Differential Evolution with A New Encoding Mechanism for Optimizing Wind Farm Layout

Yong Wang, *Member, IEEE,* Hao Liu, Huan Long, *Student Member, IEEE,* Zijun Zhang, *Member, IEEE,* and Shengxiang Yang, *Senior Member, IEEE*

***Abstract*—This paper presents a differential evolution algorithm with a new encoding mechanism for efficiently solving the optimal layout of the wind farm, with the aim of maximizing the power output. In the modeling of the wind farm, the wake effects among different wind turbines are considered and the Weibull distribution is employed to estimate the wind speed distribution. In the process of evolution, a new encoding mechanism for the locations of wind turbines is designed based on the characteristics of the wind farm layout. This encoding mechanism is the first attempt to treat the location of each wind turbine as an individual. As a result, the whole population represents a layout. Compared with the traditional encoding, the advantages of this encoding mechanism are twofold: 1) the dimension of the search space is reduced to two, and 2) a crucial parameter (i.e., the population size) is eliminated. In addition, differential evolution serves as the search engine and the caching technique is adopted to enhance the computational efficiency. The comparative analysis between the proposed method and seven other state-of-the-art methods is conducted based on two wind scenarios. The experimental results indicate that the proposed method is able to obtain the best overall performance, in terms of the power output and execution time.**

***Index Terms*—Wind farm layout; optimization; wake effect; encoding mechanism; differential evolution.**

I. Introduction

WIND energy plays an important role in the field of renewable energy worldwide [1], [2]. The wind farm layout is a key factor which determines the power output of a wind farm during its life cycle. A general target of wind farm layout is to maximize the total power output through optimizing the locations of wind turbines. Note that there exist wake effects among wind turbines in the process of wind energy generation. The wakes produced by the upstream wind turbines will impact the downstream ones, causing discount of energy generation of the wind farm. Consequently, it is vital to investigate the wind farm layout to reduce and even avoid the wake effect.

The existing wind farm layout models can be classified into two categories: the grid-based model and the coordinate-based model. In the grid-based model, the wind farm is divided into a set of square cells and each cell center is a potential location for placing a wind turbine. With respect to the coordinate-based model, a wind turbine is flexibly located in the wind farm and characterized by a two-dimensional coordinate. Since both the grid-based and coordinate-based models cannot be analytically solved, different kinds of heuristic methods have been proposed for optimizing the wind farm layout.

Evolutionary algorithms (EAs), which are a kind of population-based heuristic methods, have been broadly applied to the wind farm layout. For instance, Mosetti *et al.* [3] formulated the wind farm layout problem as the grid-based model and introduced genetic algorithm to optimize wind turbine locations. Based on the same model, Grady *et al.* [4] achieved better results through increasing population size as well as generations of genetic algorithm. In [5], a genetic algorithm with an improved crossover operation is presented to optimize the profits of a wind farm without considering the wake effect. Pookpunt *et al.* [6] investigated a binary particle swarm optimization with time-varying acceleration coefficients to solve the grid-based model. In addition, Jiang *et al.* [7] tackled the grid-based model by designing a binary differential evolution based on smoothing operator. In [8], Kusiak and Song developed the coordinate-based model and solved it by the SPEA algorithm. In subsequent studies, the coordinate-based model has been solved by particle swarm optimization [9], seeding evolutionary algorithm [10], ant colony optimization [11], covariance matrix adaptation evolution strategy (CMA-ES) [12], and differential evolution [13], showing promising results. The aforementioned methods have a common feature: each individual in the population represents an entire wind farm layout. Because of this feature, evolutionary operators (such as mutation and crossover) can be easily implemented on the individuals in the population. However, this kind of methods exhibits low efficiency and needs tremendous computational workload to find the optimal layout. Moreover, since the dimension of the search space is relevant to the number of wind turbines, it may suffer from the curse of dimensionality with the drastic increase of the

Manuscript received June 4, 2017; accepted July 11, 2017. This work was supported in part by the National Natural Science Foundation of China under Grants 61673397 and 61673331, in part by the EU Horizon 2020 Marie Sklodowska-Curie Individual Fellowships (Project ID: 661327), in part by the Engineering and Physical Sciences Research Council of UK under Grant EP/K001310/1, and in part by the Hunan Provincial Natural Science Fund for Distinguished Young Scholars (Grant No. 2016JJ1018).

Y. Wang is with the School of Information Science and Engineering, Central South University, Changsha 410083, China, and also with the Centre for Computational Intelligence (CCI), School of Computer Science and Informatics, De Montfort University, Leicester LE1 9BH, UK. (Email: ywang@csu.edu.cn)

H. Liu is with the School of Information Science and Engineering, Central South University, Changsha 410083, China. (Email: haoliu@csu.edu.cn)

H. Long is with the School of Electrical Engineering, Southeast University, Nanjing 210096, China. (e-mail: hlong5-c@my.cityu.edu.hk)

Z. Zhang is with the Department of Systems Engineering and Engineering Management, City University of Hong Kong, Hong Kong. (e-mail: zijzhang@cityu.edu.hk)

S. Yang is with the Centre for Computational Intelligence (CCI), School of Computer Science and Informatics, De Montfort University, Leicester LE1 9BH, UK, and also with the College of Information Engineering, Xiangtan University, Xiangtan 411105, China (Email: syang@dmu.ac.uk)

number of wind turbines.

Greedy methods, as another kind of heuristic methods, have also attracted a lot of attention in the wind farm layout. In the greedy methods, a single initial layout is produced and subsequently optimized by moving one wind turbine in each iteration. To speed up the evaluation of power output, the caching technique is widely employed in the greedy methods [14]–[16]. Ozturk *et al.* [17] developed a greedy improvement methodology and designed the adding, removing, and moving operators. Saavedra-Moreno *et al.* [10] exploited a greedy algorithm to produce local optimal solutions and considered them as the initial population of genetic algorithm. Zhang *et al.* [18] revealed the submodular property of the wind turbine locating problem and suggested a lazy greedy algorithm to accelerate the process of searching for a local optimal solution. Markus *et al.* [14] incorporated domain-specific characteristics into a local search to produce a new layout. The method in [14] obtains better results while costing less computational time than an EA, i.e., CMA-ES. In [19], a bionic algorithm is proposed, in which a wind turbine is located and relocated where its own power output can be increased. Yang *et al.* [20] proposed a random search algorithm and improved it by adding some adaptive mechanisms in their later work [15]. In [16], a greedy algorithm with repeated adjustment is applied to optimize wind turbine locations. In contrast to EAs which maintain a population of layouts, the greedy methods only optimize one layout. As a consequence, this kind of method is more efficient in searching for the optimal layout. However, its global search ability is limited due to the fact that it usually uses random search or local search to relocate one wind turbine in each iteration.

Recognizing that both EAs and greedy methods have their advantages and shortcomings, a question which arises naturally is whether we can integrate the advantages of these two kinds of methods, achieving the balance between effectiveness and efficiency. Motivated by the above consideration, this paper presents a new encoding mechanism and exploits differential evolution (DE), a very population EA paradigm, as the search engine. By combining this new encoding mechanism with DE, a simple yet generic method called DEEM is presented to solve the coordinate-based model. Herein, the coordinate-based model is employed because it allows more flexible distribution of wind turbines compared with the grid-based model. To the best of our knowledge, the encoding mechanism in DEEM is the first attempt to treat the location of each wind turbine as an individual. Based on this encoding mechanism, the whole population just represents a layout. Afterward, the mutation and crossover operators of DE are implemented on each individual in the parent population to produce an offspring population. At each generation, each offspring is used to randomly replace an individual in the parent population to form a new layout. If the new layout has a better power output, this update is successful and acceptable. Furthermore, the caching technique is used to accelerate the wind power evaluation process. It is shown empirically that DEEM outperforms seven other state-of-the-art methods in terms of the power output and computational time.

The rest of this paper is organized as follows. Section II formulates the wind farm layout model. Section III gives a brief introduction of DE. Section IV elaborates the proposed DEEM. Comparative studies and discussions are conducted in Section V and Section VI, respectively. Finally, Section VII concludes this paper.

II. Problem Formulation

In this section, the power curve model aims to compute the power output of a wind turbine according to a given wind speed and the wake effect model is used to quantify the wake effect. Then, these models are combined by performing numerical integration. Finally, the wind farm layout model is formulated by considering some constraints.

A. Assumptions

Let the number of wind turbines be equal to N. To formulate a general wind farm layout model, several assumptions are considered below.

A1. All the wind turbines and their power curve functions are identical.

A2. The layout of a wind farm is based on a two-dimensional coordinate system (i.e., x axis and y axis), and the search space is $S = [\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}]$, where $\underline{x}$ and $\overline{x}$ are the lower and upper bounds of x, respectively, and $\underline{y}$ and $\overline{y}$ are the lower and upper bounds of y, respectively.

A3. For wind turbine i and wind direction θ, wind speed v follows the Weibull distribution, expressed as:

$$p(v, c_i(\theta), k_i(\theta)) = \frac{k_i(\theta)}{c_i(\theta)}\left(\frac{v}{c_i(\theta)}\right)^{k_i(\theta)-1} \times e^{-\left(\frac{v}{c_i(\theta)}\right)^{k_i(\theta)}}, \; 0^\circ \leq \theta < 360^\circ \tag{1}$$

where $k_i(\theta)$ and $c_i(\theta)$ are the shape parameter and the scale parameter, respectively, and they are continuous functions of wind direction θ.

A4. There exists a minimum distance between any two wind turbines to ensure safety, which is set to five times of the rotor radius, i.e., $5R$.

A5. A wind turbine turns its nacelle to keep the rotor plane perpendicular to wind direction θ.

B. Power Curve Model

A power curve function (denoted as $f(v)$) can be used to describe the relationship between the power output of wind turbine i and wind speed v [21]:

$$P_i = f(v) = \begin{cases} 0, & v \geq v_{co}, v < v_{ci} \\ \frac{e^v}{\alpha+\beta e^v}, & v_{ci} \leq v < v_r \\ P_r, & v_r \leq v < v_{co} \end{cases} \tag{2}$$

where P_i is the power output of wind turbine i, and α and β are constants. As shown in (2), when v is smaller than the cut-in speed v_{ci}, no power is extracted. When v is larger than the cut-out speed v_{co}, the wind turbine shuts down to protect itself. If v ranges from the rated speed v_r to v_{co}, the wind turbine control system will keep the rated power output P_r.

C. Wake Effect Model

Wake effect is the main factor physically impacting the wind power output of a wind farm. When the free stream wind passes through a wind turbine, the kinetic energy of wind is reduced and a wake behind the wind turbine occurs. As a result, the energy extracted by the downstream wind turbine in the wake will diminish. Considering simplicity and rationality, the Jensen's wake model [22] is adopted to describe this phenomenon.

