# Adaptation of the Simulated Evolution Algorithm for Wind Farm Layout Optimization

**Salman A. Khan**
Professor
College of Computing & Info. Sciences
Karachi Institute of Economics and Technology, Karachi
Pakistan

*Wind energy is a potential replacement for traditional, fossil-fuel-based power generation sources. One important factor in the process of wind energy generation is to design of the optimal layout of a wind farm to harness maximum energy. This layout optimization is a complex, NP-hard optimization problem. Due to the sheer complexity of this layout design, intelligent algorithms, such as the ones from the domain of natural computing, are required. One such effective algorithm is the simulated evolution (SE) algorithm.*

*This paper presents a simulated evolution algorithm engineered to solve the wind farm layout design (WFLD)optimization problem. In contrast to many non-deterministic algorithms, such as genetic algorithms and particle swarm optimization which operate on a population, the SE algorithm operates on a single solution, decreasing the computational time. Furthermore, the SE algorithm has only one parameter to tune as opposed to many algorithms that require tuning multiple parameters. A preliminary empirical study is done using data collected from a potential location in the northern region of Saudi Arabia. Experiments are carried out on a 10 × 10 grid with 15 and 20 turbines while considering turbines with a rated capacity of 1.5 MW. Results indicate that a simulated evolution algorithm is a viable option for the said problem.*

***Keywords:*** *Wind farm layout design, Wind energy, Optimization, Artificial Intelligence, Simulated Evolution, Nature-inspired algorithms*

## 1. INTRODUCTION

The ever-increasing need for energy has put a tremen–dous burden on traditional power generation resour–ces.These resources mainly include oil, gas, and coal. The high level of utilization of these resources has not only resulted in their rapid depletion but has also caused environmental pollution. Therefore, for the last many years, efforts have been put into utilizing renewable energy sources. One effective and established source of renewable energy stems from the power of the wind. The use of wind energy for power generation serves many purposes. The energy is cost-effective and envi–ronmentally friendly [1]. In addition, wind energy is least affected by geo-political and transportation/logistic issues [1].

A major technical concern in the process of wind farm development is the pre-deployment feasibility study. The concern is that if the wind farm layout is not designed optimally, then the turbines generate less power than the expected output threshold. As a result, the cost-to-output ratio increases, which is not desired. Therefore, it is of utmost importance that the wind farm layout is designed to its optimal structure. However, the layout design process depends on the optimality of the wind turbine system. An efficient wind turbine system heavily relies on harmonizing several processes, as highlighted by Rašuo et al. [2]. These processes include design, materials and technology, manufacturing, verifi–cation testing, and regulations & standards [2]. All these processes work in conjunction with each other. From the viewpoint presented by Rašuo et. al. [2], it can be clearly implied that if the above processes are not harmonized, and any one of them is not carried out optimally, the overall performance of the wind farm is degraded, and the farm does not generate power to its maximum threshold.

The optimal layout of a wind farm is concerned with placing turbines in their most appropriate positions relative to each other within the wind farm. Studies [3-5] have shown that this is an optimization problem with hard (non-deterministic polynomial) complexity. It is opined that for NP-hard problems, algorithms with polynomial complexity (for that sake, less complex algorithms with linear or logarithmic complexity) can not solve the problem. While simple algorithms, such as linear search, are efficient in terms of execution, they can not produce the best layout. In some cases, where other technical factors constrain the problem, simple algorithms may fail to produce a feasible solution. In contrast, algorithms from the domain of natural computing, such as genetic algorithms (GA), differential evolution (DE), simulated annealing (SA), particle swarm optimization (PSO), and many others, have effectively solved the WFLD problem [3]. These algorithms can search the huge solution space associ–

Received: August 2022, Accepted: October 2022
Correspondence to: Prof. Salman A. Khan
College of Computing & Information Sciences,
Karachi Institute of Economics & Technology
E-mail: sakhan@kiet.edu.pk
**doi: 10.5937/fme2204664K**

ated with the underlying problem in a fairly reasonable execution time.

Nature-inspired algorithms are broadly classified into population-based and non-population-based algo–rithms [6]. Traditionally, most studies on the use of nature-inspired algorithms for the WFLD problem have focused on population-based algorithms [3]. Genetic algorithm (GA) so far is the most employed algorithm [3]. Other well-known algorithms from the population-based category include the PSO [7-9], cuckoo search [10,11], and differential evolution [12,13]. Some recent applications of natural computing algorithms to the WFLD problem have focused on biogeography-based optimization (BBO) [9] and ant colony optimi–zation (ACO) [9]. In contrast, only a simulated annea–ling algorithm has been used from the non-population-based domain [12]. A major drawback of population-based algorithms is that their execution time is generally higher than that of non-population-based algorithms. The reason is that the former maintains a set of solutions in an iteration and performs algorithmic operations on this set of solutions, while the latter maintains a single solution per iteration and perturbs this solution in every iteration. Generally, population-based algorithms pro–duce better results than non-population-based algorit–hms but require higher execution times. However, there are instances where algorithms from both categories produced comparable results when mutually compared [13], which signifies the advantage of the lower execution time of non-population-based algorithms, thus saving computational resources.

Among all nature-inspired non-deterministic algo–rithms, whether population-based or non-population-based, simulated evolution[14] is one such algorithm that makes it distinct from the rest. All nature-inspired algorithms work on the concept of *fitness evaluation* of a solution. A solution is comprised of individual ele–ments. For example, in GA, the solution is called a 'chromosome' and the individual elements are called 'genes'. While evaluating a solution, a fitness function does not measure the impact of individual elements but evaluates the solution as a whole. In contrast to this approach, a distinctive attribute of the SE algorithm is that in addition to measuring the fitness of the whole solution, the algorithm also measures the fitness of each individual element within the solution. This allows the algorithm to perform a better search within the solution space. This element-level fitness is formally referred to as goodness. Despite this unique feature, the SE algorithm has received limited attention from resear–chers, and only a few studies have reported its use to solve NP-hard problems [15-18].

