Archives of Computational Methods in Engineering
https://doi.org/10.1007/s11831-021-09586-7

ORIGINAL PAPER

# Comparative Performance of Twelve Metaheuristics for Wind Farm Layout Optimisation

Tawatchai Kunakote[1] · Numchoak Sabangban[2] · Sumit Kumar[3] · Ghanshyam G. Tejani[4] · Natee Panagant[2] · Nantiwat Pholdee[2] · Sujin Bureerat[2] · Ali R. Yildiz[5]

Received: 29 September 2020 / Accepted: 7 April 2021


**Abstract**
This work bridges two research fields i.e. metaheuristics and wind farm layout design. Comparative performance of twelve metaheuristics (MHs) on wind farm layout optimisation (WFLO) was conducted. Four WFLO problems are proposed for benchmarking the various metaheuristics while the design problem is an attempt to simultaneously minimise wind farm cost and maximise wind farm totally produced power. Design variables are wind turbine placement with fixed and varied number of wind turbines. The Jansen's wake model is used while two types of energy estimation with and without considering partially overshadowed wake areas are studied. The results obtained from using various MHs are statistically compared in terms of convergence and consistency while the best performer is obtained. Comparison results indicated that moth-flame optimisation (MFO) algorithm is the most efficient algorithms. The results obtained in this work are said to be the baseline for future study on WFLO using metahueristics.

## 1 Introduction

Due to the climate change issue, green renewable energy has become a trend in modern energy production where wind energy is one of the most popular resources [1, 2]. This is due to wind energy harvesters which are wind turbines having established technology. To create a wind farm for installing a set of wind turbines, wind data at a particular area need to be acquired including wind velocities and directions over a year. Often, this is illustrated as a wind rose diagram and a Weibull distribution curve [3]. With such a diagram and the available area to install the wind farm, the next task is to define the layout or where to place those wind turbines in the farm. The concept is simple as, for two overlapping turbines, the front turbine will fully harvest the energy while the rear one will have much less energy left for harvesting. As a result, the wind farm layout design is to find the optimum placement of all turbines in order to harvest the highest possible wind energy while some constraints may be imposed. This leads to an optimization problem posed to find the optimum wind farm layout.

Generally, the wind turbines are installed on the wind farm in order to use the available land efficiently and producing more power. Installation of a wind turbine into the farm needs to account for several parameters such as the number of the wind turbines, distancing, directions, and positions.

✉ Nantiwat Pholdee
nantiwat@kku.ac.th

Sumit Kumar
sumit21sep1990@gmail.com

Ghanshyam G. Tejani
p.shyam23@gmail.com

Natee Panagant
natepa@kku.ac.th

Sujin Bureerat
sujbur@kku.ac.th

Ali R. Yildiz
aliriza@uludag.edu.tr

[1] Department of Mechanical Engineering, Faculty of Engineering, Mahasarakham University, Maha Sarakham, Thailand

[2] Sustainable and Infrastructure Research and Development Center, Department of Mechanical Engineering, Faculty of Engineering, Khon Kaen University, Khon Kaen 40002, Thailand

[3] Department of Mechanical Engineering, GPERI, Gujarat Technological University, Ahmedabad, Gujarat, India

[4] Department of Mechanical Engineering, School of Technology, GSFC University, Vadodara, Gujarat, India

[5] Department of Automotive Engineering, Bursa Uludag University, Bursa, Turkey

Published online: 18 April 2021

This is because when the wind comes into contact with the turbine, kinetic energy is absorbed and converted to be a rotational motion of the turbine blade to produce electric energy. With this phenomenon, wake zone is generated and the velocity of the wind at the wake is reduced leading to the reduced efficiency of the downstream turbine [4–6].

A wind turbine placement problem also known as wind farm layout optimisation (WFLO) problem is said to be an optimisation problem due to the requirement of minimising installation cost and maximising energy production. Such a design problem has been widely investigated by numerous researchers. The WFLO problem is a combinatorial problem in which the classical gradient-based optimisation technique is difficult to apply due to use of discrete design variables [7, 8]. Therefore, metaheuristics (MHs) are an alternative choice, which has been proven to be a powerful tool for this design task. Some MHs applied to WFLO include GA [9], DE [10], ACO [11], RS [12], SA [13], PSO [14], TLBO [15] etc. Using MHs is advantageous since they are derivative-free. The method works by exploiting a set of design solutions usually termed a population. The population is then evolved by means of reproduction and selection. This makes MHs be able to tackle global optimisation. They can deal with almost any kind of design variables. This makes the methods more popular in some engineering fields. The WFLO problem is said to be complicated and difficult to solve as the layout leads to discrete design variables which is difficult to use classical mathematical programming. With a given layout, the wake of all the wind turbines are found while any turbine set up in the wake areas can produced less energy. The Jensen's wake decay model [16] is arguably the most popular wake model while there are other wake models in the literature e.g. the Frandsen's wake model [17], the Larsen's wake model [18], the Ishihara's wake model [19], the Bastankhah and Porte–Agel's (BP) wake model [20], and Xie and Archer's (XA) wake model [21]. The most popular approach WFLO is a grid-based approach which is carried out by predefining a grid covering the wind farm area while only one turbine is allowed to be placed on the centre of a particular grid. Meanwhile, the metaheuristics, although being popular with their ease of use and derivative free nature, usually have slow convergence rate and lag search consistency meaning it is unlikely to obtain the same optimum solution if they are run several times. Also, the so-called no free lunch theory stating that there are no MHs that are powerful for all types of optimisation problems, is always applied. Moreover, as an MH procedure has no serious restrictions and uses randomisation, there have been numerous MHs developed recently. It is limited to see the comparative performance of a number of established and newly invented metaheuristics for solving WFLO [15].

Therefore, this work is set to bridge the gap between the two research fields, WFLO and MHs. The comparative performance of many MHs is conducted for WFLO optimisation problems. Those include artificial bee colony method (ABC) [22], real-code ant colony optimization (ACOR) [23], differential evolution (DE) [24], particle swarm optimization (PSO) [25], teaching–learning based optimisation (TLBO) [26], evolution strategy with covariance matrix adaptation (CMA-ES) [27], moth-flame optimization algorithm (MFO) [28], sine cosine algorithm (SCA) [29], whale optimization algorithm (WOA) [30], crow search optimization algorithm (CSOA) [31], salp swarm optimizer (SSO) [32] and grasshopper optimization algorithm (GOA) [33]. The Jansen' wake model is used while two types of energy estimation with and without considering a partially overshadowed area are studied. Four WFLO problems are assigned for benchmarking. The results are compared in terms of both convergence and consistency while the best performer is obtained. The remainder of the paper are: formulation of the wind farm layout optimisation problem, numerical experiment, results and discussion, and conclusions.

## 2 Formulation of a Wind Farm Layout Optimisation Problem

### 2.1 Wind Turbine Wake Models

Wind farm layout design is carried out to find the best placement of a number of wind turbine in an available area so as to obtain the maximum power produced by the wind turbines. In this work, only a horizontal-axis wind turbine (HAWT) is considered. It is also possible to add other design criteria such as noise [34] and cost [15] to the design problem. In order to estimate the total power produced by a given wind layout, some parameters of the wind farm and wind turbines have to be predefined including wind speed and direction, wind turbine efficiency, turbine thrust coefficient, hub height, turbine diameter, terrain conditions, etc. A turbine wake is one of the most vital elements for wind farm design. In fact, it is simply the conservation of energy as once the turbine harvests the energy from full wind speed, the energy left after turbine must be decreased. However, as induced by full wind speed outside the wake, the velocity in the turbine wake increase as it moves away from the turbine. Over several decades, the issue of modelling a turbine wake has been focused by researchers around the globe. The more accurate turbine wake the more accurate power prediction. One of the most popularly used models, however, is the Jensen's wake decay model [35] although there have been some improved models in later years [17–21]The Jensen wake model assumes the initial wake to have the same diameter to that of the turbine. Then, it expends linearly behind the turbine. Figure 1 shows the Jensen wake model given that the wind velocity is uniform. Once the wind passes through