Suppose that wind turbines i and j are located at (x_i, y_i) and (x_j, y_j) in the wind farm, respectively. If wind turbine i is affected by the wake of wind turbine j, the velocity deficit of wind turbine i caused by the wake of wind turbine j is denoted as $VD_{j,i}$ and calculated by (3)-(5):

$$a = 0.5(1 - \sqrt{1 - C_T}), \kappa = 0.5/\ln(z/z_0) \tag{3}$$

$$d_{j,i} = |(x_j - x_i)\cos\theta + (y_j - y_i)\sin\theta| \tag{4}$$

$$VD_{j,i} = 1 - v_{dn}/v_{up} = 2a/(1 + \kappa d_{j,i}/R)^2 \tag{5}$$

where a is the axial induction factor, C_T is the fixed thrust coefficient [23], z is the tower height of a wind turbine, z_0 is the ground surface roughness, v_{dn} is the wind speed at downstream wind turbine i, and v_{up} is the wind speed at upstream wind turbine j.

Afterward, the total velocity deficit VD_i of wind turbine i caused by the wakes of all the other wind turbines can be derived as follows:

$$VD_i = \sqrt{\sum\nolimits_{j=1, j\neq i}^{N} (VD_{j,i})^2}, i = 1, 2, ..., N \tag{6}$$

It is worth noting that the scale parameter $c_i(\theta)$ of the Weibull distribution is influenced by the wake effect [8] and the updated $c_i(\theta)$ (denoted as $c'_i(\theta)$) is computed by (7)

$$c'_i(\theta) = c_i(\theta) \times (1 - VD_i),\ i = 1, 2, ..., N \tag{7}$$

D. Numerical Integration of Expected Power Output

After calculating the integral of the product of (1) and (2) with respect to wind speed v and wind direction θ, the expected power output of wind turbine i is calculated by (8):

$$\begin{aligned} E(P_i) = \int_{0^\circ}^{360^\circ} p(\theta) \int_0^\infty f(v) \frac{k_i(\theta)}{c'_i(\theta)} \left(\frac{v}{c'_i(\theta)}\right)^{k_i(\theta)-1} \times \\ e^{-\left(\frac{v}{c'_i(\theta)}\right)^{k_i(\theta)}} dv d\theta \end{aligned} \tag{8}$$

where $p(\theta)$ is the probability density function of θ.

Due to the fact that $f(v)$ is a piecewise function, the integral over v in (8) can be divided into four parts according to the intervals $[0, v_{ci})$, $[v_{ci}, v_r)$, $[v_r, v_{co})$, and $[v_{co}, +\infty)$. In the case of $[0, v_{ci})$ and $[v_{co}, +\infty)$, (8) is equal to 0 because $f(v) = 0$. Regarding $[v_r, v_{co})$, the integral over v is equal to $P_r \times (e^{-(v_r/c'_i(\theta))^{k_i(\theta)}} - e^{-(v_{co}/c'_i(\theta))^{k_i(\theta)}})$. Since it is challenging to analytically obtain the integral in $[v_{ci}, v_r)$, a numerical integration technique, i.e., Riemann sum [24], is adopted. Under this condition, wind speed v is quantized into s intervals with the same width: $[v_0, v_1)$, $[v_1, v_2)$, ..., $[v_{s-1}, v_s)$, where $v_0 = v_{ci}$ and $v_s = v_r$. Similarly, wind direction θ is quantized into h intervals with the same width: $[\theta_0, \theta_1)$, $[\theta_1, \theta_2)$, ..., $[\theta_{h-1}, \theta_h)$, where $\theta_0 = 0^\circ$ and $\theta_h = 360^\circ$. After these processes, the expected power output of wind turbine i can be obtained in the following discrete form [8]:

$$\begin{aligned} E(P_i) = \sum\nolimits_{n=1}^{h} \xi_n \Bigg\{ & P_r \times \Big(e^{-(v_r/c'_i((\theta_{n-1}+\theta_n)/2))^{k_i((\theta_{n-1}+\theta_n)/2)}} \\ & - e^{-(v_{co}/c'_i((\theta_{n-1}+\theta_n)/2))^{k_i((\theta_{n-1}+\theta_n)/2)}} \Big) \\ & + \sum\nolimits_{j=1}^{s} \Big(e^{-(v_{j-1}/c'_i((\theta_{n-1}+\theta_n)/2))^{k_i((\theta_{n-1}+\theta_n)/2)}} \\ & - e^{-(v_j/c'_i((\theta_{n-1}+\theta_n)/2))^{k_i((\theta_{n-1}+\theta_n)/2)}} \Big) \times \\ & \frac{e^{(v_{j-1}+v_j)/2}}{\alpha + \beta e^{(v_{j-1}+v_j)/2}} \Bigg\} \end{aligned} \tag{9}$$

where ξ_n is the frequency of the interval $[\theta_{n-1}, \theta_n)$.

E. Wind Farm Layout Model

In this paper, the objective is to find the optimal layout of all the wind turbines to maximize the power output of a wind farm, which is expressed by (10). Since the coordinate-based model is used, each wind turbine can be located anywhere in the wind farm as long as the constraints are satisfied. In (10), we mainly take three constraints into account. The first two constraints enable a wind turbine to lie within the wind farm. Additionally, the third constraint guarantees the distance between wind turbine i and any other wind turbine not shorter than $5R$.

$$\begin{cases} \text{maximize: } P = \sum\nolimits_{i=1}^{N} E(P_i) \\ \text{subject to: } \underline{x} + R \leq x_i \leq \overline{x} - R, \\ \qquad \underline{y} + R \leq y_i \leq \overline{y} - R, \\ \qquad \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \geq 5R, \\ \qquad j = 1, 2, ..., N \text{ and } j \neq i \end{cases} \tag{10}$$

III. Differential Evolution

Differential evolution (DE) is a population-based optimizer [25]. As a very popular paradigm of EAs, DE has been widely applied to solve a variety of optimization problems. At the beginning of evolution, DE randomly samples NP individuals from the search space, each of which is also called a target vector:

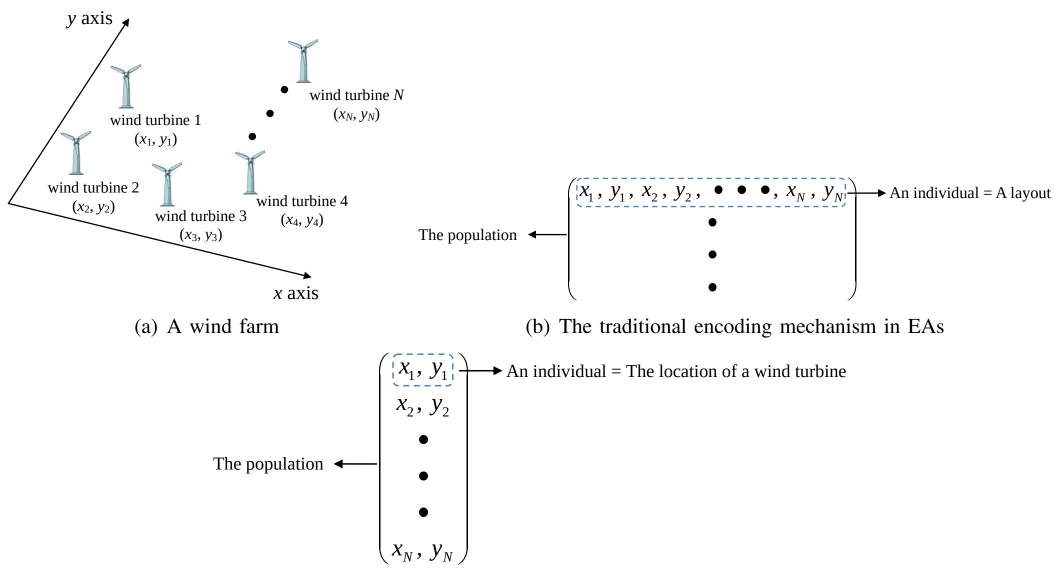
$$\vec{x}_i = (x_{i,1}, x_{i,2}, ..., x_{i,D}),\ i = 1, 2, ..., NP \tag{11}$$

where D is the dimension of the target vector. Afterward, DE implements three main operators, i.e., mutation, crossover, and selection to evolve the population.

Mutation: The mutation operator generates a mutant vector $\vec{v}_i = (v_{i,1}, v_{i,2}, ..., v_{i,D})$ for each target vector $\vec{x}_i$ via (12):

$$\vec{v}_i = \vec{x}_{r1} + F \times (\vec{x}_{r2} - \vec{x}_{r3}),\ i = 1, 2, ..., NP \tag{12}$$

where $r1$, $r2$, and $r3$ are three mutually distinct integers randomly chosen from $[1, NP]$ and also different from i, F is the scaling factor, and $(\vec{x}_{r2} - \vec{x}_{r3})$ is the difference vector.



(a) A wind farm (b) The traditional encoding mechanism in EAs

(c) The proposed encoding mechanism in this paper

Fig. 1. Difference between the traditional encoding mechanism in EAs and the proposed encoding mechanism in this paper

Crossover: The binomial crossover operator is implemented on each pair of $\vec{x}_i$ and $\vec{v}_i$ to produce a trial vector $\vec{u}_i = (u_{i,1}, u_{i,2}, ..., u_{i,D})$ via (13):

$$u_{i,j} = \begin{cases} v_{i,j}, & if\ rand_j < CR\ or\ j = j_{rand} \\ x_{i,j}, & otherwise \end{cases} \tag{13}$$

where $i = 1, 2, ..., NP$, $j = 1, 2, ..., D$, $rand_j$ is a uniformly distributed random number on the interval $[0, 1]$, j_{rand} is a randomly chosen integer between 1 and D, and $CR \in [0, 1]$ is the so-called crossover control parameter.

Selection: Considering a maximization problem, the target vector $\vec{x}_i$ is compared with its trial vector $\vec{u}_i$ based on the objective function $f(\cdot)$, and the better one will survive into the next generation:

$$\vec{x}_i = \begin{cases} \vec{u}_i, & if\ f(\vec{u}_i) \geq f(\vec{x}_i) \\ \vec{x}_i, & otherwise \end{cases}, \ i = 1, 2, ..., NP \tag{14}$$

IV. Differential Evolution with A New Encoding Mechanism for Optimizing Wind Farm Layout

A. Motivation

When optimizing the wind farm layout, the main feature of greedy methods is that only one layout is considered and only one wind turbine in the layout is moved by some strategies similar to random search or local search in each iteration. As a result, the evaluation of the layout can be sped up by utilizing the caching technique. Nevertheless, the above feature also results in the poor global search ability of greedy methods.

In contrast, EAs work with a population of candidate solutions and are well suited for global search. When EAs are applied to optimize the wind farm layout, each individual usually represents an entire layout and each dimension of an individual denotes a coordinate of a wind turbine. After implementing evolutionary operators on the individuals, maybe many wind turbines rather than just one wind turbine are updated. Therefore, it is hard to use the caching technique to accelerate the evaluation of a layout under this condition, which leads to consuming a great deal of computational time.