Another important factor to consider in a nature-inspired algorithm is the algorithmic parameters. These parameters strongly impact the search capabilities of nature-inspired algorithms, leading to efficient solutions. Therefore, it is necessary to tune these para–meters to the best values. The number of algorithmic parameters is a reason to prefer SE over several other algorithms (whether population-based or non-population based). Typical algorithms that have been used for WFLD problems (such as genetic algorithms, particle swarm optimization, differential evolution, and simulated annealing, among others) have several parameters to tune to obtain optimal results. Finiding, the best combination of these parameters, is cumbersome and is itself classified as an NP-hard problem. In contrast, SE has only a single parameter known as *bias* that requires tuning. This significantly reduces the computational effort in the optimization process.

Motivated by the aforementioned observations, this paper aims to use the SE algorithm for the WFLD problem. Accordingly, the main contributions of this study are enumerated as follows:

1) The Simulated Evolution algorithm is adapted for the WFLD problem. More specifically, the *simulated evolution algorithm's evaluation selection* and *allo–cation* phases are modified to incorporate the WFLD problem model.

2) A problem-specific goodness function is deve–loped. This goodness function, which serves as the core of the evaluation phase, evaluates the performance of a turbine within the wind farm. The goodness function calculates the ratio between the power generated by the turbine in its current location compared to the turbine's rated power.

3) Simulations are done using data obtained from a site in Saudi Arabia. These simulations focus on the impact of the SE algorithmic parameter called *bias* on the quality of the final configuration generated.

The rest of the paper is organized as follows. Section 2 provides a literature review. This is followed by a description of the problem model in Section 3. The proposed simulation evolution algorithm is explained in Section 4. Results are presented and discussed in Section 5. The paper ends with a conclusion and future directions in Section 6.

## 2. LITERATURE REVIEW

Mosetti et al. [5] were the first to model and study the WFLD problem, which they solved using GA. They assumed a number of hypothetical wind scenarios and evaluated the performance of their adapted GA using these scenarios. Grady et al. [4] also employed GA using the same wind conditions as Mosetti et al. In another study, Emami and Noghreh [19] proposed a binary the GA, which generated quality results and reduced execution time. Gonzalez et al. [20] also utilized a GA while assuming investment risk as the optimization objective. The GA adopted by Wang et al. [21] considered physical landownership as the optimization objective. The purpose of the said Rehman et al. [11] adapted the cuckoo search (CS) algorithm to solve the WFLD problem while considering power and cost as the optimization objectives. The objective function was the minimization of the cost/power ratio. The objective was to show that in case of multiple ownership for a piece of land where a wind farm is to be developed, this approach effectively helps deal with infeasible solutions.

**Table 1. Summary of previous studies.**

| Reference | Year | Algorithm(s) | Optimization objective(s) |
|---|---|---|---|
| Mosetti et al. [5] | 1994 | Genetic Algorithms | Power, Cost |
| Grady et al. [4] | 2005 | Genetic Algorithms | Power, Cost |
| Huang [22] | 2009 | Genetic Algorithms | Annual profit |
| Emami and Noghreh [19] | 2010 | Genetic Algorithms | Power, Cost |
| Gonzalez et. al. [20] | 2010 | Genetic Algorithms | Net present value |
| Song et al. [23] | 2010 | Genetic Algorithms | Power, Constraint reduction |
| Rašuo et al. [12][13] | 2010 | Differential Evolution | Power efficiency, Investment costs |
| Eroğlu and Seçkiner [24] | 2013 | Ant colony optimization | Expected energy output |
| Wang et al. [21] | 2015 | Genetic Algorithms | Land ownership |
| Rehman et al. [11] | 2016 | Cuckoo Search | Power, Cost |
| Afanasyeva et. al. [10] | 2018 | Cuckoo Search, Genetic Algorithms | Net present value |
| Chahrouni et. al. [25] | 2019 | Genetic Algorithms | Power, Cost |
| Wang [26] | 2019 | Genetic Algorithms | Power, Cost |
| Gao et al. [27] | 2020 | Genetic Algorithms | Power, Cost, Wind Farm efficiency |
| Wu et. al. [28] | 2020 | Particle Swarm Optimization, Augmented Particle Swarm Optimization | Power |
| Shin et al. [8] | 2021 | Evolutionary Algorithm, Particle Swarm Optimization | Wake loss, AEP |
| Aggarwal et al. [9] | 2021 | Biogeography-based optimization, Genetic algorithms, Particle Swarm Optimization, Ant Colony Optimization | Energy output |
| Al Shereiqi [29] | 2021 | Genetic Algorithms | energy production, cable routing |
| Kirchner-Bossi [30] | 2021 | Genetic Algorithms, Hybrid GA | energy production, cable length |

A hybrid GA was proposed by Huang [22], who incorporated the hill-climbing property in GA. Song et al. [23]adapted the PSO algorithm and proposed a sensitivity index while evaluating power generation variation relative to wind direction variation. Their test scenarios considered both complex and flat terrains. Eroğlu and Seçkiner [24] used ACO with expected energy output as the optimization objective. In two separate studies, Rašuo et al. [12][13] used the differential evolution (DE) algorithm while considering the power efficiency and investment costs as the optimization objectives.

In two independent studies, Charhouni et al. [25] and Wang et al. [26] adapted GA with power and cost as the optimization objectives. Shin et al. [8] proposed a PSO-based approach as well as an evolutionary algorithm while optimizing wake loss and annual energy production. Al Shereiqi et al. [29] proposed a bi-objective GA with energy production and cable routing as the optimization objectives.

