Mohamed Mohandes
Professor
Electrical Engineering Department
King Fahd University of Petroleum & Minerals, Dhahran 31261
Saudi Arabia

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

Shafiqur Rehman
Research Engineer II (Associate Professor)
Center for Engineering Research
Research Institute
King Fahd University of Petroleum & Minerals, Dhahran 31261
Saudi Arabia

Ali Al-Shaikhi
Professor
Electrical Engineering Department
King Fahd University of Petroleum & Minerals, Dhahran 31261
Saudi Arabia

Bo Liu
Assistant Professor
Electrical Engineering Department
King Fahd University of Petroleum & Minerals, Dhahran 31261
Saudi Arabia

Kashif Iqbal
Graduate Student
College of Computing & Info. Sciences
Karachi Institute of Economics and Technology, Karachi
Pakistan

GARM: A Stochastic Evolution based Genetic Algorithm with Rewarding Mechanism for Wind Farm Layout Optimization

Wind energy has emerged as a potential alternative to traditional energy sources for economical and clean power generation. One important aspect of wind energy generation is the layout design of the wind farm so as to harness maximum energy. Due to its inherent computational complexity, the wind farm layout design problem has traditionally been solved using nature-inspired algorithms. An important issue in nature-inspired algorithms is the termination condition, which governs the execution time of the algorithm. To optimize the execution time, appropriate termination conditions should be employed. This study proposes the concept of a rewarding mechanism to achieve optimization in termination conditions while maintaining the solution quality. The proposed rewarding mecha–nism,adopted from the stochastic evolution algorithm, is incorporated into a genetic algorithm. The proposed genetic algorithm with the rewarding mechanism (GARM) is empirically tested using real data from a potential wind farm site with different rewarding iterations.

Keywords: *Wind farm layout design, Wind farm micrositing, Wind energy, Optimization, Stochastic Evolution, Artificial Intelligence, Nature-inspired algorithms, Genetic Algorithms*

1. INTRODUCTION

The last two decades have seen significant growth in the use of renewable sources of energy, particularly wind and solar energy. This growth is motivated by several factors to reduce the dependency on traditional sources of fuel as well as to tackle the issue of environmental pollution [1]. Furthermore, the impact of geo-political conflicts on wind power generation is almost non–existent [2]. In addition, the least dependency on cross-border logistics and transportation is a plus point in wind energy. This issue is a major consideration when it comes to fossil fuels [2]. The cost-effectiveness of wind energy in contrast to oil or gas-based energy generation is another major advantage of the former [2].

Globally, wind power is considered a technolo–gically matured and commercially accepted technology [3]. Wind farms cover up a large geographical area, and therefore, the land area needs to be efficiently utilized so as to generate maximum power. Several factors govern the maximum power generation. One of the most important factors is the layout of the wind farm, or in otherwords, the micrositing of the wind turbines. The optimal layout guarantees maximum energy harnessing by turbines such that wake effects, turbulence, and other similar disturbances are reduced (if not completely avoided) as much as possible. The performance of each individual wind turbine is considered pivotal. The importance of design, manufacturing, and verification testing for a turbine's performance is emphasized by Rašuo et al. in several studies [4-6]. Furthermore, Rašuo et al. [7] also identified that an efficient wind turbine system depends on harmonizing various pro–cesses such as design, manufacturing, materials, tech–nology, regulations & standards, and verification tes–ting. These processes need to be completely harmonized with each other, and failing to do so would result in reduced efficiency of the wind turbines. This would consequently reduce the overall power generation capa-bility of the wind farm.