In the wind farm layout, it is clear that the location of a wind turbine is determined by a two-dimensional coordinate system (i.e., x axis and y axis) and each dimension of all the wind turbines has the same search region (i.e., $[\underline{x}, \bar{x}]$ or $[\underline{y}, \bar{y}]$). This property motivates us to design a new encoding mechanism, in which each wind turbine is considered to be an individual and all the wind turbines form a population. Fig. 1 depicts the difference between the traditional encoding mechanism in EAs and the proposed encoding mechanism in this paper. As shown in Fig. 1, for the proposed encoding mechanism, each individual contains two dimensions and the population is an $N \times 2$ matrix. However, for the traditional encoding mechanism in EAs, each individual contains $2N$ dimensions and the population is an $NP \times 2N$ matrix, where NP is the user-defined population size.

The proposed encoding mechanism has the following characteristics:

- A population represents a layout. Under this condition, if only one wind turbine (i.e., one individual) is moved in each iteration, the caching technique can be applied.
- Each individual can be updated by evolutionary operators. In this manner, the global search ability can be strengthened.
- The population size does not need to be predefined since it is equal to the number of wind turbines, i.e., N.

Therefore, the advantages of greedy methods and EAs can be combined effectively by this encoding mechanism.

B. DEEM

Due to its simple structure and ease to implement, DE serves as the search engine in this paper. By combining DE with this new encoding mechanism, we propose a simple yet generic

Algorithm 1 Initialization

```
1: $k = 0$;
2: Put a wind turbine into the wind farm randomly;
3: for $i = 2$ to $N$ do
4:     If $k > 200$, delete all the wind turbines in the wind farm and go to Step 1;
       otherwise, put wind turbine $i$ into the wind farm randomly;
5:     If wind turbine $i$ cannot satisfy the third constraint in (10), then $k = k + 1$ and
       go to Step 4;
6:     $k = 0$;
7: end for
8: Output the initial layout
```

Algorithm 2 The Framework of DEEM

```
1: Generate an initial population $\boldsymbol{P}$ and evaluate the wind power output of $\boldsymbol{P}$ based
   on (10);
2: $FEs = 0$; // $FEs$ denotes the number of fitness evaluations of the wind power
   output
3: while $FEs < MaxFEs$ do
4:     Implement the mutation and crossover of DE in (12) and (13) on $\boldsymbol{P}$ to generate
       an offspring population $\boldsymbol{Q}$;
5:     for $i = 1$ to $N$ do
6:         Utilize the $i$th offspring in $\boldsymbol{Q}$ to replace a randomly selected individual in $\boldsymbol{P}$
           and denote the updated $\boldsymbol{P}$ as $\boldsymbol{S}$;
7:         if $\boldsymbol{S}$ satisfies the constraints in (10) then
8:             Evaluate the wind power output of $\boldsymbol{S}$ based on (10);
9:             $FEs = FEs + 1$;
10:            if $\boldsymbol{S}$ offers higher wind power output than $\boldsymbol{P}$ then
11:                $\boldsymbol{P} = \boldsymbol{S}$;
12:            end if
13:        end if
14:    end for
15: end while
16: Output $\boldsymbol{P}$
```

method, called DEEM, to solve the wind farm layout model in (10). In DEEM, each individual (i.e., the location of a wind turbine) is denoted as (x_i, y_i) $(i \in \{1, 2, \ldots, N\})$ and the population is denoted as $\boldsymbol{P} = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$.

In the initialization, a wind turbine is firstly put into the wind farm randomly. Next, the second wind turbine is also put into the wind farm randomly and the third constraint in (10) is checked. If the second wind turbine satisfies this constraint, then the allocation is successful; otherwise, the location of the second turbine will be regenerated. Afterward, the above process will be executed on the third wind turbine and so forth. At last, all the wind turbines are located at the wind farm and an initial layout (i.e., an initial population $\boldsymbol{P}$) is produced. Note that if the number of relocation for a wind turbine is more than 200, the initialization will restart. Algorithm 1 shows the implementation of the initialization. It is necessary to emphasize that this initialization process is used in all the compared algorithms in this paper.

During the evolution, an offspring population $\boldsymbol{Q}$ is firstly generated by implementing the mutation and crossover operators of DE in (12) and (13) on $\boldsymbol{P}$. Afterward, the first individual in $\boldsymbol{Q}$ is used to replace a randomly selected individual in $\boldsymbol{P}$. As a result, we obtain an updated $\boldsymbol{P}$, denoted as $\boldsymbol{S}$. Obviously, $\boldsymbol{S}$ represents a new layout. If $\boldsymbol{S}$ satisfies the constraints in (10) and the wind power output of $\boldsymbol{S}$ is higher than that of $\boldsymbol{P}$, $\boldsymbol{P}$ is replaced with $\boldsymbol{S}$; otherwise, $\boldsymbol{P}$ is kept unchanged. Subsequently, the above process is implemented on the remaining individuals in $\boldsymbol{Q}$ one by one. When the maximum number of fitness evaluations (denoted as $MaxFEs$) is reached, DEEM will stop. The framework of DEEM is given in Algorithm 2.

As mentioned previously, the main feature of greedy methods is that only one layout is considered and only one wind turbine is moved in each iteration. As a result, the evaluation of the layout can be sped up by the caching technique. For DEEM, in each updating of $\boldsymbol{P}$, only one offspring in $\boldsymbol{Q}$ is used to randomly replace an individual in $\boldsymbol{P}$. Thus, similar to greedy methods, the evaluation of the wind power output in DEEM is also efficient. On the other hand, DEEM can benefit from the global search ability of DE by taking advantages of the mutation and crossover of DE to yield the offspring population $\boldsymbol{Q}$. Therefore, DEEM is capable of achieving the balance between effectiveness and efficiency. It is evident from Algorithm 2 that the implementation of DEEM is quite simple. Moreover, DEEM eliminates a user-specified parameter (i.e., the population size) and only contains two control parameters: F in (12) and CR in (13).

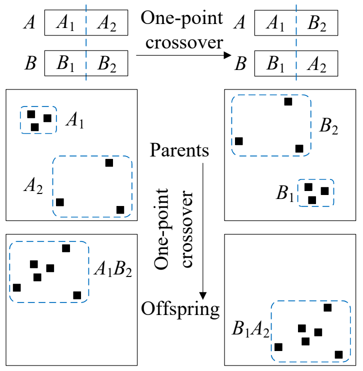


Fig. 2. Explanation of how parents produce bad offspring in the traditional encoding of EAs.

C. Principle Analysis

In the following, we will analyze the principles of EAs, greedy methods, and DEEM.

- In current EAs, an individual usually represents an entire wind farm layout. With respect to such traditional encoding mechanism, the probability that the offspring created by the evolutionary operator are better than the parents might be very low. Fig. 2 gives an example. Suppose that: 1) a layout contains six wind turbines, 2) there are two parents A and B, and 3) the crossover site splits both A and B into two segments (i.e., A_1 and A_2, and B_1 and B_2). As shown in Fig. 2, after implementing the one-point crossover on A and B, the offspring have lower power output since the wind turbines in the offspring cluster in a small part of the search space. The above phenomenon can be attributed to the random selection of the crossover site in the one-point crossover, which results in the information exchange between two parents being quite random. In contrast, in DEEM only one wind turbine is moved to produce a new layout so DEEM updates the layout in a more stable manner.
- In the previous work, when applying EAs to optimize the wind farm layout, the dimension of the search space is dependent mainly on the number of wind turbines and equal to $2N$. Consequently, EAs will suffer from the curse of dimensionality with the drastic increase

of the number of wind turbines. However, in DEEM, each individual contains two decision variables and the dimension of the search space is thus equal to two, regardless of the number of wind turbines. Clearly, it is much easier for DEEM to search for the optimal layout, owing to the low-dimensional search space.

- Existing greedy methods usually move a wind turbine randomly or merely according to the locations of neighbor wind turbines in each updating, which is similar to random search or local search, respectively. Thus, the global search ability of existing greedy methods is limited. Similar to greedy methods, DEEM also moves one wind turbine to a new location in each updating. Nevertheless, in DEEM the new location is generated based on the mutation and crossover operators of DE. Under this condition, DEEM has the potential to utilize the information of all the other wind turbines when moving a specific wind turbine. As a consequence, DEEM exhibits better global search ability.

D. Evaluation Acceleration

Evaluating the power output of a layout is an important step in the optimization of the wind farm layout. It is noteworthy that this step is time-consuming and occupies most of the computational time of the whole procedure. Fortunately, we can exploit the caching technique [14] to reduce the computational time thanks to the encoding mechanism of DEEM. The caching technique accelerates the evaluation by simplifying the computation of the velocity deficit. When wind turbine j is moved, its velocity deficit is calculated as (6). For each unmoved wind turbine, we only need to reconsider its velocity deficit induced by wind turbine j.

As far as the commonly used evaluation method is concerned, the velocity deficit needs to be computed between any two wind turbines. The computational time complexity is thus $O(N^2)$. However, the caching technique only requires $2(N-1)$ checks of the velocity deficit and the computational time complexity is $O(N)$.

Remark 1: There are two major differences between DEEM and the general DE introduced in Section III:

- The encoding mechanism as explained in Fig. 1.
- The selection operator: After evaluating the trial vector and its target vector by (10), the general DE adopts a one-to-one selection between them and the better one will survive into the next generation. However, in DEEM each trial vector will firstly randomly replace a target vector. Afterward, the updated population is evaluated by (10). If the updated population is better than the previous one, then the replacement will occur.

V. Computational Studies

In order to verify the effectiveness of DEEM, it was compared with an outstanding greedy method (i.e., turbine distribution algorithm (TDA) [14]) and five state-of-the-art EAs: CMA-ES [12], two variants of particle swarm optimization (i.e., MSO [26] and CLPSO [27]), and two variants of DE (i.e., JADE [28] and SHADE [29]). In TDA, a wind turbine is moved by a displacement vector. The length and direction of the displacement vector are computed according to the locations of two nearest wind turbines. CMA-ES is a well-known evolution strategy proposed by Hansen and Ostermeier [30]. It searches for the optimal solution by making use of covariance matrix adaptation and has been applied to wind farm layout in [12]. In MSO, the population is divided into a number of sub-swarms and these sub-swarms are regrouped frequently to achieve better diversity of the population. CLPSO is a comprehensive learning particle swarm optimizer, in which all other particles' historical best information is used to update a particle's velocity. JADE implements a new mutation strategy "DE/current-to-$p$best" with optional external archive and updates control parameters in an adaptive manner. SHADE is an enhanced JADE, which uses a history based parameter adaptation scheme.