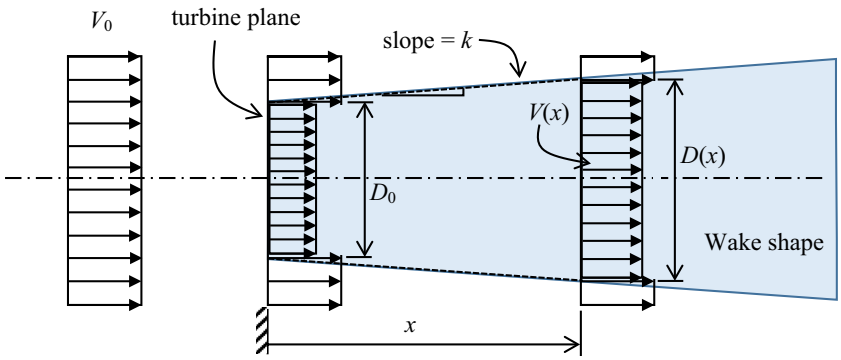


**Fig. 1** Wind turbine wake decay model

the rotating turbine, the wake is generated while it is linearly expanded as the wake travels away from the turbine. Based on the conservative of momentum, the velocity assumed being uniform over a particular wake disk can be computed as [16]

$$1 - \frac{V(x)}{V_0} = a\left(\frac{D_0^2}{D(x)^2}\right) \tag{1}$$

where $V_0$ is the full wind velocity, $V(x)$ is wind wake velocity at the distance $x$ away from the turbine, $D_0$ is a turbine diameter, $D(x)$ is a wake diameter at the distance $x$ from the turbine. The left hand term is called the velocity deficit in the wake area. The parameter $a$ can be determined as

$$a = 1 - \sqrt{1 - C_T} \tag{2}$$

where $C_T$ is a turbine thrust coefficient. The wake diameter at the distance x can be computed as

$$D(x) = D_0 + 2kx \tag{3}$$

where the constant $k$ relies on hub height and terrain roughness of the wind farm. Figure 2 shows a particular wind turbine wake where $V_0 = 12$ m/s, $D_0 = 70$ m, $C_T = 0.88$, $\eta = 0.4$, $\rho_{air} = 1.225$ kg/m$^3$, $k = 0.086$ and wind direction is 30° about the $x$-axis. The variables $\eta$ and $\rho_{air}$ are respectively turbine efficiency and air density. It is seen that any turbine placed in the wake area will produce less energy compared to original turbine. In cases that the turbine is subject to many wakes from the upstream turbines, the total velocity deficit of the turbine can be computed as the summation of kinetic energy, which can be expressed as

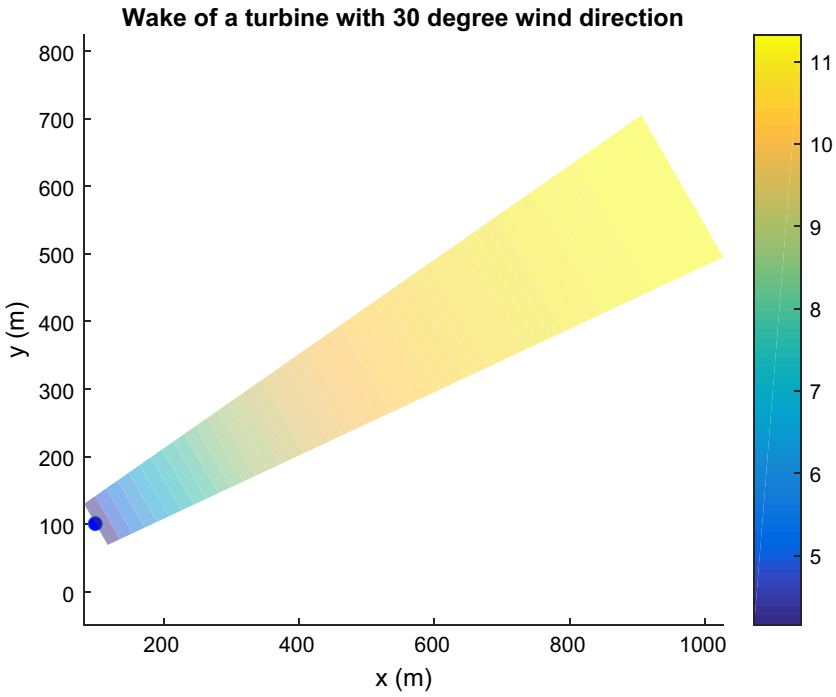


**Fig. 2** Wind velocity at a turbine wake

$$\left(1-\frac{V}{V_0}\right)^2=\sum_{i=1}^{N_{EC}}\left(1-\frac{V(x_i)}{V_0}\right)^2 \tag{4}$$

where $N_{EC}$ is the number of turbines whose wakes eclipse the considered turbine. Then the inflow velocity of the turbine can be computed using (4). The total power produced by the wind farm can then be estimated as:

$$P_{total}=\sum_{i=1}^{N_{WT}}\frac{1}{2}\eta_i\rho_{air}A_iV_i^3 \tag{5}$$

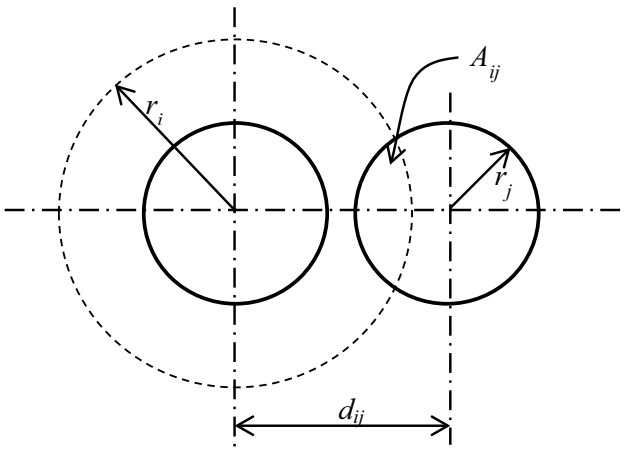


**Fig. 3** Intersection area

For simplicity, a particular turbine is said to be under the wake if the turbine hub is located inside the wake area. However, this may not be accurate as it is possible that the downstream turbine may partially shadowed by the upstream one as shown in Fig. 3 where the $i$-th turbine is upstream and the $j$-th turbine is downstream. The dashed circle is the wake diameter of turbine $i$ at the same plane as turbine $j$. it is illustrated that the turbine $j$ only partially shadowed by the wake. In order to more accurately predict the velocity deficit experienced by turbine $j$, the effect of the intersection are $A_{ij}$ can be accounted [36]. The area can be computed as [37]

$$A_{ij}=r_i^2\cos^{-1}\left(\frac{d_{ij}^2+r_i^2-r_j^2}{2d_{ij}r_i}\right)+r_j^2\cos^{-1}\left(\frac{d_{ij}^2+r_j^2-r_i^2}{2d_{ij}r_j}\right)-\frac{1}{2}\sqrt{(-d_{ij}+r_i+r_j)(d_{ij}-r_i+r_j)(-d_{ij}+r_i-r_j)(d_{ij}+r_i+r_j)} \tag{6}$$

where $d_{ij}$ is the distance between the centres of both turbines, $r_i$ is the wake radius of turbine $i$ at the same plane turbine $j$, and $r_j$ is the radius of turbine $j$. The velocity deficit at the $j$-th turbine can then be computed as

$$1-\frac{V(x)}{V_0}=a\left(\frac{D_0^2}{D(x)^2}\right)\left(\frac{A_{ij}}{A_0}\right) \tag{7}$$

where $A_0$ is the area of the disk of turbine $j$. Figure 4 shows the layout of 5 HAWTs where the wind turbines and wind conditions are those used in Fig. 2. In the figure, turbine 2 is partially eclipsed by the wake of turbine 1 while turbine 5 is partially shadowed by the wake of turbine 2. On the