In computational terms, the wind farm layout opti–mization (WFLO) problem is classified as an NP-hard problem, as proven by several studies [8-10]. This means that simple approaches, such as linear search algorithms, cannot solve the WFLO problem. This turns researchers' attention towards algorithms that are

Received: July 2023, Accepted: September 2023
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/fme2304575M

inspired by natural phenomena and work in an iterative manner, such as the evolutionary computation (EC) algorithms or swarm intelligence (SI) algorithms. Some well-known algorithms from the said domains applied to the WFLOproblem include genetic algorithms (GA), differential evolution (DE), particle swarm optimization (PSO), ant colony optimization (ACO), and cuckoo search (CS), among others. These nature-inspired algo–rithms (NIAs) have proven their effectiveness in solving several variants of the WFLO problem. However, in contrast to linear search algorithms, a major drawback of EC and SI approachesis their high computational time requirement. EC and SI algorithms take a fair amount of time for execution to reach a quasi-optimal or optimal solution. From the computational point of view, the execution time should be optimized. That is, the algorithm should only execute appropriately within the required iterations. High execution time would result in higher energy consumption and processorover-utili–zation, which causes hardware stress. Moreover, since the processor is busy executing the code, a new processing task cannot be assigned to the processor, resulting in the wastage of computing resources.

Optimization of the number of iterations to achieve computational efficiency is a crucial issue. This con–cerns the fact that the number of iterations should be set at a value where over-utilizationor under-utilization of the execution time is not violated. Over-utilization of the algorithm time refers to a situation where the algorithm continues execution even after the best (or optimal) solution is obtained, thus resulting ina wastage of computational resources. Underutilization concerns the scenario where the algorithm stops prematurely be–fore reaching an optimal solution. Therefore, a mecha–nism is required capable of defining just the right number of iterations.

A study by Goreishi et al. [11] revealed that over 25 different stopping conditions have been proposed in the literature on NIAs. However, concerning the WFLO problem, only a few have been used. Table 1 shows a historical chronology of several studies that utilized NIAs for the WFLO problems. It is evident from the table that over 70% of the studies used a *fixed number of iterations*(FNI) as the termination criterion. A major concern with FNI is that it is a user-defined parameter, and finding the most appropriate value of FNI is done by trial and error. This causes a waste of computational time and effort. Furthermore, like FNI, other termi–nation criteria reported in the literature are based on the same fundamental principle; they require a stopping condition to be defined *a priori* by the user. Once the stopping condition is reached, the execution terminates.

Motivated by the above discussion, this study proposes a hybrid genetic algorithm. The contributions of the study are enumerated as follows:

1) Agenetic algorithm with a rewarding mechanism (GARM) that incorporates characteristics of the stoc–hastic evolution (StocE) algorithm is proposed. More specifically, the rewarding mechanism from the StocE algorithm is embedded into the genetic algorithm. This rewarding mechanism allows GA to add reward iterations when an improvement is observed in the quality of the solution.

2) The proposed GARM is empirically tested with different rewarding iterations, using real data from a potential wind farm site.

3) A performance measure, termed the *percentage of wasted iterations* (PWI), is developed to evaluate the computational effort with different rewarding iterations.

The rest of the paper is organized as follows. The problem model is given in Section 2. The proposed GARM algorithm is explained in Section 3. Results and discussion are presented in Section 4. The paper ends with a conclusion and future work in Section 5.

2. WAKE AND OPTIMIZATION MODELS

Traditionally, the WFLO problem is modeled as a dis–crete optimization problem, and the current study also assumes a discrete model. The complexity of the problem is 2^{c^2}, where C is the number of cells in the x-axis (or y-axis). For example, if the wind farm's physical area (which is squared shape) is divided into a 10×10 grid, then the total number of possible configu–rations for the layout will be 2^{10^2} (or 2^{100}). Figure 1 shows an example layout of a 10×10 grid. The grid is divided into 100 equal-sized cells, each representing a potential location for turbine placement. If the grid size increases, the possible layouts will also increase. For example, for a grid size of 15×15, the possible number of layouts will be 2^{225}. However, most studies on the WFLO problem assumed a 10×10 configuration and the same is used in the current study. The configuration is shown in Figure 1, with the turbine locations at the front (facing the wind) marked as 1 to 10, while the back of the farm has locations from 91 to 100.