TABLE I
Parameter Settings of Wind Turbines

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 turbines (m/s)	25
α	The parameter of power curve function	6.0268
β	The parameter of power curve function	0.0007

TABLE II
Side Lengths of A Wind Farm with Different Number of Wind Turbines

N	15	20	25	30	35	40	60	80	100
Side Length (m)	2000	2000	2000	2200	2400	2600	3100	3600	4000

Two wind scenarios with different number of wind turbines were used to compare the performance of TDA, CMA-ES, MSO, CLPSO, JADE, SHADE, and DEEM. In this paper, we considered the following number of wind turbines: N = 15, 20, 25, 30, 35, 40, 60, 80, and 100. For each compared algorithm, 30 independent runs were executed on each scenario with a specified number of wind turbines. To test the statistical significance between DEEM and each competitor, Wilcoxon's rank sum test at a 0.05 significance level was applied. In all the tables of this section, "$+$", "$-$", and "$\approx$" denotes the performance of DEEM is better than, worse than, and similar to that of its competitor, respectively. In addition, "Mean PO" and "Std Dev" indicate the average and standard deviation of the power output (kW) in 30 runs, respectively, and percentages in parentheses denote the improvement rates of DEEM against other algorithms.

A. Parameter Settings

For the wind farm layout model, GE1.5-77 wind turbine was considered and its detailed parameters are shown in Table I. The number of wind direction intervals h and wind speed intervals s was set to 24 and 36, respectively. The shape of the wind farm was set to a square area with varying side lengths,

TABLE III
PARAMETER SETTINGS OF THE SEVEN COMPARED ALGORITHMS

Algorithm	Parameter Settings
TDA	$\sigma_{dis} = 500, \sigma_{dir} = \pi/6$
CMA-ES	$\sigma = 15$, $\lambda = 4 + \lfloor 3ln(2N) \rfloor$, $\mu = \lfloor \lambda/2 \rfloor$;
MSO	$s_1 = 10, s_2 = 3$, ω: linear decreasing from 0.9 to 0.2, $c_1 = 2$, $c_2 = 2$
CLPSO	$NP = 60$, ω: linear decreasing from 0.9 to 0.4, $c = 1.49445$
JADE	$NP = 100$
SHADE	$NP = 100$
DEEM	$F = 0.9, CR = 0.9$

TABLE IV
WIND SCENARIO 1

n	θ_{n-1}	θ_n	$k_i(\theta)$	$c_i(\theta)$	ξ_n	n	θ_{n-1}	θ_n	$k_i(\theta)$	$c_i(\theta)$	ξ_n
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	300°	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

which are relevant to the number of wind turbines as shown in Table II.

The parameter settings of the seven compared algorithms are given in Table III. In TDA, the initial displacement distance standard deviation σ_{dis} and the initial direction standard deviation σ_{dir} were set to 500 and $\pi/6$, respectively. In CMA-ES, the step size σ is self-adaptively updated during the evolution. However, we found that if σ is fixed, better performance can be obtained for the wind farm layout. Therefore, in this paper σ was set to 15. For MSO, there were 10 sub-swarms and each sub-swarm had three particles. With respect to CLPSO, the population size NP was set to 60. The settings of the inertia weight and acceleration constants in MSO and CLPSO were consistent with their original papers. Regarding JADE and SHADE, NP was set to 100. Moreover, F and CR were adaptively tuned in JADE and SHADE as in their original papers. As mentioned previously, DEEM only contains two parameters, which were set as follows: $F = 0.9$ and $CR = 0.9$. For each algorithm, $MaxFEs$ was set to $150,000$.

B. Wind Scenario 1

The details of wind scenario 1 are summarized in Table IV, where i is the index of wind turbine, n is the index of the wind direction interval, and ξ_n is the frequency associated with the wind direction interval $[\theta_{n-1}, \theta_n)$. The wind direction from west to east is defined as $0°$ and the wind direction from south to north is defined as $90°$. It can be observed from Table IV that the wind directions are mainly distributed from $120°$ to $225°$. Therefore, in order to maximize the wind power output in this scenario, it is necessary to reduce the wake effect along a wide range of wind directions (i.e., from $120°$ to $225°$), which poses a great challenge for an algorithm to produce the optimal layout.

Firstly, Table V shows the maximal power output of the seven compared algorithms among 30 independent runs. As

TABLE V
THE MAXIMAL POWER OUTPUT (KW) OF THE SEVEN COMPARED ALGORITHMS IN WIND SCENARIO 1. THE HIGHEST POWER OUTPUT AMONG THE SEVEN COMPARED ALGORITHMS IS HIGHLIGHTED IN BOLDFACE FOR EACH CASE.

N	TDA	CMA-ES	MSO	CLPSO	JADE	SHADE	DEEM
15	6106.74	6023.09	6129.19	5961.60	5993.91	6082.06	**6275.03**
20	7585.92	7504.91	7303.67	7280.71	7328.88	7374.05	**7763.10**
25	8588.45	8859.97	8129.23	7991.80	8251.03	8253.73	**8991.92**
30	9719.08	10152.25	9233.02	9000.15	9102.20	9191.07	**10280.63**
35	11123.86	11051.74	10011.11	10104.07	10076.10	10300.94	**11631.64**
40	12160.36	12194.98	11022.51	11138.53	11028.41	11413.96	**12966.75**
60	15875.54	14862.37	13462.44	14233.88	13839.48	14356.68	**16975.80**
80	19485.85	17373.35	15597.53	17207.92	16424.15	17054.04	**20334.99**
100	22856.68	19163.00	16660.70	19438.94	18656.75	19566.80	**23415.03**

shown in Table V, DEEM consistently provides the best performance in terms of the maximal power output. Subsequently, Table VI summarizes the average and standard deviation of the power output derived from the seven compared algorithms over 30 independent runs. Next, we will discuss the experimental results from the following three aspects.

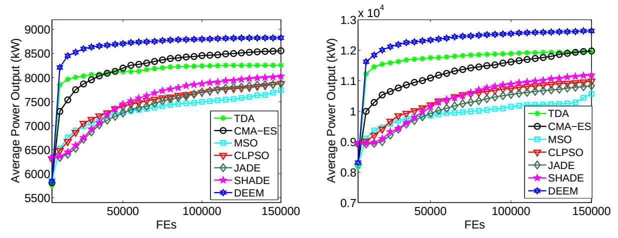
- As shown in Table VI, DEEM performs significantly better than the six competitors on all the cases, according to the Wilcoxon's rank sum test at a 0.05 significance level. One may be interested in why DEEM with simple DE operators even outperforms two well-established adaptive DE variants, i.e., JADE and SHADE. The reason is the following. In JADE and SHADE, the successful parameter settings, which can generate better offspring in previous generations, are used to create future parameter values. However, for wind farm layout, very often the offspring could not satisfy the constraints in (10). Under this condition, the offspring is worse than the parents. As a result, the amount of successful parameter settings is limited at the end of each generation, which results in insufficient information collected for updating the parameter settings.
- We also calculated the improvement rate of DEEM against the other six algorithms based on the average power output. It can be seen that, overall, DEEM has the increasing advantage over all the competitors except TDA as the number of wind turbines increases. For example, in the case of $N = 15$, the average power output of DEEM is 5.09%, 4.33%, 4.38%, 3.83%, and 2.33% higher than that of CMA-ES, MSO, CLPSO, JADE, and SHADE, respectively. When $N = 100$, DEEM can achieve 22.00%, 44.57%, 20.49%, 26.75%, and 20.79% performance improvement compared with CMA-ES, MSO, CLPSO, JADE, and SHADE, respectively. It is because when the number of wind turbines is small, the wind turbines can be placed sparsely in the wind farm easily. As a result, the wake effect can be reduced and the performance difference among the compared algorithms is not significant. However, when the number of wind turbines increases, the dimension of the search space increases drastically for CMA-ES, MSO, CLPSO, JADE, and SHADE. Since DEEM searches for the optimal layout in a two-dimensional search space, it has more potential to obtain better results. In addition, DEEM has good global search ability, hence it is also

TABLE VI
EXPERIMENTAL RESULTS OF THE SEVEN COMPARED ALGORITHMS IN WIND SCENARIO 1 (KW).

N	TDA Mean PO ± Std Dev	CMA-ES Mean PO ± Std Dev	MSO Mean PO ± Std Dev	CLPSO Mean PO ± Std Dev	JADE Mean PO ± Std Dev	SHADE Mean PO ± Std Dev	DEEM Mean PO ± Std Dev
15	5923.30 ± 105.26 + (4.38%)	5883.61 ± 111.15 + (5.09%)	5926.47 ± 136.49 + (4.33%)	5923.60 ± 35.84 + (4.38%)	5954.68 ± 24.70 + (3.83%)	6042.09 ± 30.65 + (2.33%)	6183.32 ± 49.90
20	7308.67 ± 151.64 + (5.00%)	7278.84 ± 143.16 + (5.43%)	7106.34 ± 163.91 + (7.99%)	7125.12 ± 75.66 + (7.71%)	7178.90 ± 103.86 + (6.90%)	7281.92 ± 58.42 + (5.39%)	7674.78 ± 62.76
25	8257.07 ± 178.85 + (6.91%)	8565.16 ± 175.58 + (3.07%)	7759.28 ± 212.48 + (13.77%)	7892.72 ± 71.44 + (11.85%)	7886.95 ± 155.52 + (11.93%)	8039.07 ± 85.18 + (9.81%)	8828.37 ± 156.98
30	9487.57 ± 129.37 + (6.29%)	9778.64 ± 160.79 + (3.13%)	8740.24 ± 270.88 + (15.38%)	8902.46 ± 73.32 + (13.28%)	8996.17 ± 56.71 + (12.10%)	9123.25 ± 60.71 + (10.54%)	10085.18 ± 112.36
35	10764.48 ± 220.19 + (6.02%)	10882.14 ± 127.82 + (4.88%)	9727.18 ± 222.82 + (17.33%)	9977.47 ± 67.45 + (14.39%)	9936.61 ± 85.80 + (14.86%)	10184.13 ± 68.88 + (12.07%)	11413.48 ± 151.55
40	11954.19 ± 133.80 + (5.73%)	11992.08 ± 141.93 + (5.40%)	10657.96 ± 242.13 + (18.59%)	11001.07 ± 77.25 + (14.89%)	10857.41 ± 81.75 + (16.41%)	11197.96 ± 106.42 + (12.87%)	12640.05 ± 225.18
60	15512.11 ± 235.16 + (6.61%)	14662.88 ± 121.89 + (12.79%)	12915.52 ± 532.44 + (28.05%)	14024.53 ± 145.60 + (17.92%)	13647.24 ± 117.31 + (21.18%)	14179.61 ± 134.04 + (16.63%)	16538.61 ± 209.79
80	19171.57 ± 169.58 + (4.35%)	17017.62 ± 267.48 + (17.56%)	14646.77 ± 539.44 + (36.59%)	16935.41 ± 138.92 + (18.13%)	16235.83 ± 177.18 + (23.22%)	16783.04 ± 283.50 + (19.20%)	20006.09 ± 150.02
100	22340.84 ± 309.75 + (3.58%)	18968.05 ± 168.45 + (22.00%)	16007.52 ± 300.91 + (44.57%)	19206.77 ± 126.08 + (20.49%)	18256.93 ± 246.90 + (26.75%)	19158.66 ± 413.39 + (20.79%)	23142.42 ± 204.74
+	9	9	9	9	9	9	/