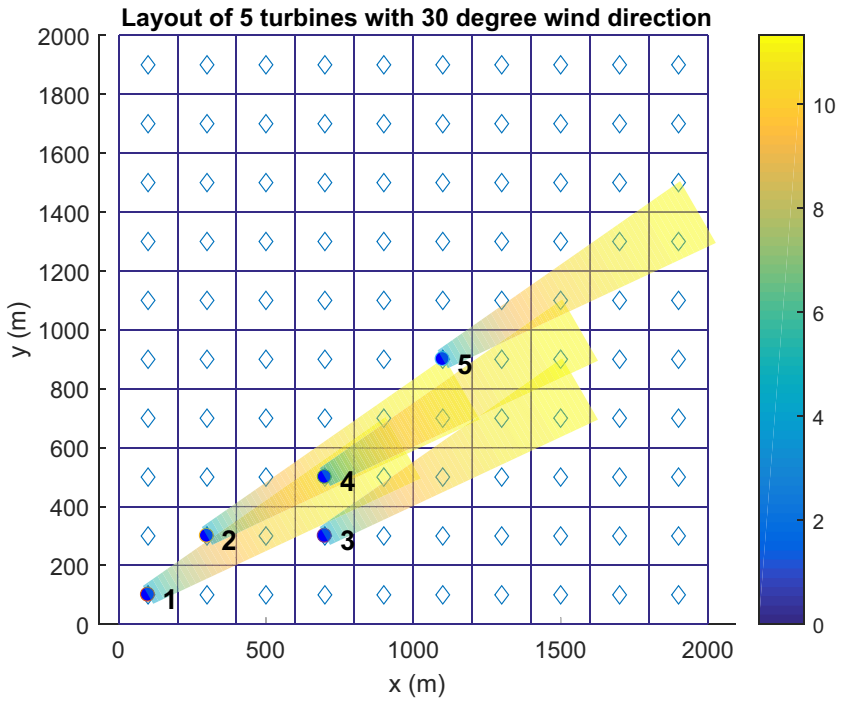


**Fig. 4** Layout of 5 wind turbines

other hand, turbine 4 is fully shadowed by turbines 1 and 2. If the first wake concept (Concept1) which defines the total eclipse if the centre of HAWT is placed in the wake area is employed, the total power is 6.29 MW. If the second wake concept (Concept2) where the partial eclipse is taken into account is used, the total power is 7.10 MW. This implies that the former concept underpredicts the total power produced by the wind farm. Both concepts will be used for comparative performance studies in this paper.

## 2.2 Optimisation Problem

The design problem for optimal wind farm layout can be posed to minimise cost, maximise total power, minimise noise, and other performance and cost indicators. Design variables determine the layout of the wind turbines. The grid-based approach is arguably the most popular WFLO approach due to its effectiveness and simplicity to implement. The idea is to create square grids covering the wind farm area while the grid centres are the choices for wind turbine installation. Normally, one type and one size of HAWT is used in the design problem [10, 38–40], however, it is possible to implement turbines with different hub heights [41, 42], and different turbines [14, 43, 43–47]. Moreover, uncertainties can be added to the design problem leading to robust design [48]. In this study, the WFLO problem is expressed as:

$$\text{Min}: \quad f(\mathbf{x}) = \frac{Cost}{P_{total}} \tag{8}$$

where $\mathbf{x}$ is vector of design variables, $P_{total}$ is the total power produced by a wind farm $\mathbf{x}$ as expressed in Eq. (5). The function *Cost* can be calculated as [15]

$$Cost = N_T\left(\frac{2}{3} + \frac{1}{3}e^{-0.00174N_T^2}\right) \tag{9}$$

This implies that the design problem is an attempt to simultaneously minimise wind farm cost and maximise wind farm totally produced power. In this work, the traditional $2000 \times 2000$ square meter grid area, discretised into $10 \times 10$ grids, is used for creating the WFLO test problems whereas the turbine characteristics are given in Table 1. The turbine locations are some of those 100 grid centres, which are confined to have the distance of at least $5D_0$ between turbines. Four WFLO problems are proposed based on the two power calculation concepts in Section in 2.1, which can be detailed as:

*Case I* the design problem is posed to find the locations of wind turbines on the $10 \times 10$ grid. The problem has 100 design variables for the 100 square grids while the number of wind turbines in the wind farm is allowed to be in between 1 and 100 turbines. 36 wind directions at a constant wind speed of 12 m/s are simulated while the total energy is calculated based on wake model with full eclipse (Concept1).

*Case II* It is Case I using the wake model taking into account partial eclipse (Concept2).

*Case III* The design problem is set to find the placements of 39 turbines [15]. The total number of design variables is 39 with 39 wind turbines being allowed to install while the range of position to be placed is [1, 100] meters as the $10 \times 10$ square grid in the Case I is used. 36 wind directions at a constant speed of 12 m/s is simulated whereas the total energy is calculated based on the first concept of wake model. The Concept1 wake model is used for total power calculation.

*Case IV* Case III using the Concept2 wake model.

**Table 1** characteristics of wind turbine

| | |
|---|---|
| Thrust coefficient ($C_T$) | 0.88 |
| Ground surface roughness ($h_0$) | 0.3 |
| Hub height ($h$) | 60 m |
| Rotor diameter ($D_0$) | 40 m |
| Turbine efficiency ($\eta$) | 40% |
| Air density ($\rho_{air}$) | 1.225 kg/m3 |
| Wind speed ($V_0$) | 12 m/s |
| Wisnd directions (degree) | {10°, 20°, 30°, …, 360°} wind direction angle w.r.t. + x (East) |

For encoding/decoding of design variables, the variables are assigned to be real numbers in the range of [0, 1] for the Case I and Case II, and in the range of [1, 100] for the Case III and Case IV. For the Case I and Case II, the elements of a design solution vector $\mathbf{x}$ sized $100 \times 1$ represent 100 positions in the $10 \times 10$ square grids. The vector $\mathbf{x}$, which all elements are in the range of [0, 1], are rounded to their nearest integers i.e. to be either 0 or 1. The elements which have the value 1 is the position for turbine placement. In cases that there is no turbine placed ($\mathbf{x}$ is a zero vector), the solution is defined as infeasible solution. The decoding for Case III and Case IV uses the decoding concept of the traveling salesman problem. The elements of a solution vector $\mathbf{x}$ sized $39 \times 1$ represent 39 turbines. Each element is in the range of [1, 100]. It is then rounded off to its nearest integer which represents the position to be selected in the $10 \times 10$ grid centres. A list containing integers $\mathbf{G}_c = \{1,.., 100\}$ is then created. If the value of the $i$-th element of the rounded-off $\mathbf{x}$ is lower than the number of remaining elements in $\mathbf{G}_c$, assign the grid number $\mathbf{G}_c(round(x_i))$ as the position of the $i$-th wind turbine and delete the selected grid position from the list $\mathbf{G}_c$. Otherwise, assign the last element of Gc as the

