation and crossover operations to improve the evolution success rate. Besides, the adaption mechanism efficiently reduced the requirement for parameter selection.

The ADE-GRNN, compared with ADE, JADE_m and CLPSO, was tested with various numbers of wind turbines under two different wind scenarios. In most cases, the ADE-GRNN showed better or competitive optimization performance than other algorithms in terms of the wind farm power output. The difference among four algorithms reduced under WS2 due to its concentrated wind resource distribution. When the wind resources distributed widely and the number of wind turbines increased, the difficulty of solving the wind farm layout problem greatly increased. In the case of $N = 100$, the average power output of ADE-GRNN improved by about 20% compared with the benchmark algorithm, CLPSO. The layout of ADE-GRNN was also more orderly than JADE_m and CLPSO. It was implied the advantage of ADE-GRNN in solving

complex problems. Although the output of ADE-GRNN decreased less than 1% compared to ADE with $N = 60$ and 100, the algorithm efficiency of ADE-GRNN was dramatically improved. ADE-GRNN could save up to 40% runtime when N was larger than 60, which showed a promising application in real complex optimization problems.

In the future, the complex wind farm topology and wake effect model should be considered. The three-dimension wind farm layout model will be instead of the two-dimension one to be close to the real situation. To obtain a stable surrogate model by the various data-driven algorithm is also the other research direction.

CRediT authorship contribution statement

Huan Long: Conceptualization, Methodology, Writing - review & editing. **Peikun Li:** Data curation, Software, Writing - original draft. **Wei Gu:** Supervision.

Declaration of competing interest

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

Acknowledgement

I wish to acknowledge our funding sources namely National Natural Science Foundation of China, China, with Grant ID: 51807023 and Natural Science Foundation of Jiangsu Province, China

Appendix

Calculation of Power Expectation.

Based on the power curve function (7), the expected power production of the i-th wind turbine is shown as (A1) from the integration of the wind speed and wind direction.

$$E(P_i) = \int_0^{360} g(\theta) \int_0^{\infty} f(v) \frac{k(\theta)}{c_i'(\theta)} \left(\frac{v}{c_i'(\theta)}\right)^{k(\theta)-1} e^{-\left(\frac{v}{c_i'(\theta)}\right)^{k(\theta)}} d\theta dv = \int_0^{360} g(\theta) \int_0^{\infty} f(v) \frac{k(\theta) v^{k(\theta)-1}}{c_i'(\theta)^{k(\theta)}} e^{-\left(\frac{v}{c_i'(\theta)}\right)^{k(\theta)}} d\theta dv \tag{A1}$$

Assume the wind direction is discretized into h bins of equal width; let $\theta_1, \theta_2, \ldots, \theta_{h-1}$ be the dividing points of wind direction with $0^\circ < \theta_1 < \theta_2 < \ldots < \theta_{h-1} < 360^\circ$, $\theta_1 = 0^\circ$ $\theta_h = 360^\circ$. Each bin is associated with a relative frequency $0 \le \zeta_i \le 1$, $i = 0, \ldots, h$. For example, ζ_0 is the frequency of bin $[0^\circ, \theta_1]$, ζ_h is the frequency of bin $[\theta_{h-1}, 360^\circ]$. The frequency ζi can be easily estimated from the wind data. Since $g_\theta((\theta_l + \theta_{l-1})/2)$ can be approximated as ζ_l.

Wind speed is also discretized into s bins of equal width. Let $v_1, v_2, \ldots, v_{s-1}$ be the dividing points of wind direction with $v_{ci} < v_1 < v_2 < \ldots < v_{s-1} < v_r$, $v_0 = v_{ci}$, $v_s = v_r$. The numerical integration method is the midpoint rule. The Eq(A1) can be divided into three part based on the cut-in speed, rated speed and cut-out speed, shown as follows.

- $v > v_{co}$, $v < v_{ci}$ $E(P_i) = 0$ (A2)
- $v_{ci} < v < v_r$

$$\int_{v_{ci}}^{v_r} f(v) \frac{k(\theta)}{c_i'(\theta)} \left(\frac{v}{c_i'(\theta)}\right)^{k(\theta)-1} e^{-\left(\frac{v}{c_i'(\theta)}\right)^{k(\theta)}} dv = \int_{v_{ci}}^{v_r} \frac{e^v}{\alpha + \beta e^v} d\left(1 - e^{-(v/c_i'(\theta))^{k(\theta)}}\right) = \sum_{j=1}^{s} \left(e^{-(v_{j-1}/c_i'(\theta))^{k(\theta)}} - e^{-(v_j/c_i'(\theta))^{k(\theta)}}\right) \frac{e^{(v_{j-1}+v_j)/2}}{\alpha + \beta e^{(v_{j-1}+v_j)/2}} \tag{A3}$$