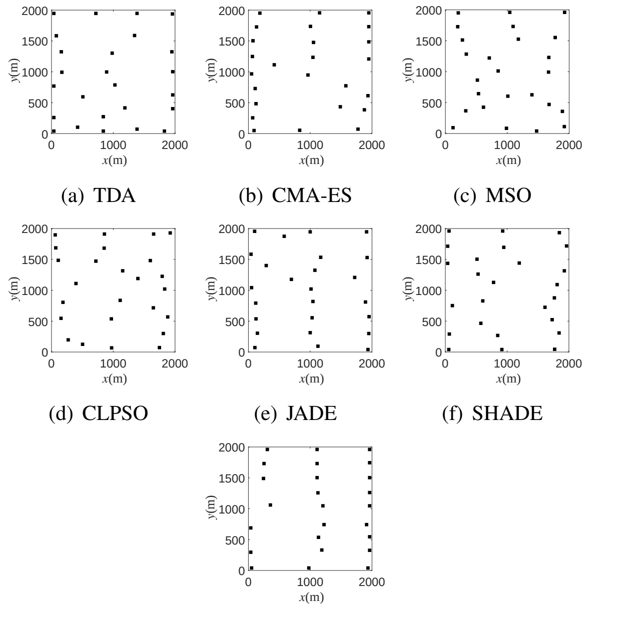
TABLE VII
RANKINGS OBTAINED BY THE FRIEDMAN'S TEST FOR THE SEVEN COMPARED ALGORITHMS IN WIND SCENARIO 1. THE BEST AND THE SECOND BEST RESULTS ARE HIGHLIGHTED IN BOLDFACE AND ITALIC, RESPECTIVELY.

Algorithm	Ranking
TDA	*2.8889*
CMA-ES	3.3333
MSO	6.6667
CLPSO	4.8889
JADE	5.4444
SHADE	3.7778
DEEM	**1**



(a) $N = 25$ (b) $N = 40$

Fig. 3. The evolution of the average power output provided by the seven compared algorithms for wind scenario 1.



(a) TDA (b) CMA-ES (c) MSO (d) CLPSO (e) JADE (f) SHADE (g) DEEM

Fig. 4. The best layouts of the seven compared algorithms with $N = 25$ in wind scenario 1.

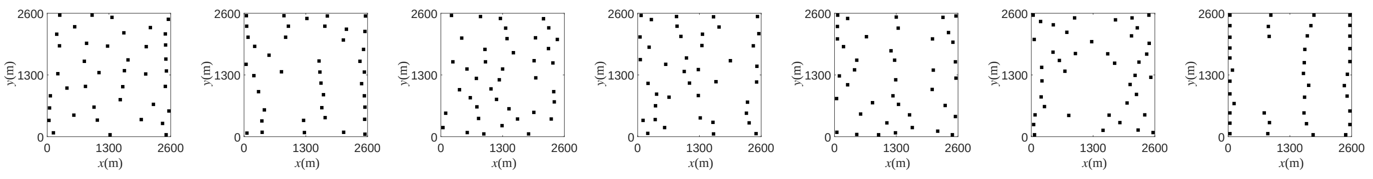
consistently better than TDA which adopts local search.

- Furthermore, based on the average power output, the Friedman's test was carried out by making use of KEEL software [31], in which the Bonferroni–Dunn method was chosen for the $post-hoc$ test. Table VII summarizes the statistical test results. It can be observed from Table VII that DEEM ranks the first, followed by TDA.

Fig. 3 presents the evolution of the average power output achieved by the seven compared algorithms on $N = 25$ and $N = 40$. From Fig. 3, at the initial stage (i.e., less than $10,000$ FEs), the average power output of DEEM reaches a large improvement rapidly. Moreover, DEEM maintains the highest average power output among the seven compared algorithms in the whole evolutionary process. The above phenomenon implies that DEEM converges faster than the six competitors.

The best layouts of the seven compared algorithms on $N = 25$ and $N = 40$ are shown in Fig. 4 and Fig. 5, respectively. When $N = 25$ in Fig. 4, CMA-ES, JADE, and DEEM enlarge the distances among the wind turbines along the predominant wind direction (i.e., from 120° to 225°) and prefer to place the wind turbines close to the left and right boundaries of the wind farm. With the increase of the number of wind turbines, such as $N = 40$ in Fig. 5, the layouts of all the algorithms except DEEM are relatively disordered. Overall, DEEM can generate more regular and symmetric layouts than other algorithms.

Fig. 6 summarizes the runtime of the seven compared algorithms versus the number of wind turbines. The first observation from Fig. 6 is that the seven compared algorithms can be divided into three groups: 1) DEEM and TDA, 2) MSO and CMA-ES, and 3) CLPSO, JADE, and SHADE. The algorithms in each group have the similar runtime. DEEM and TDA need the least runtime due to the usage of the



(a) TDA (b) CMA-ES (c) MSO (d) CLPSO (e) JADE (f) SHADE (g) DEEM

Fig. 5. The best layouts of the seven compared algorithms with $N = 40$ in wind scenario 1.

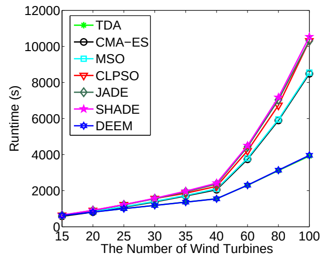


Fig. 6. Runtime of the seven compared algorithms for wind scenario 1.

TABLE IX
THE MAXIMAL POWER OUTPUT (KW) OF THE SEVEN COMPARED ALGORITHMS IN WIND SCENARIO 2. THE HIGHEST POWER OUTPUT AMONG THE SEVEN COMPARED ALGORITHMS IS HIGHLIGHTED IN BOLDFACE FOR EACH CASE.

N	TDA	CMA-ES	MSO	CLPSO	JADE	SHADE	DEEM
15	13007.32	12926.37	12922.7	12926.37	12925.13	12949.01	**13065.80**
20	16616.11	16499.97	16473.56	16499.97	16511.69	16614.43	**17068.28**
25	19896.88	18947.07	18938.67	18947.07	19062.41	19245.13	**20181.91**
30	23012.64	21965.25	21846.16	21965.25	21936.02	22351.84	**23894.34**
35	26541.68	24904.19	24569.87	24904.19	25088.55	25412.10	**27058.16**
40	29839.90	27931.65	27116.66	27931.65	27730.16	28398.42	**30791.14**
60	41489.54	37138.68	36156.21	37138.68	36656.46	37748.24	**42110.22**
80	51893.55	46319.76	43959.58	46319.76	44641.72	46884.31	**53413.98**
100	61332.58	53465.69	48959.17	53807.68	51559.90	53769.95	**62830.18**

TABLE VIII
WIND SCENARIO 2

n	θ_{n-1}	θ_n	$k_i(\theta)$	$c_i(\theta)$	ξ_n	n	θ_{n-1}	θ_n	$k_i(\theta)$	$c_i(\theta)$	ξ_n
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	0.2	18	255°	270°	2	13	0.01
7	90°	105°	2	13	0.6	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	300°	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

caching technique. Although all the algorithms in the second and third groups make use of the traditional encoding shown in Fig. 1(b), the runtime of the second group is less than that of the third group. It is probably because the implementation of CLPSO, JADE, and SHADE is more complicated than that of MSO and CMA-ES. Additionally, the runtime of the second and third groups is considerably higher than that of the first group with the increase of the number of wind turbines. Specifically, in the case of $N = 100$, CLPSO, JADE, and SHADE are nearly three times slower than DEEM and TDA.

C. *Wind Scenario* 2

The details of wind scenario 2 are presented in Table VIII, in which the prevailing wind directions are between 75° to 105°. The wind distribution is relatively simple and the wind speed is higher than wind scenario 1.

The maximal power output of the seven compared algorithms is provided in Table IX, which again indicates that DEEM shows the best performance. It seems that the maximal power output of each algorithm in wind scenario 2 is higher than wind scenario 1 on each case. This phenomenon can be explained as follows: the wind distribution focuses on a small scale such that it is easier to avoid the wake effect.

In addition, Table X recodes the average and standard deviation of the power output resulting from the seven compared algorithms over 30 runs. As depicted in Table X, DEEM performs significantly better than the six competitors on all the cases, according to the Wilcoxon's rank sum test at a 0.05 significance level. In terms of the improvement rate, the superiority of DEEM over all the competitors except TDA is more obvious with the increase of the number of wind turbines. For example, in the case of $N = 15$, the average power output of DEEM is 1.33%, 1.49%, 1.29%, 1.17%, and 0.90% higher than that of CMA-ES, MSO, CLPSO, JADE, and SHADE, respectively. In contrast, when $N = 100$, DEEM improves the performance by 18.93%, 38.60%, 16.65%, 23.62%, and 19.46% against CMA-ES, MSO, CLPSO, JADE, and SHADE, respectively. Fig. 7 exhibits the convergence graphs of the average power output of the seven compared algorithms on $N = 25$ and $N = 40$. Similar to wind scenario 1, DEEM has the capability to converge very fast. Table XI reports the statistical test results based on the Friedman's test. As shown in Table XI, DEEM has the best ranking, followed by TDA. Owing to the fact that an algorithm has the same computational time complexity in both wind scenario 1 and wind scenario 2, the runtime of each algorithm in wind scenario 2 is similar to that in wind scenario 1, and thus is omitted.

The best layouts of the seven compared algorithms on $N = 25$ and $N = 40$ are shown in Fig. 8 and Fig. 9, respectively. As shown in Figs. 8 and 9, many wind turbines in the layout of DEEM are located at the up and down boundaries of the wind farm. As a result, the distances among the wind turbines along the predominant wind directions (i.e., from 75° to 105°) are relatively larger, which suggests that the downstream wind turbines can reduce their wake effects caused by the upstream ones. Consequently, the layout of DEEM is expected to generate higher power output. On the contrary, the layouts provided by the six competitors do not show obvious pattern.

Remark 2: The experimental results in Section V-B and

TABLE X
EXPERIMENTAL RESULTS OF THE SEVEN COMPARED ALGORITHMS IN WIND SCENARIO 2 (KW).