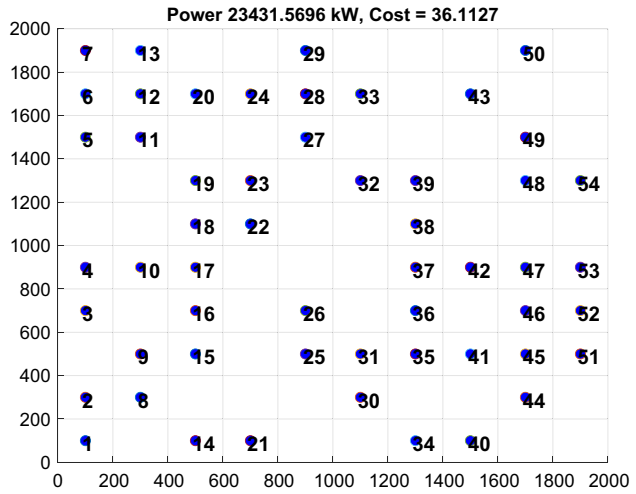


**Fig. 5** Example of turbine placement in the 2000×2000 square meter grid area and power calculated using Concept1

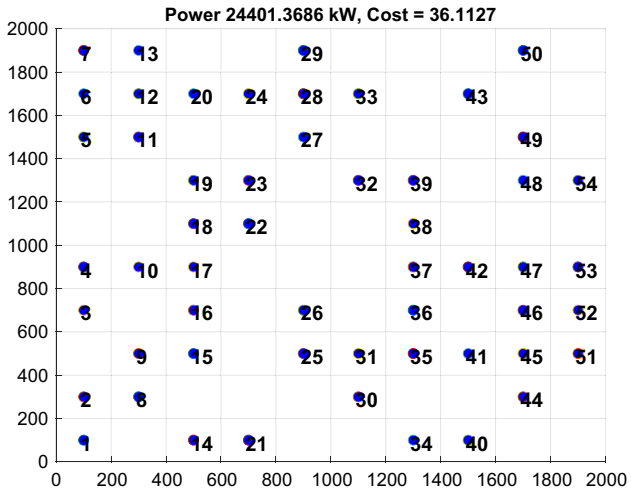


**Fig. 6** Example of turbine placement in the 2000×2000 square meter grid area and power calculated using Concept2

position of the *i*-th wind turbine and delete the selected grid position from the list $\mathbf{G}_c$. Once all 39 wind turbine positions are assigned on the 39 grid points of the 10×10 grid points, the wind farm layout is obtained. This ensures that any existing metaheuristic can be used to solve the WFLO problem without major modification. Besides, it is difficult for gradient-based optimisers to handle such design solution decoding. Algorithm 1 and Algorithm 2 shown the encoding/decoding of design variables for the Cases I & II and Case III & IV. Figures 5, 6 show examples of turbine placement in the 2000×2000 square meter grid area for Case I and Case II while the total power is calculated using the first and second wake concepts respectively.

Algorithm 1 encoding/decoding for Case I and Case II

Input $\mathbf{x} \in [\mathbf{0}, \mathbf{1}]$
Output $\mathbf{x}$
Main
1.$\mathbf{x}$=*round*($\mathbf{x}$)
2. If *sum*($\mathbf{x}$) ≥ 1
3. Return $\mathbf{x}$
4. Else
5. Return $\mathbf{x}$ as infeasible solution (the objective function value is infinite)
6. End

Algorithm 2 encoding/decoding for Case III and Case IV

Input $\mathbf{x} \in [\mathbf{1}, \mathbf{100}]$
Output **y** wind turbine placement vector
Main
1.$\mathbf{x}$=*round*($\mathbf{x}$)
2. Initialise a vector of centres of grids, $\mathbf{G}_c$= {1,…,100}.
3. For $i$ = 1 to 39
4. If $\mathbf{x}(i) \leq$ length ($\mathbf{G}_c$)
5. assign $\mathbf{y}(i) = \mathbf{G}_c(i)$
6. delete $\mathbf{G}_c(i)$ from the list $\mathbf{G}_c$.
7. Else
8. assign $\mathbf{y}(i) = \mathbf{G}_c(end)$
9. delete $\mathbf{G}_c(end)$ from the list $\mathbf{G}_c$.
10. End If
11.End For

## 3 Numerical Experiment

MH is one type of soft computing which is used for optimisation search. The method can be thought of as an optimisation algorithm exploiting a set of design solutions or a population for searching. The main operators of MH are reproduction for creating a new set of design offspring and selection for classifying solutions to the next generation or iteration. The optimiser of this kind has some obvious advantages in that it is simple to understand, code, and use [49]. The method is derivative-free and somewhat based on randomisation. As a result, it has less restrictions inside the algorithm compared to classical mathematical programming meaning that multiple versions of one original MH can be created without any violation. Over the last three decades, numerous (probably over one hundred) original MHs have been proposed. Counting all of their variants, there have been probably over one thousand algorithms in the literature. Up to the present time, only a few of them were implemented in WFLO, thus, this work is motivated by this reason. The main aim is to bridge the gap between the fields of MHs and WFLO. Comparative performance studies of MHs for a particular type of design problems are of importance since one MH algorithm can be efficient for a set of design problems but it is possible to be inefficient for another set of design problems. This situation is often referred to as the no free lunch theory, thus, comparison of many MHs for WFLO problems should be investigated. In this work, twelve established and recently invented MHs are used to solve the abovementioned WFLO test problems. The various MHs are briefly detailed as follows.

- Artificial bee colony method (ABC) [22]

ABC is one of the most popular MHs. The ABC algorithm mimics intelligent scavenging activities of honey bee swarms in nature. The three crowds of honey bees called employed bees, onlookers, and scouts that accomplish food-gathering activities. The ABC algorithm is more popular due to its high convergence rate, requires fewer controlling parameters, and solution quality.

- Real-code ant colony optimization (ACOR) [23]

ACOR is inspired by its original version called the ACO algorithm. The ACO algorithm is a nature-inspired method that works on the ants' foraging behaviour to searching for food. ACOR incorporated a continuous probability distribution whereas the original ACO used a discrete probability distribution. Thus, ACOR is an effective method to handle continuous and discrete design variables, both simultaneously and also separately.

- Differential evolution (DE) [24]

The differential evolution (DE) algorithm is a parallel direct search approach similar to a genetic algorithm GA with a specialized crossover mechanism. DE is a simple and

powerful global optimization heuristic method comprising high convergence speed, particularly for non-linear, and multi-modal continuous functions. The reproduction of DE are mutation and crossover, which have been modified leading to a great number of reproduction scheme where some of the most popular are DE/rand/2/bin and DE/current-to-best/1/bin. This method arguably has highest number of variants while some of their variants are regarded as the state-of-the-art such as Adaptive differential evolution (JADE) [50] and Success-history based adaptive differential evolution (SHADE) [51].

- Particle swarm optimization (PSO) [25]

PSO works on the social behaviour of a swarming flock towards food search. PSO is governed by optimum positions and velocities of the personal best and the global best particles. The method is one of the early swarm intelligent metaheuristics mimicking a flock of birds or fish seeking for food. Rather than reproducing a new population as with DE or a genetic algorithm, PSO starts with a set of solutions or particle, and updates the particle positions iteratively.

- Teaching–learning based optimization (TLBO) [26]

TLBO works on the teaching–learning in a classroom. Teacher's influence on the outcomes of learners in a classroom is formulated in the teacher phase. Learners also improve their grades from other learners, which is formulated in the learner phase. The TLBO algorithm does not need algorithm-specific governing parameters and is simply organized and easy to use MHs. Thus, this algorithm is widely applied to solve various problems.

- Covariance matrix adaptation evolution strategy (CMA-ES) [27]

The CMA-ES is one of the state-of-the-art evolution strategies, which shows favourable convergence behaviour for arduous non-linear, non-convex, and black-box optimization problems in a continuous domain when contrasted with other evolution strategies. Like other evolutionary methods, it comprised of three phases i.e. recombination, mutation, and selection which leads in a faster convergence to a global optimum in the least generations and time obscurity. It has characteristics like derivative-free, covariant, off-the-shelf, scalable which makes it fit for non-separable, ill-conditioned, multi-modal problems.

- Moth-flame optimization (MFO) algorithm [28]

The MFO algorithm framed the navigation techniques of moths, the group of insects, in nature. A moth follows a straight path due to the presence of natural moonlight called transverse orientation and it follows a spiral flying path in the presence of artificial light source. These characteristics are modelled mathematically to the MFO algorithm.

- Sine cosine algorithm (SCA) [29]

SCA mimics a mathematical model based on sine and cosine functions. The cyclic pattern of sine and cosine functions guide the exploration and exploitation of the search space. Thus, SCA searches new regions for the early period and moves towards a fine search of the explored regions with the progression.

- Whale optimization algorithm (WOA) [30]

WOA simulates the hunting activities of humpback whales, which works on the bubble-net hunting strategy. The humpback whales create bubbles in a circular or spiral-shaped path, called the bubble-net, to hunt small fishes. WOA comprises three activities i.e. coral loop, lobtail, and capture loop to update the solutions.

- Crow search algorithm (CSA) [31]

The CSA modelled the intelligent behaviour of crows that store excess food in a hidden place, retrieves the stored food when needed, and follow each other to do thievery.