$$E(P_i) = \sum_{j=1}^{s} \frac{e^{(v_{j-1}+v_j)/2}}{\alpha + \beta e^{(v_{j-1}+v_j)/2}} \int_0^{360} p_\theta(\theta) \left(e^{-(v_{j-1}/c_i'(\theta))^{k(\theta)}} - e^{-(v_j/c_i'(\theta))^{k(\theta)}}\right) d\theta = \sum_{j=1}^{s} \frac{e^{(v_{j-1}+v_j)/2}}{\alpha + \beta e^{(v_{j-1}+v_j)/2}} \sum_{l=1}^{h} \xi_l \left(e^{-(v_{j-1}/c_i'((\theta_l+\theta_{l-1})/2))^{k((\theta_l+\theta_{l-1})/2)}} - e^{-(v_j/c_i'((\theta_l+\theta_{l-1})/2))^{k((\theta_l+\theta_{l-1})/2)}}\right) \tag{A4}$$

- $v_r < v < v_{co}$

$$\int_{v_r}^{v_{co}} f(v) \frac{k(\theta)}{c_i'(\theta)} \left(\frac{v}{c_i'(\theta)}\right)^{k(\theta)-1} e^{-\left(\frac{v}{c_i'(\theta)}\right)^{k(\theta)}} du = \int_{v_r}^{v_{co}} P_r d\left(1 - e^{-(v/c_i'(\theta))^{k(\theta)}}\right) = P_r \left(e^{-(v_r/c_i'(\theta))^{k(\theta)}} - e^{-(v_{co}/c_i'(\theta))^{k(\theta)}}\right) \tag{A5}$$

$$E(P_i) = P_r \int_0^{360} g(\theta)\left(e^{-(v_r/c_i'(\theta))^{k(\theta)}} - e^{-(v_{co}/c_i'(\theta))^{k(\theta)}}\right)d\theta$$
$$= P_r \cdot \sum_{l=1}^{h} \xi_l \left(e^{-(v_r/c_i'((\theta_l+\theta_{l-1})/2))^{k((\theta_l+\theta_{l-1})/2)}} - e^{-(v_{co}/c_i'((\theta_l+\theta_{l-1})/2))^{k((\theta_l+\theta_{l-1})/2)}}\right) \quad \text{(A6)}$$

So,

$$E(P_i) = \sum_{j=1}^{s} \frac{e^{(v_{j-1}+v_j)/2}}{\alpha+\beta e^{(v_{j-1}+v_j)/2}} \sum_{l=1}^{h} \xi_l \left(e^{-(v_{j-1}/c_i'((\theta_l+\theta_{l-1})/2))^{k((\theta_l+\theta_{l-1})/2)}} - e^{-(v_j/c_i'((\theta_l+\theta_{l-1})/2))^{k((\theta_l+\theta_{l-1})/2)}}\right) +$$
$$P_r \cdot \sum_{l=1}^{h} \xi_l \left(e^{-(v_r/c_i'((\theta_l+\theta_{l-1})/2))^{k((\theta_l+\theta_{l-1})/2)}} - e^{-(v_{co}/c_i'((\theta_l+\theta_{l-1})/2))^{k((\theta_l+\theta_{l-1})/2)}}\right)$$
$$= \sum_{l=1}^{h} \xi_l \left\{ \sum_{j=1}^{s} \left(e^{-(v_{j-1}/c_i'((\theta_l+\theta_{l-1})/2))^{k((\theta_l+\theta_{l-1})/2)}} - e^{-(v_j/c_i'((\theta_l+\theta_{l-1})/2))^{k((\theta_l+\theta_{l-1})/2)}}\right) \frac{e^{(v_{j-1}+v_j)/2}}{\alpha+\beta e^{(v_{j-1}+v_j)/2}} \right.$$
$$\left. + P_r \cdot \left(e^{-(v_r/c_i'((\theta_l+\theta_{l-1})/2))^{k((\theta_l+\theta_{l-1})/2)}} - e^{-(v_{co}/c_i'((\theta_l+\theta_{l-1})/2))^{k((\theta_l+\theta_{l-1})/2)}}\right) \right\} \quad \text{(A7)}$$

Once wind speed and wind direction are discretized into small bins. cj(·) and k(·) can be estimated from the historical wind data.