N	TDA Mean PO ± Std Dev	CMA-ES Mean PO ± Std Dev	MSO Mean PO ± Std Dev	CLPSO Mean PO ± Std Dev	JADE Mean PO ± Std Dev	SHADE Mean PO ± Std Dev	DEEM Mean PO ± Std Dev
15	12860.78 ± 96.24 + (1.40%)	12869.62 ± 74.00 + (1.33%)	12849.88 ± 44.96 + (1.49%)	12874.64 ± 27.85 + (1.29%)	12890.42 ± 22.43 + (1.17%)	12924.43 ± 22.92 + (0.90%)	13041.60 ± 21.09
20	16413.36 ± 172.64 + (3.31%)	16586.29 ± 284.96 + (2.23%)	16251.58 ± 118.87 + (4.34%)	16304.34 ± 105.90 + (4.00%)	16357.57 ± 89.68 + (3.66%)	16466.05 ± 67.35 + (2.98%)	16957.70 ± 66.12
25	19639.62 ± 160.21 + (1.74%)	19637.62 ± 210.43 + (1.75%)	18681.85 ± 214.98 + (6.95%)	18824.44 ± 76.65 + (6.14%)	18916.37 ± 90.27 + (5.63%)	19135.22 ± 71.76 + (4.42%)	19981.99 ± 159.58
30	22830.94 ± 157.07 + (3.03%)	22788.90 ± 137.94 + (3.22%)	21366.47 ± 235.61 + (10.10%)	21747.00 ± 138.87 + (8.17%)	21778.58 ± 114.99 + (8.01%)	22115.38 ± 108.30 + (6.37%)	23524.68 ± 219.16
35	26088.91 ± 385.63 + (2.78%)	25929.40 ± 205.95 + (3.41%)	23928.27 ± 488.62 + (12.06%)	24737.83 ± 113.95 + (8.39%)	24610.20 ± 217.57 + (8.95%)	25133.57 ± 174.85 + (6.68%)	26814.60 ± 154.97
40	29432.24 ± 299.83 + (3.42%)	28899.48 ± 182.20 + (5.33%)	26600.74 ± 478.82 + (14.43%)	27731.31 ± 203.63 + (9.76%)	27449.53 ± 112.49 + (10.89%)	28116.44 ± 171.70 + (8.26%)	30440.38 ± 191.49
60	40618.21 ± 618.58 + (2.52%)	38036.21 ± 430.71 + (9.48%)	34205.70 ± 1572.50 + (21.74%)	36894.43 ± 171.90 + (12.87%)	36194.62 ± 281.48 + (15.05%)	37410.54 ± 194.25 + (11.31%)	41644.07 ± 425.66
80	51563.61 ± 247.88 + (2.02%)	46384.36 ± 442.76 + (13.41%)	40711.39 ± 2243.20 + (29.22%)	45930.86 ± 238.22 + (14.53%)	44113.56 ± 389.51 + (19.25%)	45923.53 ± 650.48 + (14.55%)	52608.78 ± 380.53
100	60755.42 ± 427.75 + (2.59%)	52407.19 ± 797.30 + (18.93%)	44972.44 ± 2522.27 + (38.60%)	53432.41 ± 310.76 + (16.65%)	50420.59 ± 1241.15 + (23.62%)	52174.49 ± 812.11 + (19.46%)	62332.23 ± 300.90
+	9	9	9	9	9	9	/

TABLE XI
RANKINGS OBTAINED BY THE FRIEDMAN'S TEST FOR THE SEVEN COMPARED ALGORITHMS IN WIND SCENARIO 2. THE BEST AND THE SECOND BEST RESULTS ARE HIGHLIGHTED IN BOLDFACE AND ITALIC, RESPECTIVELY.

Algorithm	Ranking
TDA	*2.6667*
CMA-ES	3.2222
MSO	7
CLPSO	4.8889
JADE	5.3333
SHADE	3.8889
DEEM	**1**

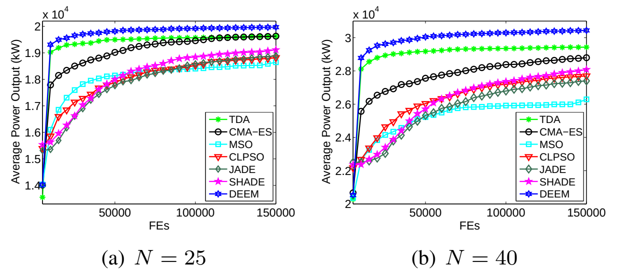


Fig. 7. The evolution of the average power output provided by the seven compared algorithms for wind scenario 2.

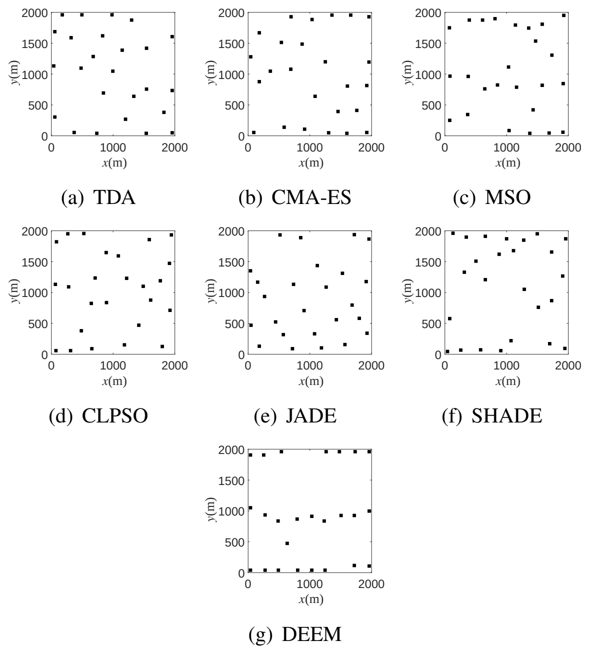


Fig. 8. The best layouts of the seven compared algorithms with $N = 25$ in wind scenario 2.

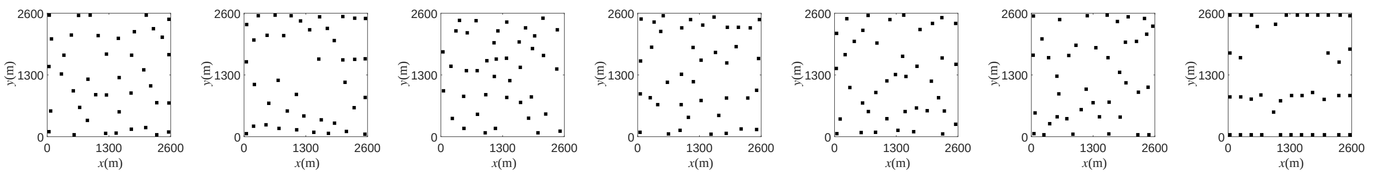
Section V-C reveal that DEEM succeeds in achieving higher power output as well as faster convergence speed than the six competitors, i.e., TDA, CMA-ES, MSO, CLPSO, JADE, and SHADE. The superior performance of DEEM could be due to two facts: 1) with the proposed encoding mechanism, DEEM consistently searches for the optimal layout in a two-dimensional search space, remarkably enhancing the search efficiency; and 2) By utilizing DE as the search engine, DEEM shows good global search ability.

D. Comparison with A Latest Greedy Algorithm

This subsection aims at comparing DEEM with a latest greedy algorithm proposed by Chen *et al.* [32] in 2016. This greedy algorithm firstly divides the wind farm into grids and then all grid cells are numbered. Afterward, each wind turbine is in turn located into an empty grid cell to achieve the minimum evaluation value designed in [32]. If all the wind turbines have been placed in the grid cells, the process of this greedy algorithm is completed.

Due to the space limitation, wind scenario 1 was used to produce the experimental results. For this greedy algorithm, all the parameter settings were kept the same with [32]. Note that this greedy algorithm is a deterministic algorithm; thus the experimental result for each case is unchanged in different independent runs. The experimental result of this greedy algorithm and the maximal power output of DEEM are given in Table XII. As shown in Table XII, DEEM provides higher power output on all the cases with the exception of $N = 60$. In the case of $N = 60$, the greedy algorithm performs slightly better than DEEM.

In addition, Fig. 10 plots the runtime of the greedy algorithm and DEEM versus the number of wind turbines. From Fig. 10,



(a) TDA (b) CMA-ES (c) MSO (d) CLPSO (e) JADE (f) SHADE (g) DEEM

Fig. 9. The best layouts of the seven compared algorithms with $N = 40$ in wind scenario 2.

TABLE XII
EXPERIMENTAL RESULTS OF THE GREEDY ALGORITHM IN [32] AND DEEM IN WIND SCENARIO 1. THE HIGHER POWER OUTPUT (KW) BETWEEN THE TWO COMPARED ALGORITHMS IS HIGHLIGHTED IN BOLDFACE FOR EACH CASE.

N	The Greedy Algorithm in [32]	DEEM
15	6080.83	**6275.03**
20	7511.22	**7763.10**
25	8588.47	**8991.92**
30	10190.07	**10280.63**
35	11394.42	**11631.64**
40	12717.68	**12966.75**
60	**17025.11**	16975.80
80	19757.00	**20334.99**
100	23173.74	**23415.03**

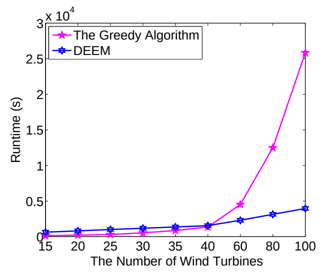


Fig. 10. Runtime of the greedy algorithm in [32] and DEEM for wind scenario 1.

it is easy to see that when N is between 15 and 40, the two compared algorithms show similar computational time. However, the runtime of the greedy algorithm drastically increases from $N = 40$. For instance, in the case of $N = 100$, DEEM is six times faster than the greedy algorithm. This can be attributed to the fact that the number of FEs consumed by the greedy algorithm exponentially increases from $N = 40$. According to our observation, the number of FEs in the greedy algorithm is $24,000$, $32,000$, $40,000$, $58,080$, $80,640$, $108,160$, $230,640$, $414,720$, and $640,000$ for $N = 15$, 20, 25, 30, 35, 40, 60, 80, and 100, respectively.

VI. DISCUSSIONS

Additional experiments were conducted in this section to study the following five issues:

- Can the performance of DEEM be improved via adaptive parameter settings?
- What is the effect of the mutation operators on the performance of DEEM?
- Is DEEM sensitive to its two control parameters F and CR?

TABLE XIII
EXPERIMENTAL RESULTS OF THE ADAPTIVE DEEM AND DEEM IN WIND SCENARIO 1 (KW).