- Salp swarm optimizer (SSO) [32]

The SSO algorithm emulates the swarming behaviour of salps that navigate and forage in oceans. In this methodology, the food source is treated equivalent to global optimum, towards which leading salps proceed. The leading salps are

**Table 2** Objective function values obtained for the Case I

| MHs | Mean | STD | Min | Max |
|---|---|---|---|---|
| ABC | 0.001468 | 0.000003 | 0.001463 | 0.001472 |
| ACOR | **0.001452** | **0.000001** | **0.001450** | **0.001453** |
| DE | 0.001459 | 0.000003 | 0.001455 | 0.001462 |
| PSO | 0.001476 | 0.000007 | 0.001470 | 0.001494 |
| TLBO | 0.001465 | 0.000003 | 0.001460 | 0.001470 |
| CMAES | 0.001457 | 0.000003 | 0.001453 | 0.001460 |
| MFO | 0.001456 | 0.000002 | 0.001453 | 0.001458 |
| SCA | 0.001483 | 0.000004 | 0.001478 | 0.001487 |
| WOA | 0.001483 | 0.000003 | 0.001479 | 0.001489 |
| CSA | 0.001464 | 0.000003 | 0.001460 | 0.001470 |
| SSA | 0.001464 | 0.000004 | 0.001459 | 0.001469 |
| GOA | 0.001481 | 0.000007 | 0.001467 | 0.001493 |

*Bold numbers represent the best results

**Table 3** Objective function values obtained for the Case II

| MHs | Mean | STD | Min | Max |
|---|---|---|---|---|
| ABC | 0.001425 | 0.000002 | 0.001422 | 0.001429 |
| ACOR | **0.001409** | **0.000000** | **0.001409** | **0.001410** |
| DE | 0.001416 | 0.000002 | 0.001413 | 0.001418 |
| PSO | 0.001431 | 0.000005 | 0.001423 | 0.001440 |
| TLBO | 0.001423 | 0.000003 | 0.001417 | 0.001426 |
| CMAES | 0.001414 | 0.000001 | 0.001411 | 0.001415 |
| MFO | 0.001413 | 0.000002 | 0.001410 | 0.001417 |
| SCA | 0.001437 | 0.000002 | 0.001435 | 0.001439 |
| WOA | 0.001435 | 0.000004 | 0.001430 | 0.001442 |
| CSA | 0.001421 | 0.000003 | 0.001416 | 0.001425 |
| SSA | 0.001424 | 0.000004 | 0.001417 | 0.001429 |
| GOA | 0.001433 | 0.000005 | 0.001426 | 0.001440 |

*Bold numbers represent the best results

**Table 4** Objective function values obtained for the Case III

| MHs | Mean | STD | Min | Max |
|---|---|---|---|---|
| ABC | 0.001458 | **0.000001** | 0.001456 | 0.001460 |
| ACOR | 0.001468 | 0.000002 | 0.001464 | 0.001470 |
| DE | 0.001467 | 0.000002 | 0.001464 | 0.001470 |
| PSO | 0.001461 | 0.000002 | 0.001458 | 0.001464 |
| TLBO | 0.001466 | 0.000002 | 0.001461 | 0.001469 |
| CMAES | 0.001458 | 0.000002 | 0.001456 | 0.001461 |
| MFO | **0.001457** | **0.000001** | **0.001455** | **0.001459** |
| SCA | 0.001474 | 0.000003 | 0.001468 | 0.001481 |
| WOA | 0.001469 | 0.000004 | 0.001465 | 0.001477 |
| CSA | 0.001476 | 0.000004 | 0.001468 | 0.001481 |
| SSA | 0.001465 | 0.000003 | 0.001461 | 0.001470 |
| GOA | 0.001465 | 0.000002 | 0.001462 | 0.001469 |

*Bold numbers represent the best results

**Table 5** Objective function values obtained for the Case IV

| MHs | Mean | STD | Min | Max |
|---|---|---|---|---|
| ABC | 0.001418 | **0.000001** | 0.001417 | **0.001419** |
| ACOR | 0.001427 | **0.000001** | 0.001426 | 0.001429 |
| DE | 0.001427 | **0.000001** | 0.001425 | 0.001429 |
| PSO | 0.001423 | 0.000003 | 0.001420 | 0.001427 |
| TLBO | 0.001426 | **0.000001** | 0.001423 | 0.001427 |
| CMAES | 0.001418 | **0.000001** | **0.001416** | 0.001420 |
| MFO | **0.001417** | **0.000001** | **0.001416** | **0.001419** |
| SCA | 0.001433 | 0.000002 | 0.001429 | 0.001437 |
| WOA | 0.001428 | 0.000003 | 0.001423 | 0.001432 |
| CSA | 0.001436 | 0.000003 | 0.001431 | 0.001439 |
| SSA | 0.001424 | 0.000002 | 0.001421 | 0.001427 |
| GOA | 0.001425 | 0.000002 | 0.001421 | 0.001429 |

*Bold numbers represent the best results

always followed by follower salps which result in automatic chasing movement of the whole chain of salp toward the global optimal solution.

- Grasshopper optimization algorithm (GOA) [33]

GOA mimics the swarming behavior of grasshoppers in nature. The original version of this algorithm requires a set of solutions to update their positions in an n-dimensional space, which is defined by the problem.

Each optimiser is used to solve all test problems for 10 independent runs. The Population size and number of iterations are set to be $n_P = 100$ and $n_{Iter} = 250$, respectively. For any optimisers using different population size, they will be terminate at $100 \times 250$ function evaluations (FEs). The details of the parameter settings of the optimisers are as follows:

- ABC: The number of food sources for employed bees is set to be $n_P/2$. A trial counter to discard a food source is 100.
- ACOR: The parameter settings are $q = 0.2$, and $\xi = 1$.
- DE: DE/best/2/bin strategy was used. A scaling factor, crossover rate and probability of choosing elements of mutant vectors are 0.5, 0.7, and 0.8 respectively.
- PSO: The starting inertia weight, ending inertia weight, cognitive learning factor, and social learning factor are assigned as 0. 5, 0.01, 2.8 and 1.3 respectively.
- TLBO: Parameter settings are not required.
- CMAES: Parameter settings are not required.
- MFO: The constant parameter, $b = 1$ while other parameters are iteratively adapted.
- SCA: The constant parameter, $a = 2$.
- WOA: The parameter $b = 1$ while other parameters are iteratively adapted.
- CSA: Awareness probability is set to be 0.1 while the flight length is set to be 2.
- SSA: All parameters are iteratively adapted.
- GOA: All parameters are iteratively adapted.

Note that the definitions of the variables can be seen from the references of the algorithms.

## 4 Results and Discussion

After performing 10 optimisation runs of the various MHs for solving the WFLO problems Case I–Case IV, the comparative results based on the objective function values are shown in Tables 2, 3, 4, 5. Based on the tables, the mean values of objective function values (Mean) is used to measure the MHs convergence rate and consistency. In cases that two MHs are equally good, the standard deviations of

**Table 6** Comparison of objective function values based on statistical Wilcoxon rank sum test

| MHs | (1) | (2) | (3) | (4) | (5) | (6) | (7) | (8) | (9) | (10) | (11) | (12) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ABC (1) | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| ACOR (2) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| DE (3) | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| PSO (4) | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| TLBO (5) | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| CMAES (6) | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| MFO (7) | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SCA (8) | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| WOA (9) | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| CSA (10) | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| SSA (11) | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| GOA (12) | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| Sum | 4 | 11 | 8 | 2 | 4 | 8 | 9 | 0 | 0 | 5 | 5 | 0 |
| Ranking | 7 | 1 | 3 | 9 | 7 | 3 | 2 | 10 | 10 | 5 | 5 | 10 |

**Table 7** Comparative objective function values based on statistical Wilcoxon rank sum test for all cases

| MHs | Case I | Case II | Case III | Case IV | Sum | Ranking |
|---|---|---|---|---|---|---|
| ABC | 7 | 6 | 1 | 2 | 16 | 3 |
| ACOR | 1 | 1 | 8 | 8 | 18 | 4 |
| DE | 3 | 4 | 8 | 8 | 23 | 6 |
| PSO | 9 | 9 | 4 | 4 | 26 | 8 |
| TLBO | 7 | 6 | 7 | 5 | 25 | 7 |
| CMAES | 3 | 2 | 1 | 2 | 8 | 2 |
| MFO | 2 | 2 | 1 | 1 | 6 | 1 |
| SCA | 10 | 11 | 11 | 11 | 43 | 12 |
| WOA | 10 | 11 | 8 | 8 | 37 | 11 |
| CSA | 5 | 5 | 11 | 12 | 33 | 10 |
| SSA | 5 | 6 | 5 | 5 | 21 | 5 |
| GOA | 10 | 9 | 5 | 5 | 29 | 9 |