Afanasyeva et al. [10] adapted GA and CS algorithms while optimizing the Net Present Value. The results indicated that CS was able to produce better results than GA. Gao et al. [27] utilized a multi-population genetic algorithm with cost, power, and efficiency as the optimization objectives. A PSO algorithm and a modified variant were proposed by Wu et al. [28] with power as the objective to be optimized. Results indicated that the modified PSO performed better than the basic PSO, with better power output and a reduced runtime.

Aggarwal et al. [9] adapted numerous algorithms from the domain of evolutionary computation and swarm intelligence while optimizing the energy output. Kirchner-Bossi et. al. [30] developed severalhybrid GAs. Two optimization objectives were used: energy production and cable length. Results indicated that among the selected algorithms, BBO demonstrated the best performance.

Table 1 summarizes the previous studies with respect to the year of publication, algorithms used, and optimization objectives.

## 3. PROBLEM MODEL

As mentioned in Section 1, the complexity of the WFLD problem is NP-hard. In most studies, the problem is modelled as a discrete optimization problem. The problem requires placement of turbines in the best possible configuration so that the generated power is maximized. Asadopted by most previous studies on the WFLD problem (as cited in the literature review), the land in which the turbines are placed is marked as a square-shaped area, where this area is typically divided into 100 equal size sub-areas (also called *cells*) divided into a 10×10 grid, as shown in Figure 1. Each cell can accommodate a turbine in its center. This leads to having a maximum of $2^{100}$ possibilities of turbine placement.

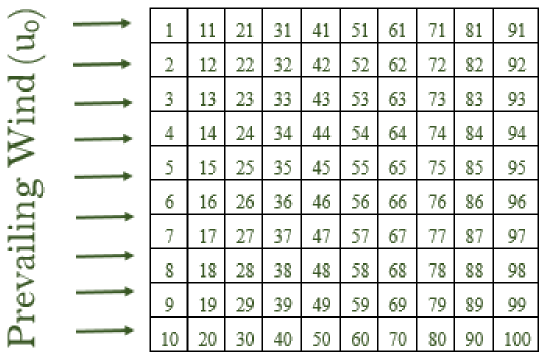

**Figure 1. Wind farm layout is divided into 10 x 10 grid with 100 cells, with each location identified by a number.**

An important consideration in evaluating the wind power generated by a turbine is the impact of the wake generated by other turbines. Turbines that are directly in front of the prevailing wind (such as locations 1 to 10 in Figure1) are not subjected to any wake. However, if turbine *i* has one or more turbines (directly in front or at an angle), then the wake generated by other turbines on the turbine *i* affects the absorption of wind by turbine *i*. Consequently, the power generated by turbine *i* is reduced. Therefore, it is important to calculate the wake as accurately as possible. Numerous wake effect models are used in relevant studies. One established method is the Jansen's wake effect model, which has been adopted by several studies [4,5,19-21]. The same model is assumed in this study. The model treats single and multiple wakes separately and addresses both scenarios through two different mathematical representations. A detailed discussion of the Jansen model can be found in Mosetti et al. [5].

According to Mosetti et al., we have

$$u_i = u_0 \tag{1}$$

If the turbine encounters a single wake, the wind speed is calculated as follows:

$$u_i = u_0\left[1-\frac{2a}{\left(1+\alpha\left(\frac{x_{ij}}{rd_0}\right)\right)^2}\right] \tag{2}$$

In the presence of multiple wakes, the resulting wind speed is determined as follows:

$$u_i = u_0\left[1-\sqrt{\sum_{j\in m_i}\left(\left[1-\frac{2a}{\left(1+\alpha\left(\frac{x_{ij}}{r_{d0}}\right)\right)^2}\right]\right)}\right] \tag{3}$$

The radius $r_{do}$ of the downstream wake immediately after a turbine is calculated using

$$r_{d0} = r_r\sqrt{\frac{1-a}{1-2a}} \tag{4}$$

The following equation is used to calculate the radius $r_{d1}$ of the wake at a distance $x_{ij}$ downstream of any wind turbine

$$r_{d1} = \alpha x_{ij} + rd_0 \tag{5}$$

The relationship between the axial induction factor and thrust coefficient is given by

$$C_t = 4a(1-a) \tag{6}$$

The thrust coefficient is normally known for the system. Therefore, the axial induction factor *a* can be calculated instead of $C_t$. The solution of Eq. (6) gives two values of *a*. The value which gives a real number for $r_{d0}$ in Eq. (4) is selected.

Finally, the entrainment factor *α* is found using:

$$\alpha = \frac{0.5}{\ln\left(z/z_0\right)} \tag{7}$$

If *N* turbines are placed in the grid, then the total power generated by these turbines under multiple wakes is calculated as follows

$$P_{actual} = \sum_i^N z_0 u_i^3 \tag{8}$$

The total power generated by *N* turbines without any wake is calculated as follows

$$P_{ideal} = \sum_i^N z_0 u_0^3 \tag{9}$$

The objective function requires maximization of efficiency of the wind power generation and is given by the following equation.

**Maximize**
Efficiency = $P_{actual} / P_{ideal}$ (10)

## 4. PROPOSED SIMULATED EVOLUTION ALGORITHM

This section discusses the SE algorithm for the WFLD problem. An overview of the basic SE algorithm is pro–vided first. This is followed by the proposed adaptation of the SE algorithm for the underlying problem.

### 4.1 Basic Simulated Evolution Algorithm

SE was originally proposed by Kling and Banerjee[16]. Inspired by the theory of evolution, the algorithm has several distinctive features as mentioned in Section 1. Despite its distinctive features and excellent perfor–mance, the algorithm and its hybrid variants have rece–ived little attention from researchers in some domains such as healthcare[31], internet traffic engineering [18], network design optimization [6,32], microelectronics [33-35], and cloud computing [17,36,37].