N	Adaptive DEEM Mean PO ± Std Dev	DEEM Mean PO ± Std Dev
15	6182.08 ± 33.60 ≈	6183.33 ± 49.90
20	7718.82 ± 86.51 ≈	7674.79 ± 62.76
25	8801.48 ± 148.27 ≈	8828.38 ± 156.98
30	10085.19 ± 104.53 ≈	10085.18 ± 112.36
35	11291.30 ± 178.15 ≈	11413.48 ± 151.55
40	12803.96 ± 207.90 ≈	12640.05 ± 225.18
60	16465.62 ± 105.60 ≈	16538.61 ± 209.79
80	20191.99 ± 417.33 ≈	20006.09 ± 150.02
100	23258.62 ± 263.68 ≈	23142.43 ± 204.74
≈	9	/

- Is the performance of DEEM better than that of DE with the traditional encoding?
- Can DEEM be used for wind turbine layout optimization with multiple hub height wind turbines?

Next, we will address these five issues one by one. Our experiments focused on wind scenario 1. In all the experiments, 30 independent runs were implemented for each algorithm, and the parameter settings were the same as those introduced in Section V-A, unless we mentioned new settings. Wilcoxon's rank sum test at a 0.05 significance level was performed to test the statistical significance between two algorithms. In all the tables of this section, "+", "−", and "≈" denotes the performance of DEEM is better than, worse than, and similar to that of another algorithm, respectively. In addition, "Mean PO" and "Std Dev" indicate the average and standard deviation of the power output (kW) in 30 runs, respectively, "Maximal PO" denotes the maximal power output, and percentages in parentheses denote the improvement rates of DEEM against other algorithms.

1) *Adaptive DEEM Versus DEEM*: By incorporating the adaptive parameter settings of SHADE [29] into DEEM, we obtained a variant of DEEM, called adaptive DEEM. It can be seen from Table XIII that the adaptive DEEM and DEEM show similar overall performance, which implies that the direct use of the adaptive mechanism from SHADE cannot significantly improve the performance of DEEM. It is perhaps because the dimension of the search space is quite low (i.e., 2), and under this condition DEEM can already achieve competitive performance without any further improvements.
2) *Effect of the Mutation Operators*: In order to study the effect of the mutation operators on the performance of DEEM, we replaced DE/rand/1 in the orig-

TABLE XIV
EXPERIMENTAL RESULTS OF DEEM WITH DIFFEREN MUTATION OPERATORS IN WIND SCENARIO 1 (KW).

N	DEEM/rand/2 Mean PO $\pm$ Std Dev	DEEM/current-to-rand/1 Mean PO $\pm$ Std Dev	DEEM Mean PO $\pm$ Std Dev
15	6185.68$\pm$40.60 $\approx$	6193.78$\pm$28.32 $\approx$	6183.33$\pm$49.90
20	7762.31$\pm$71.58 $-$	7673.73$\pm$109.08 $\approx$	7674.79$\pm$62.76
25	8801.91$\pm$139.93 $\approx$	8829.04$\pm$119.60 $\approx$	8828.38$\pm$156.98
30	10020.82$\pm$89.27 $\approx$	10094.05$\pm$110.55 $\approx$	10085.18$\pm$112.36
35	11316.04$\pm$184.31 $\approx$	11361.88$\pm$195.31 $\approx$	11413.48$\pm$151.55
40	12568.67$\pm$233.63 $\approx$	12703.84$\pm$229.07 $\approx$	12640.05$\pm$225.18
60	16260.32$\pm$121.13 $+$	16378.01$\pm$103.84 $+$	16538.61$\pm$209.79
80	19786.87$\pm$202.87 $\approx$	19975.75$\pm$202.13 $\approx$	20006.09$\pm$150.02
100	23002.12$\pm$212.61 $\approx$	23087.34$\pm$126.71 $\approx$	23142.43$\pm$204.74
$+$	1	1	/
$-$	1	0	/
$\approx$	7	8	/

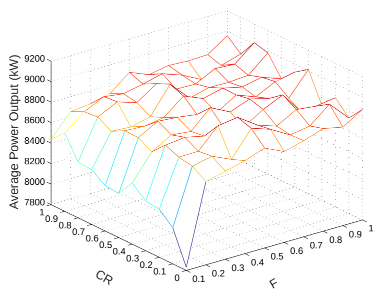


Fig. 11. The average power output of DEEM with different combinations of F and CR on wind scenario 1 with $N = 25$.

inal DEEM with two other commonly used mutation operators (DE/rand/2 and DE/current-to-rand/1). The resultant variants of DEEM are called DEEM/rand/2 and DEEM/current-to-rand/1, respectively. Note that in the DE community, there are several mutation operators which utilize the information of the best individual, such as DE/best/1, DE/best/2, and DE/current-to-best/1. Since in DEEM the population represents an entire layout, it cannot define the best individual in the population. Thus, such mutation operators were not applied to DEEM in this paper. As shown in Table XIV, DEEM performs similarly to DEEM/rand/2 and DEEM/current-to-rand/1 on seven and eight out of nine cases, respectively. Therefore, DEEM can still maintain its performance after combining with DE/rand/2 or DE/current-to-rand/1.

3) *Sensitivity in Relation to F and CR*: In order to investigate the sensitivity of F and CR, we tested DEEM with different values of F and CR on wind scenario 1 with $N = 25$, shown in Fig. 11. From Fig. 11, DEEM is not sensitive to F and CR, and they can be set into values in a large range (for instance, $F \in [0.2, 1.0]$ and $CR \in [0.0, 1.0]$). Obviously, $F = 0.1$ causes clear performance degradation. It is because $F = 0.1$ has a side effect on the exploration ability of DEEM due to the small perturbation. It is also interesting to note that DEEM with $CR = 0$ performs well in the case of $F > 0.1$. The reason is the following: even though $CR = 0$, the trial vector can still inherit some information from the mutant vector if the condition "$j = j_{rand}$" is satisfied as shown in (13).

TABLE XV
EXPERIMENTAL RESULTS OF DE WITH THE TRADITIONAL ENCODING AND DEEM IN WIND SCENARIO 1 (KW).

N	DE with the Traditional Encoding Mean PO $\pm$ Std Dev	DEEM Mean PO $\pm$ Std Dev
15	5448.62 $\pm$ 77.04 $+$ (13.48%)	6183.33 $\pm$ 21.09
20	6263.80 $\pm$ 116.14 $+$ (22.53%)	7674.79 $\pm$ 66.12
25	6543.73 $\pm$ 145.24 $+$ (34.91%)	8828.38 $\pm$ 159.58
30	7283.25 $\pm$ 153.29 $+$ (38.47%)	10085.18 $\pm$ 219.16
35	7990.41 $\pm$ 96.65 $+$ (42.84%)	11413.48 $\pm$ 154.97
40	8881.62 $\pm$ 101.87 $+$ (42.32%)	12640.05 $\pm$ 191.49
60	11370.47 $\pm$ 136.13 $+$ (45.45%)	16538.61 $\pm$ 425.66
80	13743.65 $\pm$ 122.91 $+$ (45.57%)	20006.09 $\pm$ 380.53
100	15775.49 $\pm$ 156.33 $+$ (46.70%)	23142.43 $\pm$ 300.90
$+$	9	/

TABLE XVI
EXPERIMENTAL RESULTS OF THE EIGHT COMPARED ALGORITHMS FOR MULTIPLE HUB HEIGHT WIND TURBINES IN WIND SCENARIO 1 WITH $N = 20$ (KW). THE BEST MAXIMAL POWER OUTPUT AMONG THE EIGHT COMPARED ALGORITHMS IS HIGHLIGHTED IN BOLDFACE.

Algorithm	Mean PO $\pm$ Std DeV	Maximal PO
TDA	7343.74 $\pm$ 135.23 $+$ (6.19%)	7579.10
CMA-ES	7429.57 $\pm$ 133.49 $+$ (4.96%)	7593.48
MSO	7019.67 $\pm$ 191.14 $+$ (11.09%)	7275.37
CLPSO	7189.88 $\pm$ 46.21 $+$ (8.46%)	7256.16
JADE	7201.42 $\pm$ 68.04 $+$ (8.29%)	7310.46
SHADE	7341.60 $\pm$ 47.30 $+$ (6.22%)	7419.87
The Greedy Algorithm in [32]	/	7572.77
DEEM	7798.11 $\pm$ 64.55	**7931.71**

4) *DE with the Traditional Encoding Versus DEEM*: The experimental results of DE with the traditional encoding in Fig. 1(b) and DEEM are presented in Table XV. For DE with the traditional encoding, the population size NP was set to 100, and the settings of F and CR were the same with DEEM. From Table XV, DEEM is significantly superior to DE with the traditional encoding on all the cases, which verifies the rationality of our main motivation – the new encoding mechanism.

5) *DEEM for Wind Farm Layout Design with Multiple Hub Height Wind Turbines*: In the above experiments, all wind turbines have the identical height. However, wind turbines may have different heights in the real-world wind farm layout design. As pointed out in [32], wind turbines with multiple hub heights can reduce the wake effect and extract more wind power. To this end, we considered wind turbines with two optional hub heights, i.e., 50 m and 78 m. The experimental results of TDA, CMA-ES, MSO, CLPSO, JADE, SHADE, the greedy algorithm in [32], and DEEM are given in Table XVI for wind scenario 1 with $N = 20$. As shown in Table XVI, DEEM provides the best maximal power output among the eight compared algorithms. Moreover, DEEM is statistically better than TDA, CMA-ES, MSO, CLPSO, JADE, and SHADE. The

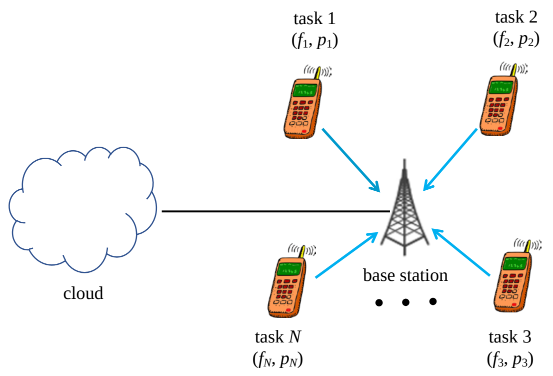


Fig. 12. Resource optimization in mobile cloud computing.

above comparison reveals the potential of DEEM for wind farm layout design with multiple hub height wind turbines.

VII. Conclusions

In this paper, a differential evolution (DE) algorithm with a new encoding mechanism (called DEEM) was proposed for the layout optimization of a wind farm. The coordinate-based model was employed and maximizing the power output of the wind farm was regarded as the optimization objective. The new encoding mechanism views each wind turbine as an individual. Thus, the whole population represents an entire layout and the search space only contains two dimensions irrespective of the number of wind turbines. DEEM also benefits from DE for the global search. Moreover, by only updating one wind turbine in one iteration, the caching technique can be used to accelerate the evaluation. The implementation of DEEM is simple and it includes few control parameters.

Systematic experiments were conducted on DEEM and seven other state-of-the-art algorithms. The experimental results confirmed that, overall, DEEM achieves the highest power output with the fastest convergence speed. The robustness of DEEM was also demonstrated by investigating two wind scenarios with various number of wind turbines. Besides, a comprehensive set of experiments were carried out to study the effect of the mutation operators and the control parameters on the performance of DEEM, the performance of adaptive DEEM, the performance difference between DEEM and DE with the traditional encoding, and the applicability of DEEM to wind turbines with multiple hub heights.