References

[1] Qiu H, Zhao B, Gu W, Bo R. Bi-level two-stage robust optimal scheduling for AC/DC hybrid multi-microgrids. IEEE T SMART GRID 2018;9. 1-1.
[2] GWEC. Global wind report. 2018. 2019, https://gwec.net/wp-content/uploads/2019/04/GWEC-Global-Wind-Report-2018.pdf.
[3] Jensen NO. A note on wind generator interaction. Roskilde, Denmark: Riso National Laboratory; 1983. Technical Report Riso-M-2411.
[4] Brogna R, Feng J, Sørensen JN, Shen WZ, Porté-Agel F. A new wake model and comparison of eight algorithms for layout optimization of wind farms in complex terrain. APPL ENERG; 2019. p. 114189.
[5] Mosetti G, Poloni C, Diviacco B. Optimization of wind turbine positioning in large windfarms by means of a genetic algorithm. J WIND ENG IND AEROD 1994;51:105–16.
[6] Grady SA, Hussaini MY, Abdullah MM. Placement of wind turbines using genetic algorithms. RENEW ENERG 2005;30:259–70.
[7] Pookpunt S, Ongsakul W. Optimal placement of wind turbines within wind farm using binary particle swarm optimization with time-varying acceleration coefficients. RENEW ENERG 2013;55:266–76.
[8] Dazhi J, Chenfeng P, Zhun F, Yan C, Xinye C. Modified binary differential evolution for solving wind farm layout optimization problems. In: IEEE symposium on computational intelligence for engineering solutions. CIES); 2013. p. 23–8. 2013.
[9] Kusiak A, Song Z. Design of wind farm layout for maximum wind energy capture. RENEW ENERG 2010;35:685–94.
[10] Hou P, Hu W, Chen C, Soltani M, Chen Z. Optimization of offshore wind farm layout in restricted zones. Energy 2016;113:487–96.
[11] Yang K, Kwak G, Cho K, Huh J. Wind farm layout optimization for wake effect uniformity. Energy 2019;183:983–95.
[12] Shakoor R, Hassan MY, Raheem A, Wu Y. Wake effect modeling: a review of wind farm layout optimization using Jensen's model. Renew Sustain Energy Rev 2016;58:1048–59.
[13] H. Long, Z. Zhang, A two-echelon wind farm layout planning model, IEEE T SUSTAIN ENERG, 6 863-871.
[14] Wagner M, Day J, Neumann F. A fast and effective local search algorithm for optimizing the placement of wind turbines. RENEW ENERG 2013;51:64–70.
[15] Wang Y, Liu H, Long H, Zhang Z, Yang Y. Differential evolution with a new encoding mechanism for optimizing wind farm layout. IEEE T IND INFORM 2018;14:1040–54.
[16] Jin Y, Sendhoff B. A systems approach to evolutionary multiobjective structural optimization and beyond. IEEE Comput Intell Mag 2009;4:62–76.
[17] Jin Y, Wang H, Chugh T, Guo D, Miettinen K. Data-driven evolutionary optimization: an overview and case studies. IEEE T EVOLUT COMPUT; 2019.
[18] Specht DF. A general regression neural network. IEEE Trans Neural Network 1991;2(6):568–76.
[19] K. Price, R.M. Storn, J.A. Lampinen, Differential evolution: a practical approach to global optimization, Springer Science & Business Media2006.
[20] Zhang J, Sanderson AC. JADE: adaptive differential evolution with optional external archive. IEEE T EVOLUT COMPUT 2009;13:945–58.
[21] Lackner MA, Elkinton CN. An analytical framework for offshore wind farm layout optimization. Wind Eng 2007;31:17–31.
[22] Stroock DW. A concise introduction to the theory of integration. Springer Science & Business Media; 1998.
[23] Storn R, Price K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 1997;11:341–59.
[24] Hu R, Wen S, Zeng Z, Huang T. A short-term power load forecasting model based on the generalized regression neural network with decreasing step fruit fly optimization algorithm. Neurocomputing 2017;221:24–31.
[25] Bendu H, Deepak BBVL, Murugan S. Application of GRNN for the prediction of performance and exhaust emissions in HCCI engine using ethanol. ENERG CONVERS MANAGE 2016;122:165–73.
[26] Wu J, Liu C. An expert system for fault diagnosis in internal combustion engines using wavelet packet transform and neural network. Expert Syst Appl 2009;36:4278–86.
[27] Antanasijević D, Pocajt V, Ristić M, Perić-Grujić A. Modeling of energy consumption and related GHG (greenhouse gas) intensity and emissions in Europe using general regression neural networks. Energy 2015;84:816–24.
[28] Ni YQ, Li M. Wind pressure data reconstruction using neural network techniques: a comparison between BPNN and GRNN. Measurement 2016;88:468–76.
[29] Gao S, Tian J, Fang W, Yang B, Qiang Y. The study of GRNN for wind speed forecasting based on Markov chain. International Conference on Modeling; 2015.
[30] Liang JJ, Qin AK, Suganthan PN, Baskar S. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE T EVOLUT COMPUT 2006;10:281–95.