The basic SE algorithm has three main phases, which are executed iteratively until the stopping crite–rion is met. These phases are *evaluation*, *selection*, and *allocation*. In addition to the three phases, *initialization* occurs once an initial solution is generated. Each of the three main phases has a specific task to perform. Note that the SE algorithm evaluates a given solution both in terms of the individual element as well as a whole. The *evaluation* phase evaluates the individual elements usi–ng a goodness function which is represented as follows:

$$g_e = O_e / C_e \tag{11}$$

In Eq. (10), $g_e$ represents the goodness of individual elements. Goodness defines the ratio between the optimal cost ($O_e$) and current cost ($C_e$) of element *e*.

The formulation of the goodness function is problem specific and is defined by the problem solver. The goodness is calculated for each element in the solution.

The notion of goodness is that the more the current cost of an element is closer to its optimal cost, the higher its goodness.

The purpose of the *selection* phase is to segregate good elements from bad ones so that the bad elements (having low goodness) are reassigned to new positions within the solution structure in the hope of improving their goodness. This segregation is done using a func–tion that decides which elements in the current solution are to be reassigned to new positions. This reassignment is done in the allocation phase(discussed below). The following function makes the selection of elements to be reassigned:

$$\text{Random} > \text{Min}(g_e + B, 1) \quad (12)$$

In the above equation, parameter *B* controls the number of individual elements selected for reassign–ment. The values of B is user-defined and are set after experimentation. The value of *B* is defined between -1 and 1, but typically small values of *B*, somewhere between -0.2 and 0.2, are used. A higher value of *B* favors a small number of individual elements to be selected, while a lower value of *B* favors a larger number of elements chosen for reassignment.

Finally, in the *allocation* phase, the individual elements selected for reassignment are removed from their current positions one by one. Then, each selected element goes through a trial assignment at different positions. The position that gives the element the hig–hest goodness is fixed, and the element is placed at that position. This process is repeated for all selected ele–ments. At the end of this perturbation process, a new solution is obtained. This new solution is then evaluated using a fitness function.

## 4.2 Proposed Algorithm

For the WFLD problem, the basic SE algorithm needs to be tailored according to the problem structure. Details of these adaptations and other relevant issues are discussed below.

### *4.2.1 Solution Structure*

A solution in SE depicts a layout. This layout is visually represented in Figure 1, where the land area is divided into a 10×10 grid, mapping onto 100 potential locations. Each location in this grid is called a cell and can have a turbine in it or not. In terms of algorithmic design, this grid is programmed as a one-dimensional array with binary data. That is, if a turbine is present in a cell, then it is represented by a '1', while the absence of a turbine is identified by a '0'. The index of the array represents the position in the grid. An example solution is given in Figure 2.

| **Cell** | 1 | 2 | 3 | 4 | 5 | …. | 100 |
|---|---|---|---|---|---|---|---|
| **Turbine** | 0 | 1 | 1 | 0 | 1 | …. | 1 |

**Figure 2. An example solution**

### *4.2.2 Initialization*

The initial solution is generated randomly. That is, '1' and '0's are randomly assigned to the 100 cells. During the construction of the initial solution, a check is performed to ensure that the number of '1's is the same as defined by the user. For example, if 20 turbines are to be placed in the grid, then the generated initial solution should contain exactly 20'1's. If this is violated, then the solution is marked invalid. Once the initial solution is generated, its fitness is evaluated using Eq. (2).

### *4.2.3 Evaluation*

During the evaluation phase, the goodness of each element is evaluated, as explained in Section 4.1. In the case of the underlying problem, an element is a turbine. Furthermore, the evaluation function has to be defined according to the problem domain. For this purpose, the goodness function for turbine *i* is defined as follows

$$g_i = P_{i\_current} / P_{i\_rated} \quad (12)$$

In Eq. (12), $P_{i\_current}$ is the power generated by tur–bine *i* in its current placement within the grid, while $P_{i\_rated}$ is the turbine's rated power (rated power is the max threshold power that a turbine can ideally produce). The interpretation of the above goodness function is that if the current power generated by the turbine is close to its rated power, then the turbine is near its optimal pla–cement, and vice versa. Once the goodness of each turbine is calculated, the selection step is carried out.

### *4.2.4 Selection*

In this stage, for each turbine *i*, (where *i* = 1,.....*N*) in the current placement configuration, a random number RANDOM in the range [0,1] is generated. This number is then compared with $g_i + B$. If the condition as given in Eq. (11) is satisfied, then the turbine is selected for reassignment; otherwise, the turbine retains its position. However, the value of bias is important here. A high bias value would cause more turbines to be selected for reassignment, and vice versa.

### *4.2.5 Allocation*

During the allocation phase, each turbine selected in the selection phase is tried one by one at different available cells (i.e., empty cells without a turbine) within the grid. As an example, consider the scenario in Figure3, which shows a turbine (highlighted by black) in its current position. Assume that this turbine has been selected for reassignment during the *evaluation* phase. There are four possible locations for this cell to be reassigned, shown in red. Note that the approach adopted in this study is to consider cell positions adjacent to the tur–bine's current position. In an ideal condition, if all adja–cent four locations are available, then trial reassign–ments are done on all four positions. After each trial reassignment, the solution's fitness is evaluated using Eq. (10). The trial position, which results in the highest fitness value among the four trials, is fixed for the turbine, and the turbine is moved to the new position. It may so happen that one or more of the adjacent cells are already occupied by another turbine (s). In this case, only the available adjacent positions are tried. In the worst case, if all four positions are already occupied, then the turbine is not moved and remains in its current position.

The above process of reassignment is repeated for each selected turbine until all selected turbines are reassigned to new positions. The fitness of the whole new solution is then calculated using Eq. (10).