In the future, we will built wind farm layout models with more complicated properties and deal with them via DEEM. Moreover, we are considering the possibility of applying the proposed encoding mechanism to optimization problems in other fields, such as resource optimization in mobile cloud computing. As an emerging technology, mobile cloud computing can bridge the gap between limited capabilities of mobile devices and increasing demand of resource-intensive applications, by offloading the tasks to cloud infrastructures [33]. However, offloading will incur extra overhead of energy and latency, and the amount of extra overhead is determined by the resources allocated to each task, such as computation and communication resources. Therefore, in order to improve the offloading performance, it is necessary to optimize the resource allocation in mobile cloud computing, with the aim of reducing the energy and latency. Assuming that the scenario contains N mobile devices as shown in Fig. 12. Each mobile device has a task to be completed, and the computation resource and communication resource allocated to each task are denoted as f_i and p_i $(i = 1, 2, ..., N)$, respectively. Subsequently, through the base station these tasks can be offloaded to the cloud to be executed. During the resource optimization, by making use of the proposed encoding mechanism, each task can be considered as an individual containing two dimensions (i.e., f_i and p_i), and all the tasks thus form a population.

References

[1] M. B. Ozkan and P. Karagoz, “A novel wind power forecast model: Statistical hybrid wind power forecast technique (shwip),” *IEEE Transactions on Industrial Informatics*, vol. 11, no. 2, pp. 375–387, 2015.

[2] M. Tan and Z. Zhang, “Wind turbine modeling with data-driven methods and radially uniform designs,” *IEEE Transactions on Industrial Informatics*, vol. 12, no. 3, pp. 1261–1269, 2016.

[3] G. Mosetti, C. Poloni, and B. Diviacco, “Optimization of wind turbine positioning in large windfarms by means of a genetic algorithm,” *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 51, no. 1, pp. 105–116, 1994.

[4] S. Grady, M. Hussaini, and M. M. Abdullah, “Placement of wind turbines using genetic algorithms,” *Renewable Energy*, vol. 30, no. 2, pp. 259–270, 2005.

[5] J. C. Mora, J. M. C. Barón, J. M. R. Santos, and M. B. Payán, “An evolutive algorithm for wind farm optimal design,” *Neurocomputing*, vol. 70, no. 16, pp. 2651–2658, 2007.

[6] S. Pookpunt and W. Ongsakul, “Optimal placement of wind turbines within wind farm using binary particle swarm optimization with time-varying acceleration coefficients,” *Renewable Energy*, vol. 55, pp. 266–276, 2013.

[7] D. Jiang, C. Peng, Z. Fan, Y. Chen, and X. Cai, “Modified binary differential evolution for solving wind farm layout optimization problems,” in *2013 IEEE Symposium on Computational Intelligence for Engineering Solutions (CIES)*, April 2013, pp. 23–28.

[8] A. Kusiak and Z. Song, “Design of wind farm layout for maximum wind energy capture,” *Renewable Energy*, vol. 35, no. 3, pp. 685–694, 2010.

[9] C. Wan, J. Wang, G. Yang, and X. Zhang, “Optimal micro-siting of wind farms by particle swarm optimization,” in *International Conference in Swarm Intelligence*, 2010, pp. 198–205.

[10] B. Saavedra-Moreno, S. Salcedo-Sanz, A. Paniagua-Tineo, L. Prieto, and A. Portilla-Figueras, “Seeding evolutionary algorithms with heuristics for optimal wind turbines positioning in wind farms,” *Renewable Energy*, vol. 36, no. 11, pp. 2838–2844, 2011.

[11] Y. Eroğlu and S. U. Seçkiner, “Design of wind farm layout using ant colony algorithm,” *Renewable Energy*, vol. 44, pp. 53–62, 2012.

[12] M. Wagner, K. Veeramachaneni, F. Neumann, and U.-M. OReilly, “Optimizing the layout of 1000 wind turbines,” *European Wind Energy Association Annual Event*, pp. 205–209, 2011.

[13] D. Jiang, C. Peng, Z. Fan, Y. Chen, and X. Cai, “Modified binary differential evolution for solving wind farm layout optimization problems,” in *2013 IEEE Symposium on Computational Intelligence for Engineering Solutions (CIES)*, 2013, pp. 23–28.

[14] M. Wagner, J. Day, and F. Neumann, “A fast and effective local search algorithm for optimizing the placement of wind turbines,” *Renewable Energy*, vol. 51, pp. 64–70, 2013.

[15] J. Feng and W. Z. Shen, “Solving the wind farm layout optimization problem using random search algorithm,” *Renewable Energy*, vol. 78, pp. 182–192, 2015.

[16] K. Chen, M. Song, Z. He, and X. Zhang, “Wind turbine positioning optimization of wind farm using greedy algorithm,” *Journal of Renewable and Sustainable Energy*, vol. 5, no. 2, p. 023128, 2013.

[17] U. A. Ozturk and B. A. Norman, “Heuristic methods for wind energy conversion system positioning,” *Electric Power Systems Research*, vol. 70, no. 3, pp. 179–185, 2004.

[18] C. Zhang, G. Hou, and J. Wang, “A fast algorithm based on the submodular property for optimization of wind turbine positioning,” *Renewable Energy*, vol. 36, no. 11, pp. 2951–2958, 2011.

[19] M. Song, K. Chen, Z. He, and X. Zhang, "Bionic optimization for micro-siting of wind farm on complex terrain," *Renewable Energy*, vol. 50, pp. 551–557, 2013.

[20] J. Yang, J. Feng, and W. Z. Shen, "Optimization of wind farm layout: a refinement method by random search," in *International Conference on aerodynamics of Offshore Wind Energy Systems and wakes (ICOWES 2013)*, 2013, pp. 624–633.

[21] H. Long and Z. Zhang, "A two-echelon wind farm layout planning model," *IEEE Transactions on Sustainable Energy*, vol. 6, no. 3, pp. 863–871, 2015.

[22] I. Katic, J. Højstrup, and N. O. Jensen, "A simple model for cluster efficiency," in *European Wind Energy Association Conference and Exhibition*, 1986, pp. 407–410.

[23] M. A. Lackner and C. N. Elkinton, "An analytical framework for offshore wind farm layout optimization," *Wind Engineering*, vol. 31, no. 1, pp. 17–31, 2007.

[24] D. W. Stroock, *A Concise Introduction to the Theory of Integration*. Springer Science & Business Media, 1999.

[25] S. Das, S. S. Mullick, and P. Suganthan, "Recent advances in differential evolution – an updated survey," *Swarm and Evolutionary Computation*, vol. 27, pp. 1 – 30, 2016.

[26] H. Long, Z. Zhang, Z. Song, and A. Kusiak, "Formulation and analysis of grid and coordinate models for planning wind farm layouts," *IEEE Access*, vol. 5, pp. 1810–1819, 2017.

[27] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, June 2006.

[28] J. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, Oct 2009.

[29] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *2013 IEEE Congress on Evolutionary Computation*, June 2013, pp. 71–78.

[30] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[31] J. Alcalá-Fdez, L. Sánchez, S. García, M. J. del Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, J. C. Fernández, and F. Herrera, "KEEL: a software tool to assess evolutionary algorithms for data mining problems," *Soft Computing*, vol. 13, no. 3, pp. 307–318, 2009.

[32] K. Chen, M. Song, X. Zhang, and S. Wang, "Wind turbine layout optimization with multiple hub height wind turbines using greedy algorithm," *Renewable Energy*, vol. 96, Part A, pp. 676 – 686, 2016.

[33] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 4, pp. 3435–3447, April 2017.

Yong Wang (M'08) received the B.S. degree in automation from the Wuhan Institute of Technology, Wuhan, China, in 2003, and the M.S. degree in pattern recognition and intelligent systems and the Ph.D. degree in control science and engineering both from the Central South University (CSU), Changsha, China, in 2006 and 2011, respectively.

He is currently an Associate Professor with the School of Information Science and Engineering, C-SU. His current research interests include the theory, algorithm design, and applications of computational intelligence.

Dr. Wang was awarded the Hong Kong Scholar by the Mainland - Hong Kong Joint Postdoctoral Fellows Program, China, in 2013, the Excellent Doctoral Dissertation by Hunan Province, China, in 2013, the New Century Excellent Talents in University by the Ministry of Education, China, in 2013, the 2015 IEEE Computational Intelligence Society Outstanding PhD Dissertation Award, the Hunan Provincial Natural Science Fund for Distinguished Young Scholars, in 2016, the EU Horizon 2020 Marie Sklodowska-Curie Fellowship, in 2016, and a Highly Cited Researcher in computer science by Clarivate Analytics, in 2017. He is currently serving as an associate editor for the *Swarm and Evolutionary Computation*.

Hao Liu received the B.S. degree in automation from Central South University, Changsha, China, in 2015, where he is currently pursuing the M.S. degree in control science and engineering.

His research interests include real-world applications of computational intelligence and machine learning.

Huan Long (S'15) received her Ph. D. degree in Department of Systems Engineering and Engineering Management, City University of Hong Kong, Hong Kong, China, in 2017 and B.S. degree in Department of Automation from Huazhong University of Science and Technology, China, in 2013.

Her research includes data mining and computational intelligence applied in the renewable energy optimization, such as wind farm layout, hybrid renewable system configuration, renewable energy prediction, and wind turbine monitoring.

Zijun Zhang (M'12) received his Ph.D. and M.S. degrees in Industrial Engineering from the University of Iowa, Iowa City, IA, USA, in 2012 and 2009, respectively, and B.Eng. degree in Systems Engineering and Engineering Management from the Chinese University of Hong Kong, Hong Kong, China, in 2008.

Currently, he is an Assistant Professor in the Department of Systems Engineering and Engineering Management at the City University of Hong Kong, Hong Kong, China. His research focuses on data mining and computational intelligence with applications in wind energy, HVAC and wastewater processing domains.

Shengxiang Yang (M'00–SM'14) received the B.Sc. and M.Sc. degrees in automatic control and the Ph.D. degree in systems engineering from Northeastern University, Shenyang, China in 1993, 1996, and 1999, respectively.

He is currently a Professor in Computational Intelligence and Director of the Centre for Computational Intelligence, School of Computer Science and Informatics, De Montfort University, Leicester, U.K. He has over 230 publications. His current research interests include evolutionary computation, swarm intelligence, computational intelligence in dynamic and uncertain environments, artificial neural networks for scheduling, and relevant real-world applications. He serves as an Associate Editor or Editorial Board Member of 8 international journals, such as the *IEEE Transactions on Cybernetics*, *Information Sciences*, *Evolutionary Computation*, *Neurocomputing*, and *Soft Computing*.