*Bold numbers represent the best results

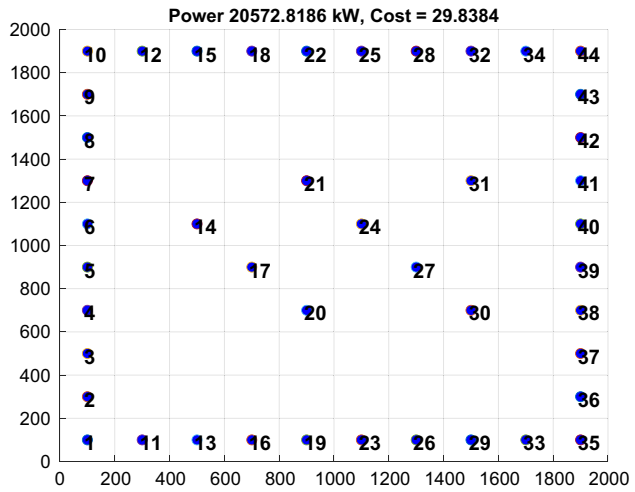


**Fig. 7** wind turbine placement, power output and cost obtained from the best run for WFLO Case I

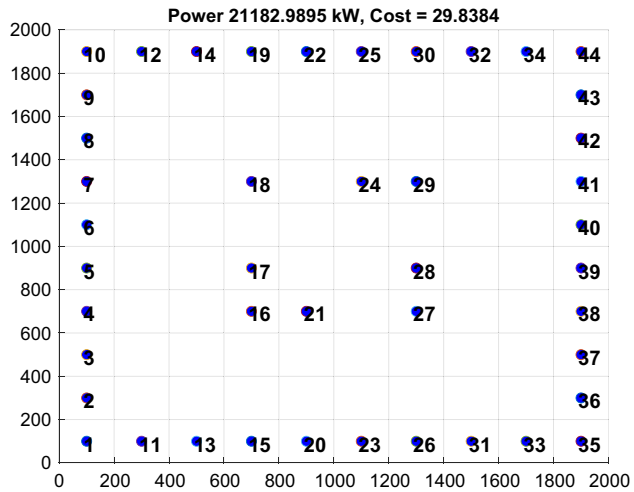


**Fig. 8** wind turbine placement, power output and cost obtained from the best run for WFLO Case II

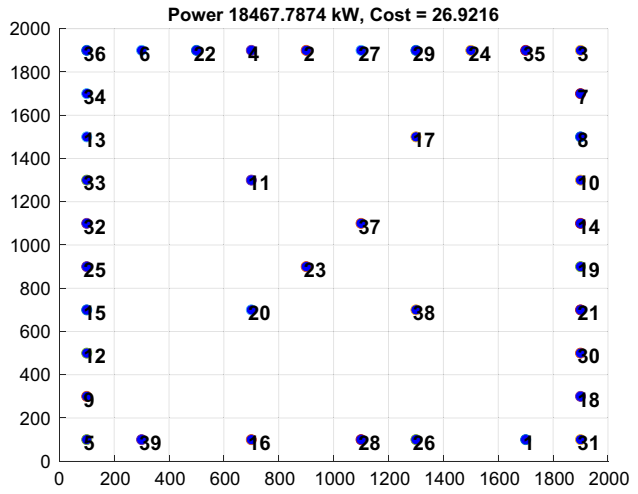


**Fig. 9** wind turbine placement, power output and cost obtained from the best run for WFLO Case III

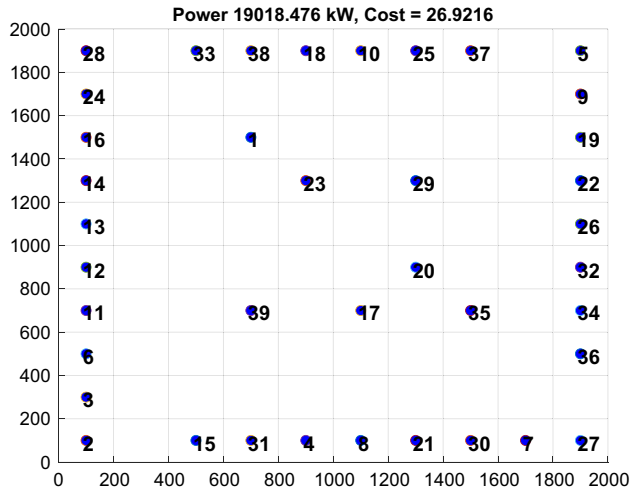


**Fig. 10** wind turbine placement, power output and cost obtained from the best run for WFLO Case IV

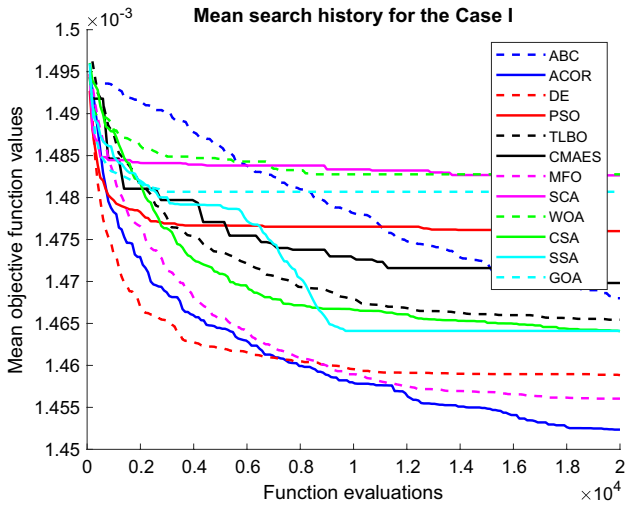


**Fig. 11** Search history of Case I

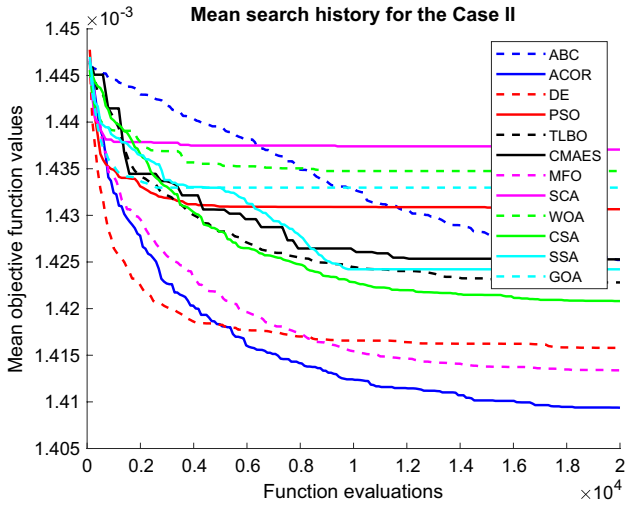


**Fig. 12** Search history of Case II

objective function values (STD) of both method will be used to decide the better performer. The lower mean value is the better convergence speed while the lower STD is the better search consistency.

For the measure of the search convergence, it is found that ACOR is the best performer for WFLO Case I and Case II while MFO is the best performer for WFLO Case III and Case VI. The second and third best algorithms for the Case I and Case II are MFO and CMAES, respectively. For the WFLO Case III and Case IV, the second best is ABC and CMAES which obtained the same mean objective value and slightly different STD. For the measure of search consistency based on the STD, the most consistent results for WFLO Case I and Case II is ACOR while the most consistent algorithm for the Case III is ABC and MFO which obtained the same STD values. For Case IV, the most consistent is ABC, ACOR, DE, TLBO, CMAES and MFO which obtained the same STD values. The best optimisation runs for the Case I and Case II are obtained from ACOR while the best optimisation run for the Case III is from MFO. For Case IV, the best optimisation run is obtained by MFO and CMAES which obtained the same objective function value.