## 5. RESULTS AND DISCUSSIONS

The proposed algorithm was tested using real data for the area of Turaif located at a height of 827 meters (aboveground level) in the northern region of Saudi Arabia. The area has an average wind speed of 6.49 m/s at the height of 80 m [38]. Only one wind scenario was assumed, which was constant wind speed from one direction only. Furthemore, only one type of turbine was assumed, and Acciona AW 70/1500 was considered for simulations. A grid size of 10 × 10 was assumed, as shown in Figure1, with an area of 2.3716 sq. km (1.54 km × 1.54 km). It was also assumed that turbines are placed in the middle of the cell. Experiments were carried out with two scenarios where 20 and 15 turbines were considered. Table 2 gives the turbine specifi–cations and other relevant information.

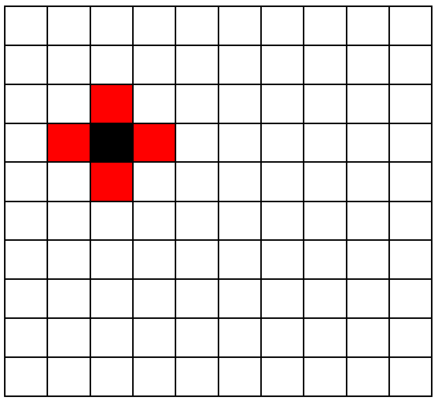

**Figure 3. Example of possible moves for reassignment**

**Table 2. Turbine and Site specifications used in the study.**

| | |
|---|---|
| Turbine | Acciona AW 70/1500 |
| Hub Height | 80 m |
| Turbine Diameter | 70 m |
| Turbine Cut-in Speed | 4 m/s |
| Turbine Cut-out Speed | 25 m/s |
| Turbine Rated Speed | 11.6 m/s |
| Turbine Rated Power | 1.5 MW |
| Axial Induction Factor | 0.09 |
| Entrainment Factor | 0.15 |
| Surface Roughness Length | 0.3 m |
| Thrust Coefficient | 0.3276 |

Since bias $B$ is the only algorithmic parameter in SE, the impact of this parameter on the algorithm's performance was assessed through parameter sensitivity analysis. Accordingly, simulations were carried out with five different values of bias. These values were 0, 0.05, 0.1, 0.15, and 0.2. Also note that SE is a non-deter–ministic algorithm, which means that each simulation run can result in a different best solution. Therefore, for each bias value, 30 independent runs were carried out following the procedure adopted in similar previous studies [4,18]. Furthermore, after several trial runs with a different number of iterations, it was observed that the algorithm converges within 300 iterations for all bias values. Therefore, all runs were executed for 300 iterations. Moreover, an average of 30 runs along with the standard deviation were reported. Simulations were carried out using Intel Core i5 processor with 32 GB RAM. A pre-generated initial solution was used in all simulations for fair comparisons.

**Table 3: Effect of bias on efficiency using 20 turbines**

| Bias | Maximum Efficiency | Minimum Efficiency | Average Efficiency |
|---|---|---|---|
| 0 | 0.799 | 0.747 | 0.774 ± 0.0114 |
| 0.05 | 0.802 | 0.750 | 0.774 ± 0.0113 |
| 0.1 | 0.801 | 0.753 | 0.770 ± 0.0136 |
| 0.15 | 0.791 | 0.735 | 0.754 ± 0.0137 |
| 0.2 | 0.804 | 0.714 | 0.749 ± 0.0176 |

**Table 4: Effect of bias on efficiency using 15 turbines**

| Bias | Maximum Efficiency | Minimum Efficiency | Average Efficiency |
|---|---|---|---|
| 0 | 0.896 | 0.868 | 0.883 ± 0.007 |
| 0.05 | 0.894 | 0.858 | 0.878 ± 0.008 |
| 0.1 | 0.893 | 0.867 | 0.880 ± 0.007 |
| 0.15 | 0.896 | 0.854 | 0.877 ± 0.010 |
| 0.2 | 0.887 | 0.847 | 0.871 ± 0.011 |

Table 3 shows the results obtained with different bias values for 20 turbines. The table shows the best efficiency of the farm averaged over 30 runs, along with the maximum and minimum efficiency of 30 runs. The standard deviation of the 30 runs is also given. The results indicate that, on average, low bias values (B = 0.0 and B = 0.05) generated the best results, with an average efficiency of 0.774 (or 77.4 %). Furthermore, the standard deviation was also almost equal (i.e., 0.113 and 0.114). In contrast, higher bias values with B = 0.1 to 0.2 generated layouts with lower efficiency, while having higher standard deviations. In particular, the worst results were produced by B = 0.2, where the average efficiency was lowest among all results (0.749) while having the highest standard deviation of 0.0176 among the results of all bias values. As far as layouts with 15 turbines are concerned, results in Table 4 reflect somewhat similar trends as observed for 20 turbines. The best results were obtained with B = 0.0, with the highest average efficiency of 0.883 and the lowest standard deviation of 0.007. Again, a high bias such as B = 0.2 produced the worst results in terms of efficiency and standard deviation.

Apart from better solutions by the SE algorithm at low bias values, the results in Tables 3 and 4 also highlight the algorithm's stability at low bias values. It is observed from both tables that when bias was low (say B =0.0), the standard deviation was also low. This indicates that every time the algorithm is run, the algorithm is able to reach the same quality of solution, indicating that the algorithm had a stable behavior in converging towards a high-quality solution. In contrast, as the bias value increased from low to high, the standard deviation kept increasing. This indicates that at high bias values, the algorithm had a somewhat unstable behavior; therefore, every time the algorithm was run, it reached a different quality of solutions.