Studies have proposed various wake-effect models. The present study adopted the wake effect model used by Ju and Liu [24], which was based on Jansen's wake model. Assume that N turbines are to be placed in the wind farm. The prevailing wind direction is assumed uni-directional (as shown in Figure 1) and has a speed of v_0. Furthermore, the prevailing wind is at an angle θ to the front of the farm. In the current study, $\theta = 0^o$ prevailing wind (Figure 1). As such, turbines directly facing the wind (in the first row) are under no-wake effect. Therefore, wind speed remains unaffected at these turbines. Turbines that are affected by the wake encounter a wind speed of v_i (i =1, 2, ... N) and $v_i < v_0$. The value of v_i depends on whether a turbine is affected by a single wake or multiple wakes. If a turbine is affected by the wake of a single turbine, the mathematical representation to calculate this wake effect is given by the following equation:

$$v_{i,j} = v_0 \left[1 - \frac{2}{3} \left(\frac{R_j}{r_j} \right)^2 \right] \tag{1}$$

where $v_{i,j}$ represents wind speed at wind turbine i under the wake effect of turbine j, and $R_j$denotes the rotor radius of wind turbine j. Furthermore, $r_j$is the wake radius and is represented as follows:

$$r_j = \alpha d_{i,j} + R_j \tag{2}$$

Table 1. Summary of previous studies.

Reference	Year	Algorithm(s)	Stopping condition
Mosetti et al. [9]	1994	Genetic Algorithms	Fixed number of iterations
Grady et al. [10]	2005	Genetic Algorithms	Fixed number of iterations
Huang [12]	2009	Genetic Algorithms	Fixed number of iterations
Emami and Noghreh [13]	2010	Genetic Algorithms	Fixed number of iterations
Gonzalez et. al. [14]	2010	Genetic Algorithms	Not specified
Kusiak and Song[15]	2010	Genetic Algorithms	Fixed number of iterations
Rašuo et al. [16][17]	2010	Differential Evolution	Fixed number of iterations
Eroğlu and Seçkiner [18]	2013	Ant colony optimization	Fixed number of iterations
Wang et al. [19]	2015	Genetic Algorithms	Fixed number of iterations
Rehman et al. [20]	2016	Cuckoo Search	Fixed number of iterations
Afanasyeva et al. [21]	2018	Cuckoo Search, Genetic Algorithms	Fixed number of iterations, minimum rate of change of cost value
Chahrouni et al. [22]	2019	Genetic algorithms	Fixed number of iterations
Wang [23]	2019	Genetic algorithms	Fixed number of iterations
Ju and Liu [24]	2019	Genetic Algorithms	Fixed number of iterations
Gao et al. [25]	2020	Genetic algorithms	Not specified
Wu et al. [26]	2020	Particle Swarm Optimization, Augmented Particle Swarm Optimization	Fixed number of iterations
Rehman et al. [27]	2020	Particle Swarm Optimization	Fixed number of iterations
Shin et al. [28]	2021	Evolutionary Algorithm, Particle Swarm Optimization	Not specified
Aggarwal et al. [29]	2021	Biogeography-based optimization, Genetic algorithms, Particle Swarm Optimization, Ant Colony optimization	Fixed number of iterations or the performance criterion is satisfied (criteria not mentioned)
Al Shereiqi [30]	2021	Genetic algorithms	Fixed number of iterations or fitness value below the threshold for a number of consecutive steps
Kirchner-Bossi [31]	2021	Genetic algorithms, Hybrid GA	Not specified
Afour et al. [32]	2022	Genetic Algorithm	Fixed number of iterations
Guoqing et al. [33]	2022	Genetic Algorithm	Not specified
Khan [5]	2022	Simulated Evolution	Fixed number of iterations
Huang et al. [34]	2023	Evolutionary Algorithm	Max number of fitness evaluation

In (2), α denotes the entrainment factor while $d_{i,}$represents the downstream distance.

In the case where turbine i is affected by multiple wakes, the following equation is employed to calculate the wake as follows:

$$v_i = v_0\left[1-\sqrt{\sum\nolimits_{j\in\Phi_j}\left(1-\frac{v_{i,j}}{v_0}\right)^2}\right] \tag{3}$$

The objective function in this study is adopted from Ju and Liu [24]. The function requires maximization of the efficiency of the wind power generation and is given by the following equation.