In addition, the ranking of the MH optimisers was made based upon the statistical Wilcoxon rank sum test. Table 6 shows the ranking of MHs for the WFLO Case I. From 13 MHs implemented, a comparison matrix sized $13 \times 13$ whose elements are full of zeros is generated. The null

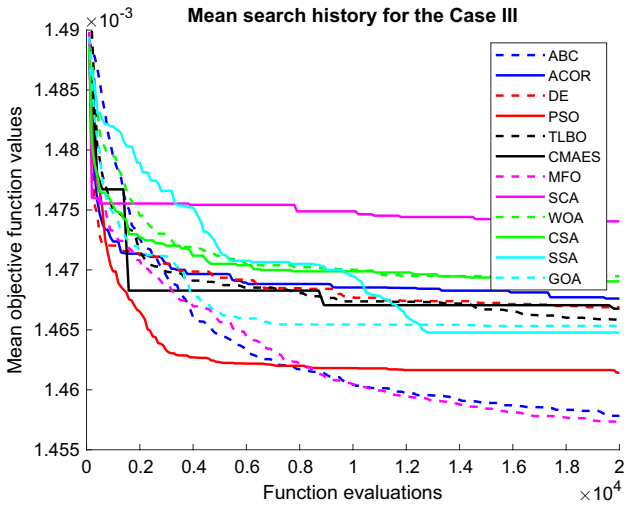


**Fig. 13** Search history of Case III

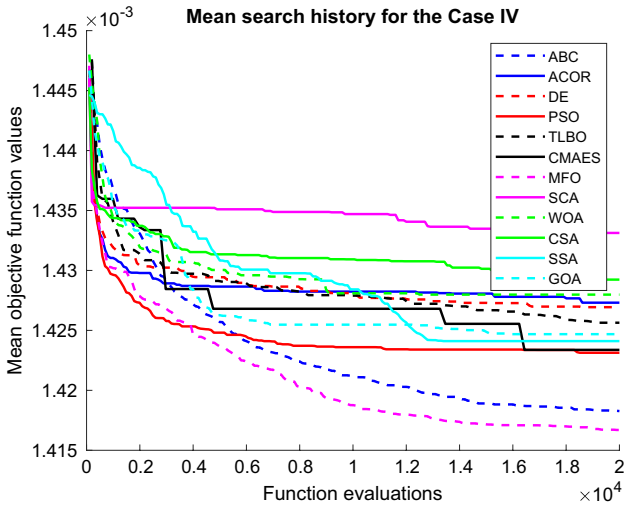


**Fig. 14** Search history of Case IV

hypothesis for the Wilcoxon rank sum test is that, for a particular test problem, the median of 10 objective function values obtained from method *I* is not different from the median of 10 objective function values obtained from using method *J* at the 5% significant level. If the null hypothesis is rejected and the median from method *J* is lower, the element *IJ* of the matrix is set to be '1'. After summing up all values in the matrix columns, the best optimiser is the one that has the highest score. The summation and the ranking are also given in the table. From the table, the best performer for the WFLO Case I is ACOR. The rankings for the all WFLO test problems are reported in Table 7. After summing up the ranks of all test problems, the MHs obtained the lowest score is the best performer. From the tables, the best method according to the median objective function values for all cases is MFO while the second best is CMAES.

Figures 7, 8, 9, 10 show the wind farm layouts, power output and cost obtained from the best runs of the all test problems. The layouts obtained from Concept1 and Concept2 wake models are slightly different while the total powers are different due to accounting for partial eclipse of Concept2. Case I and Case II give slightly better objective function values compared to Case III and Case IV respectively implying that adding the number of wind turbines as a design variable can improve the wind farm layout design. ACOR is most suitable for Case I and Case II where the number of turbines is varied whereas MFO is the best if the number of turbines is fixed during an optimisation process.

Figures 11, 12, 13, 14 show the search history as the plots of average best fitness from 10 runs versus the number of function evaluations of the implemented MHs for Case I, Case II, Case III, and Case IV respectively. For Case I, it is shown that DE converged fastest in the early stage of the optimisation run. Then, it was surpassed by ACOR and MFO. The optimisers can be classified into two groups as ACOR, MFO and DE are the methods with better convergence rate while the remainders are stuck at local optima. For Case II in Fig. 12, similar conclusions to Case I can be made.

In Fig. 13 which is Case III, MFO and ABC are separated from the rest with better convergence rates while PSO is an early bloomer but gets stuck to a local minima in the later stage of optimisation. For Case IV, the same conclusions as with Case III are obtained. PSO started faster but was trapped approximately at the mid-point of the optimisation run. MFO and ABC are clearly separated from the rest as the best optimisers. This information can be used to develop a new metaheuristic for solving WFLO in the future.

## 5 Conclusions

In this work, established and recently invented MHs are implemented on solving WFLO. Four WFLO test problems with and without varying the number of turbines in the wind farm, and taking into account wake partial eclipse for computing velocity deficit are posed. The comparative results that MFO is the overall best optimiser. In cases that the number of wind turbines is allowed to vary, ACOR is the best algorithm while MFO is the best if the number of wind turbines is fixed during an optimisation process. DE and PSO are also acceptable while the former converged faster than the rest in the early stage of the optimisation run, however, both were trapped at local optima during the runs. For future work, WFLO optimisation problem is proposed based in real area and wind data. The results from this study will

pave the way for hybridisation of several MHs' advantages so as to obtain a more powerful metaheuristic.