Figures 4 and 5 provide the frequency of solutions in different ranges to further analyze the impact of bias on the quality of solutions. Note that the SE algorithm was run 30 times at each bias setup for both 15 and 20 turbines. Thus, each bias setup produced 30 final solu–tions. These solutions were categorized into different efficiency ranges. For example, for 20 turbines, the plots in Figure 4 indicate that for Bias = 0.0, a reaso–nably good number of solutions were in the range of efficiency above 0.78 (identified by orange legend). As the bias values increased, the high-efficiency solutions kept reducing, with B = 0.15 and B = 0.2 having a negligible number of high-quality solutions (identified by a small count of orange legend). In the same sense, the low-quality solutions (such as the efficiency range of 0.7 to 0.719 and 0.72 to 0.739) were non-existent for low and medium bias values (B = 0.0, 0.05, and 0.1) but were observed for high bias values of 0.15 and 0.2. Similar observations can be made about results for 15 turbines in Figure 5, where low bias values show more solutions in the high-efficiency range, while high bias values produce results with more solutions with low-efficiency values. From the observations in Tables 4 and 5, it can be fairly claimed that the SE algorithm could produce more solutions with higher efficiencies when bias was kept low.

The better performance of low bias values can be explained as follows. At a low value of bias, more ele–ments(turbines) are selected for reassignment. This cau–ses a higher level of exploration in the existing solution, thus taking the algorithm to the other search space sub-regions and avoiding trapping in local optima. In con–trast, higher bias values, such as 0.15 and 0.20, result in fewer turbines selected for reassignment. This, in turn, results in less exploration (and thus favors more explo–itation), which decreases the algorithm's capability to get out of local optima. As a result, the algorithm keeps traversing the same search space for a longer time.

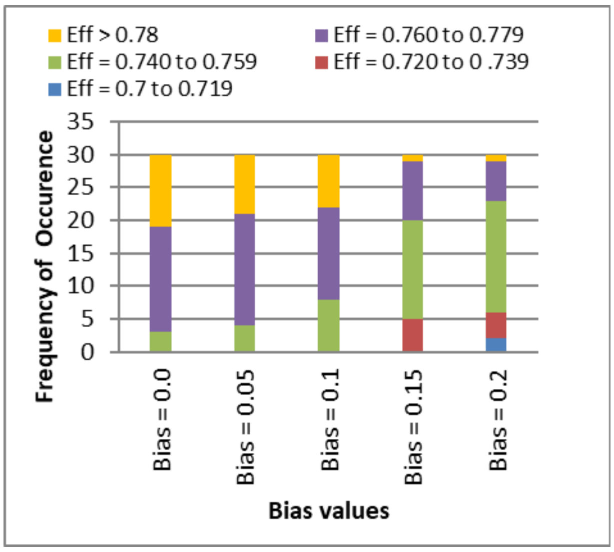

**Figure 4. Frequency of solutions for different bias values, 20 turbines. Eff = efficiency.**

What is also observed from Tables 3 and 4 (and complemented by Figures 4 and 5) is that the range of efficiencies for 15 and 20 turbines is different. For 20 turbines, the efficiencies range between 0.71 and 0.8. With 15 turbines, this range is between 0.847 and 0.896. This indicates that when the number of turbines is less, the efficiency is higher (this should not be confused with the power output, as the power generated with 20 turbines will be more when compared with 15 turbines). When the turbines are increased, the efficiency deg–rades. This is logical since the grid size is fixed (i.e., 2.3716 sq. km), and a low number of turbines have more slots for reassignment. For a higher number of turbines, the area (i.e., the number of cells) is still the same; therefore, there are fewer possibilities for turbine reassignment.

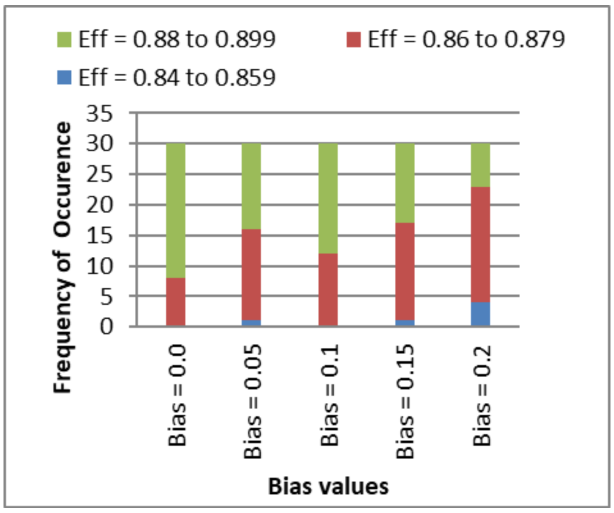

**Figure 5. Frequency of solutions for different bias values, 15 turbines. Eff = efficiency.**

## 6. CONCLUSIONS AND FUTURE DIRECTIONS

The high level of complexity in wind farm layout design makes the use of nature-inspired algorithms inevitable. While a number of population-based natural computing algorithms have been applied to the problem in the past, the use of non-population-based algorithms has been limited. This study applied the SE algorithm to the WFLD problem while using efficiency as the optimi–zation objective. A preliminary empirical analysis was carried out using data from a potential site in the northern region of Saudi Arabia. The impact of bias *B*, which is the only algorithmic parameter in the SE algorithm, was analyzed with respect to the quality of solutions generated. The results averaged over 30 runs indicated that overall, a lows bias value, particularly B = 0.0, produced the best results than the other values tried. Overall, the preliminary analysis in this study is enco–uraging and solicits the need to have an in-depth analysis of the SE algorithm with more test scenarios. For in-depth testing, data from several other sites from Saudi Arabia as well as other global locations can be used, and results can be mutually compared for different trends. Most of the required data is already available in the public domain or can be obtained from competent authorities.

Based on the work presented in this study, several potential research directions may be identified both in terms of problem model and algorithmic aspects, as follows:

- Different wake effect models can be tested with the SE algorithm, and the impact on the quality of solutions can be mutually compared.

- Instead of optimizing a single objective, several objectives may be simultaneously optimized, thus making the problem model multi-objective optimization. This will require the SE algorithm to be adapted for multi-objective optimization.
- Another direction is the development of a *dyna–mic bias* where the bias can adjust itself automa–tically based on the algorithm's performance, thus reducing or alleviating the need for a human user to define the bias value.
- Furthermore, the SE algorithm can be compared with other population and non-population-based algorithms to evaluate the quality of solutions and the execution time.
- Studies have shown that the hybridization of algorithms generally improves the performance of the algorithm. Thus, the hybridization of SE with other algorithms can also be explored.

## ACKNOWLEDGMENT

This research was supported through a grant by Prince Mohammad bin Fahd Center for Futuristic Studies (PMFCFS) at Prince Mohammad bin Fahd University, Saudi Arabia, and WFSF.

## REFERENCES

[1] S. Rehman, S. A. Khan, and L. M. Alhems, "A rule-based fuzzy logic methodology for multi-criteria selection of wind turbines, "Sustainability, vol. 12, no. 20, p. 8467, 2020.

[2] B. Rašuo, M. Dinulović, A. Veg, A. Grbović, and A. Bengin. Harmonization of new wind turbine rotor blades development process: A review, Renewable and Sustainable Energy Reviews, 39, 874-882. 2014.

[3] S. A. Khan and S. Rehman, "Iterative non-deterministic algorithmsin on-shore wind farm design: A brief survey," Renewable and Sustainable Energy Reviews, vol. 19, pp. 370–384, 2013.

[4] S. Grady, M. Hussaini, M. M. Abdullah, "Place–ment of wind turbines using genetic algorithms," Renewable energy, vol. 30, no. 2, pp. 259–270, 2005.

[5] G. Mosetti, C. Poloni, 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.

[6] S. A. Khan, "Design and analysis of evolutionary and swarm intelligence techniques for topology design of distributed local area networks," Ph.D. dissertation, University of Pretoria, 2009.

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

[8] J. Shin, S. Baek, and Y. Rhee, "Wind farm layout optimization using a metamodel and ea/pso algorithm in Korea offshore," Energies, vol. 14, no. 1, p. 146, 2020.

[9] S. K. Aggarwal, L. M. Saini, and V. Sood, "Large wind farm layout optimization using nature inspired meta-heuristic algorithms,"IETE Journal of Research, pp. 1–18, 2021.

[10] S. Afanasyeva, J. Saari, O. Pyrh¨onen, and J. Partanen, "Cuckoo search for wind farm optimization with auxiliary infrastructure,"Wind Energy, vol. 21, no. 10, pp. 855–875, 2018.

[11] S. Rehman, S. S. Ali, and S. A. Khan, "Wind farm layout design using cuckoo search algorithms," Applied Artificial Intelligence, vol. 30, no. 10, pp. 899–922, 2016.

[12] B. Rašuo, A. Bengin, and A. Veg, "On aerodynamic optimization of wind farm layout," PAMM, vol. 10, no. 1, pp. 539–540, 2010.

[13] B. Rašuo and A. Bengin, "Optimization of wind farm layout, "FME transactions, vol. 38, no. 3, pp. 107–114, 2010.

[14] M. Bilbao and E. Alba, "Chc and sa applied to wind energy optimization using real data," in IEEE congress on evolutionary computation. IEEE, 2010, pp. 1–8.

[15] T. W. Manikas and J. T. Cain, "Genetic algorithms vs. simulated annealing: a comparison of approaches for solving the circuit partitioning problem," 1996.

[16] R.M. Kling and P. Banerjee, "Optimization by simulated evolution with applications to standard cell placement," in Proceedings of the 27th ACM/IEEE Design Automation Conference, 1991, pp. 20–25.

[17] S. Sait and A. Raza, "Engineering simulated evolution for integrated power optimization in data centers," Soft Computing, vol. 22, no. 9, pp. 3033–3048, 2018.

[18] M. A. Mohiuddin, S. A. Khan, and A. P. Engelbrecht, "Simulated evolution and simulated annealing algorithms for solving multi-objective open shortest path first weight setting problem,"Applied intelligence, vol. 41, no. 2, pp. 348–365, 2014.

[19] A. Emami and P. Noghreh, "New approach on optimization in placement of wind turbines within wind farm by genetic algorithms,"Renewable Energy, vol. 35, no. 7, pp. 1559–1564, 2010.

[20] J. S. Gonzalez, A. G. G. Rodriguez, J. C. Mora, J. R. Santos, and M. B. Payan, "Optimization of wind farm turbines layout using an evolutive algorithm," Renewable energy, vol. 35, no. 8, pp. 1671–1681, 2010.

[21] L. Wang, A. C. Tan, Y. Gu, and J. Yuan, "A new constraint handling method for wind farm layout optimization with lands owned by different owners," Renewable Energy, vol. 83, pp. 151–161, 2015.

[22] H. S. Huang, “Efficient hybrid distributed genetic algorithms for wind turbine positioning in large wind farms,” in 2009 IEEE International Symposium on Industrial Electronics. IEEE, 2009,pp. 2196–2201.

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

[24] Y. Eroğlu, S. Seçkiner, “Wind farm layout optimi– zation usingparticle filtering approach,” Renewable Energy, vol. 58, pp. 95–107, 2013.

[25] N. Charhouni, M. Sallaou, and K. Mansouri, “Realistic wind farm design layout optimization with different wind turbines types,” International Journal of Energy and Environmental Engineering, vol. 10, no. 3, pp. 307–318, 2019.

[26] L. Wang, “Comparative study of wind turbine placement methods for flat wind farm layout optimization with irregular boundary,”Applied Sciences, vol. 9, no. 4, p. 639, 2019.