**Acknowledgements** The authors are grateful for the support from the Thailand Research Fund (RTA6180010).

## Declarations

**Conflict of interest** The authors declare that they have no conflict of interest.

## References


1. Reddy SR (2020) Wind farm layout optimization (WindFLO): an advanced framework for fast wind farm analysis and optimization. Appl Energy 269:115090. https://doi.org/10.1016/j.apenergy.2020.115090
2. Khan MJ, Mathew L (2020) Comparative study of optimization techniques for renewable energy system. Arch Comput Methods Eng 27(2):351–360. https://doi.org/10.1007/s11831-018-09306-8
3. Shu ZR, Li QS, Chan PW (2015) Investigation of offshore wind energy potential in Hong Kong based on Weibull distribution function. Appl Energy 156:362–373. https://doi.org/10.1016/j.apenergy.2015.07.027
4. Karad S, Thakur R (2019) Recent trends of control strategies for doubly fed induction generator based wind turbine systems: a comparative review. Arch Comput Methods Eng. https://doi.org/10.1007/s11831-019-09367-3
5. Hewitt S, Margetts L, Revell A (2018) Building a digital wind farm. Arch Comput Methods Eng 25(4):879–899. https://doi.org/10.1007/s11831-017-9222-7
6. Shourangiz-Haghighi A, Haghnegahdar MA, Wang L, Mussetta M, Kolios A, Lander M (2020) State of the art in the optimisation of wind turbine performance using CFD. Arch Comput Methods Eng 27(2):413–431. https://doi.org/10.1007/s11831-019-09316-0
7. Veisi AA, Shafiei Mayam MH (2017) Effects of blade rotation direction in the wake region of two in-line turbines using Large Eddy Simulation. Appl Energy 197:375–392. https://doi.org/10.1016/j.apenergy.2017.04.013
8. Gao X, Yang H, Lu L (2016) Optimization of wind turbine layout position in a wind farm using a newly-developed two-dimensional wake model. Appl Energy 174:192–200. https://doi.org/10.1016/j.apenergy.2016.04.098
9. Mosetti G, Poloni C, Diviacco B (1994) Optimization of wind turbine positioning in large windfarms by means of a genetic algorithm. J Wind Eng Ind Aerodyn 51(1):105–116. https://doi.org/10.1016/0167-6105(94)90080-9
10. Wang Y, Liu H, Long H, Zhang Z, Yang S (2018) Differential evolution with a new encoding mechanism for optimizing wind farm layout. IEEE Trans Ind Inf 14(3):1040–1054. https://doi.org/10.1109/TII.2017.2743761
11. Eroğlu Y, Seçkiner SU (2012) Design of wind farm layout using ant colony algorithm. Renew Energy 44:53–62. https://doi.org/10.1016/j.renene.2011.12.013
12. Feng J, Shen WZ (2015) Solving the wind farm layout optimization problem using random search algorithm. Renew Energy 78:182–192. https://doi.org/10.1016/j.renene.2015.01.005
13. Bilbao M, Alba E (2009) Simulated annealing for optimization of wind farm annual profit. In: 2009 2nd International symposium on logistics and industrial informatics, pp 1–5. https://doi.org/10.1109/LINDI.2009.5258656
14. Chowdhury S, Zhang J, Messac A, Castillo L (2013) Optimizing the arrangement and the selection of turbines for wind farms subject to varying wind conditions. Renew Energy 52:273–282. https://doi.org/10.1016/j.renene.2012.10.017
15. Patel J, Savsani V, Patel V, Patel R (2017) Layout optimization of a wind farm to maximize the power output using enhanced teaching learning based optimization technique. J Clean Prod 158:81–94. https://doi.org/10.1016/j.jclepro.2017.04.132
16. Katic I, Højstrup J, Jensen NO (1987) A simple model for cluster efficiency. In: EWEC'86, vol. 1, p 5
17. Frandsen S et al (2006) Analytical modelling of wind speed deficit in large offshore wind farms. Wind Energy 9(1–2):39–53. https://doi.org/10.1002/we.189
18. Larsen GC (2009) A simple stationary semi-analytical wake model. Risø National Laboratory for Sustainable Energy, Technical University of Denmark
19. Ishihara T, Qian G-W (2018) A new Gaussian-based analytical wake model for wind turbines considering ambient turbulence intensities and thrust coefficient effects. J Wind Eng Ind Aerodyn 177:275–292. https://doi.org/10.1016/j.jweia.2018.04.010
20. Bastankhah M, Porté-Agel F (2014) A new analytical model for wind-turbine wakes. Renew Energy 70:116–123. https://doi.org/10.1016/j.renene.2014.01.002
21. Xie S, Archer C (2015) Self-similarity and turbulence characteristics of wind turbine wakes via large-eddy simulation. Wind Energy 18(10):1815–1838. https://doi.org/10.1002/we.1792
22. Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Glob Optim 39:459–471. https://doi.org/10.1007/s10898-007-9149-x
23. Socha K, Dorigo M (2008) Ant colony optimization for continuous domains. Eur J Oper Res 185(3):1155–1173. https://doi.org/10.1016/j.ejor.2006.06.046
24. Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11(4):341–359. https://doi.org/10.1023/A:1008202821328
25. Kennedy J, Eberhart R (1995) Particle swarm optimization. In Proceedings of ICNN'95 international conference on neural networks, vol. 4, pp 1942–1948. https://doi.org/10.1109/ICNN.1995.488968
26. Rao RV, Savsani VJ, Vakharia DP (2011) Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput Aided Des 43(3):303–315. https://doi.org/10.1016/j.cad.2010.12.015
27. Hansen N, Müller SD, Koumoutsakos P (2003) Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evol Comput 11(1):1–18. https://doi.org/10.1162/106365603321828970
28. Mirjalili S (2015) Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl-Based Syst 89:228–249. https://doi.org/10.1016/j.knosys.2015.07.006
29. Mirjalili S (2016) SCA: a sine cosine algorithm for solving optimization problems. Knowl-Based Syst 96:120–133. https://doi.org/10.1016/j.knosys.2015.12.022
30. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008
31. Askarzadeh A (2016) A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm. Comput Struct 169:1–12. https://doi.org/10.1016/j.compstruc.2016.03.001

32. Mirjalili S, Gandomi AH, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM (2017) Salp Swarm algorithm: a bio-inspired optimizer for engineering design problems. Adv Eng Softw 114:163–191. https://doi.org/10.1016/j.advengsoft.2017.07.002
33. Saremi S, Mirjalili S, Lewis A (2017) Grasshopper optimisation algorithm: theory and application. Adv Eng Softw 105:30–47. https://doi.org/10.1016/j.advengsoft.2017.01.004
34. Wu X, Hu W, Huang Q, Chen C, Jacobson MZ, Chen Z (2020) Optimizing the layout of onshore wind farms to minimize noise. Appl Energy 267:114896. https://doi.org/10.1016/j.apenergy.2020.114896
35. Shakoor R, Hassan MY, Raheem A, Wu Y-K (2016) Wake effect modeling: a review of wind farm layout optimization using Jensen′s model. Renew Sustain Energy Rev 58:1048–1059. https://doi.org/10.1016/j.rser.2015.12.229
36. Sethi JK, Deb D, Malakar M (2011) Modeling of a wind turbine farm in presence of wake interactions. In: 2011 international conference on energy, automation and signal, pp 1–6. https://doi.org/10.1109/ICEAS.2011.6147144
37. Weisstein EW (2020) Circle-circle intersection. https://mathworld.wolfram.com/Circle-CircleIntersection.html. Accessed July 30 2020
38. Paul S, Rather ZH (2019) A new bi-level planning approach to find economic and reliable layout for large-scale wind farm. IEEE Syst J 13(3):3080–3090. https://doi.org/10.1109/JSYST.2019.2891996
39. Parada L, Herrera C, Flores P, Parada V (2017) Wind farm layout optimization using a Gaussian-based wake model. Renew Energy 107:531–541. https://doi.org/10.1016/j.renene.2017.02.017
40. Tao S, Kuenzel S, Xu Q, Chen Z (2019) Optimal micro-siting of wind turbines in an offshore wind farm using Frandsen–Gaussian wake model. IEEE Trans Power Syst 34(6):4944–4954. https://doi.org/10.1109/TPWRS.2019.2916906
41. Chen Y, Li H, Jin K, Song Q (2013) Wind farm layout optimization using genetic algorithm with different hub height wind turbines. Energy Convers Manage 70:56–65. https://doi.org/10.1016/j.enconman.2013.02.007
42. Vasel-Be-Hagh A, Archer CL (2017) Wind farm hub height optimization. Appl Energy 195:905–921. https://doi.org/10.1016/j.apenergy.2017.03.089
43. Abdulrahman M, Wood D (2017) Investigating the Power-COE trade-off for wind farm layout optimization considering commercial turbine selection and hub height variation. Renew Energy 102:267–278. https://doi.org/10.1016/j.renene.2016.10.038
44. Chen K, Song MX, Zhang X, Wang SF (2016) Wind turbine layout optimization with multiple hub height wind turbines using greedy algorithm. Renew Energy 96:676–686. https://doi.org/10.1016/j.renene.2016.05.018
45. Duan B, Wang J, Gu H (2014) Modified genetic algorithm for layout optimization of multi-type wind turbines. In: 2014 American control conference, pp 3633–3638. https://doi.org/10.1109/ACC.2014.6859416
46. Romanuke VV (2019) Iterative power maximization by one-half cost dichotomy for optimizing wind farm deployment. KPI Sci News. https://doi.org/10.20535/kpi-sn.2019.4.177315
47. Charhouni N, Sallaou M, Mansouri K (2019) Realistic wind farm design layout optimization with different wind turbines types. Int J Energy Environ Eng 10(3):307–318. https://doi.org/10.1007/s40095-019-0303-2
48. MirHassani SA, Yarahmadi A (2017) Wind farm layout optimization under uncertainty. Renew Energy 107:288–297. https://doi.org/10.1016/j.renene.2017.01.063
49. Turner SDO, Romero DA, Zhang PY, Amon CH, Chan TCY (2014) A new mathematical programming approach to optimize wind farm layouts. Renew Energy 63:674–680. https://doi.org/10.1016/j.renene.2013.10.023
50. Zhang J, Sanderson AC (2009) JADE: adaptive differential evolution with optional external archive. IEEE Trans Evol Comput 13(5):945–958. https://doi.org/10.1109/TEVC.2009.2014613
51. Tanabe R, Fukunaga A (2013) Evaluating the performance of SHADE on CEC 2013 benchmark problems. In: 2013 IEEE congress on evolutionary computation, pp 1952–1959. https://doi.org/10.1109/CEC.2013.6557798