[27] X. Gao, Y. Li, F. Zhao, and H. Sun, “Comparisons of the accuracy of different wake models in wind farm layout optimization,”Energy Exploration & Exploitation, vol. 38, no. 5, pp. 1725–1741, 2020.

[28] X. Wu et al. “Optimizing the layout of onshore wind farms to minimize noise,”Applied Energy, vol. 267, p. 114896, 2020.

[29] A. Al Shereiqi, B. Mohandes, A. Al-Hinai, M. Bakhtvar, R. Al-Abri, M. El Moursi, and M. Albadi, “Co-optimisation of wind farm micrositing and cabling layouts,” IET Renewable Power Generation, vol. 15, no. 8, pp. 1848–1860, 2021.

[30] N. Kirchner-Bossi and F. Port´e-Agel, “Wind farm area shape optimization using newly developed multi-objective evolutionary algorithms,” Energies, vol. 14, no. 14, p. 4185, 2021.

[31] M. Mutingi, C. Mbohwa, “Multi-objective home– care worker scheduling: A fuzzy simulated evolu– tion algorithm approach,”IIE Transactions on Heal– thcare Systems Engineering, vol. 4, no. 4, pp. 209–216, 2014.

[32] H. Youssef, S. M. Sait, S. A. Khan, “Fuzzy evolu– tionary hybrid metaheuristic for network topology design”, In International Conference on Evoluti– onary Multi-Criterion Optimization, pp. 400-415. Springer, Berlin, Heidelberg, 2001.

[33] S. M. Sait et al, “Fsm state-encoding for area and power minimization using simulated evolution algorithm, ”Journal of applied research and technology, vol. 10, no. 6, pp. 845–858, 2012.

[34] A. H. El-Maleh, “Majority-based evolution state assignment algorithm for area and power opti– misation of sequential circuits,”IET Computers & Digital Techniques, vol. 10, no. 1, pp. 30–36, 2016.

[35] A. M. Arafeh and S. M. Sait, “Cells reconfiguration around defects in cmos/nanofabric circuits using simulated evolution heuristic,” in Sixteenth Inter– national Symposium on Quality Electronic Design. IEEE, 2015, pp. 581–588.

[36] S. M. Sait, K. Shahid, “Optimal multi-dimensional vector bin packing using simulated evolution,” The Journal of Supercomputing, vol. 73, no. 12, pp. 5516–5538, 2017.

[37] S. Sait, K. Shahid, “Engineering simulated evolu– tion for virtual machine assignment problem,” App– lied Intelligence, vol. 43, no. 2, pp. 296–307, 2015.

[38] S. Rehman, S. A. Khan, L. M. Alhems, L. M. Application of TOPSIS approach to multi-criteria selection of wind turbines for on-shore sites. Applied Sciences, vol 10, no. 21, 7595.

## NOMENCLATURE

| | |
|---|---|
| a | Axial induction factor |
| $z_0$ | Surface roughness |
| $u_0$ | Mean wind speed |
| Z | Hub height |
| $C_t$ | Thrust coefficient |
| $x_{ij}$ | Distance downstream from turbine j to turbine i (i.e., distance between the current turbine and the turbine creating wake effect on it) |
| $u_i$ | Wind speed downstream under multiple wakes |
| N | Total number of turbines |
| $m_i$ | Set of all turbines creating wake effect on turbine i |
| $r_{d0}$ | Wake radius immediately downstream of the wind turbine |
| $r_{dl}$ | Wake radius at a distance x downstream of the wind turbine |
| D | Rotor diameter |
| $P_{actual}$ | Total power generated by turbines |
| $P_{ideal}$ | Ideal power generated by turbines |
| B | Bias |
| $g_e$ | Goodness of element e |
| $O_e$ | Optimal cost of element e |
| $C_e$ | Current cost of element e |
| Random | Random number in the range [0,1] |
| $g_i$ | Goodness of turbine i |
| $P_{i\_current}$ | Current power generated by turbine i |
| $P_{i\_rated}$ | Rated power of turbine i |
| α | Entrainment factor |
| GA | Genetic algorithms |
| PSO | Particle swarm optimization |
| ACO | Ant colony optimization |
| BBO | Biogeography based optimization |
| DE | Differential evolution |
| SE | Simulated evolution |

## АДАПТАЦИЈА АЛГОРИТМА СИМУЛИРАНЕ ЕВОЛУЦИЈЕ ЗА ОПТИМИЗАЦИЈУ РАСПО– РЕДА ВЕТРОПАРКА

С.А. Кхан

Енергија ветра је потенцијална замена за тради– ционалне изворе енергије на бази фосилних горива. Један важан фактор у процесу производње енергије ветра је пројектовање оптималног распореда ветропарка како би се искористила максимална енергија. Ова оптимизација распореда је сложен, НП-тврд проблем оптимизације. Због саме сложености овог дизајна изгледа, потребни су интелигентни алгоритми, попут оних из домена природног рачунарства. Један такав ефикасан алгоритам је алгоритам симулиране еволуције (СЕ). Овај рад представља симулирани алгоритам еволу–ције пројектован да реши проблем оптимизације дизајна распореда ветропарка (ВФЛД). За разлику од многих недетерминистичких алгоритама, као што су генетски алгоритми и оптимизација роја честица који раде на популацији, СЕ алгоритам ради на једном решењу, смањујући време израчунавања. Штавише, СЕ алгоритам има само један параметар за подешавање за разлику од многих алгоритама који захтевају подешавање више параметара. Пре–лиминарна емпиријска студија је урађена кориш–ћењем података прикупљених са потенцијалне локације у северном региону Саудијске Арабије. Експерименти се изводе на мрежи 10 × 10 са 15 и 20 турбина уз разматрање турбина номиналног капаци–тета 1,5 МВ. Резултати показују да је симулирани алгоритам еволуције одржива опција за наведени проблем.