Maximize

$$Efficiency = P_{current} / P_{total} \tag{4}$$

where $P_{current}$ denotes the total power generated by turbines under the wake effect in the current layout, and P_{total} is the ideal total power generated by all turbines without the impact of any wake. A detailed discussion of the optimization model can be found in the study by Ju and Liu [24].

3. PROPOSED GENETIC ALGORITHM WITH RE–WARDING MECHANISM

The genetic algorithm [35] is the first and most estab–lished NIA designed to solve NP-hard optimization problems. The algorithm uses a set of solutions called *population*, which evolves into better solutions through an iterative process. This evolution is characterized by two operations, known as *crossover*and *mutation*. The purpose of crossover is to pass on the characteristics present in the current population to the next generation, resulting in new offspring. Furthermore, mutation intro–duces new characteristics in the offspring generated during the crossover phase. Cumulatively, crossover and mutation carry out the exploitation and exploration, respectively, through an iterative process.

As mentioned earlier, the iterative process in the conventional GA is carried out using the iteration count, which is a user-defined value. However, this requires the algorithm to be user-dependent, thus instigating a forced termination. Furthermore, if the number of iterations is defined as less than what is required to converge, then the optimal solution is not guaranteed. In contrast, if the number of iterations is more than what is required, then the execution results in a waste of computational resources.

In order to control the number of wasted iterations, the rewarding mechanism, which is a characteristic and a unique feature of the stochastic evolution algorithm, has been incorporated in GARM. Stochastic evolution

(StocE) [36] is a non-deterministic iterative algorithm inspired by the behavior of biological processes. In contrast to many other iterative algorithms that operate on a population of solutions, there are only a few evo–lutionary algorithms, such as StocE [36] and Simulated Evolution [37-39], that maintain a single solution thro–ughout their execution.

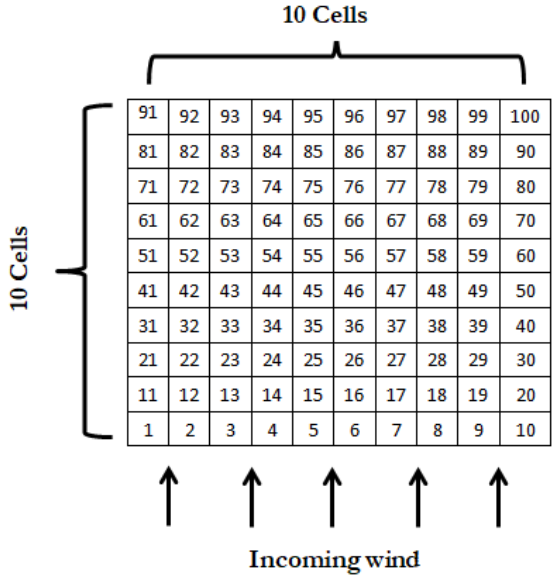


Figure 1. The wind farm layout is divided into a 10x10 grid

The single solution inStocE is perturbed iteratively to improve the quality of the solution, thus leading to an op–timal or quasi-optimal solution. The classical StocE algo–rithm has two distinctive features: the *compound move* and the *rewarding mechanism* [36,40]. In the compound move, multiple perturbations are carried out to the single solution per iteration. This is equivalent to carrying out multiple mutations fora single chromosome in GA. The size of the compound move (i.e., the number of pertur–bations) maintains a balance between exploration and exploitation. A very big compound move would lead the algorithm towards randomization (which is not desired) as well as an increase in the runtime. In contrast, a small compound move may result in the StocE algorithm getting trapped in local optima, which is also undesired. Therefore, the size of the compound move should be set carefully.

The rewarding mechanism, which is the focus of the current study, allows the StocE algorithm to extend its execution time, thus enabling the algorithm to tra–verse the search space more extensively, hoping to result in better solutions. The rewarding mechanism allows the algorithm to be rewarded with extra iterations whenever an improvement is observed in the quality of the solution. In the context of the underlying study, this improvement is measured through the objective function presented in (4). The rewarding mechanism is incorporated in the GARM, where additional iteration sare added to the genetic algorithm whenever an impro–vement is observed in the fitness value (i.e., the objective function).

Figure 2 shows a pseudo-code ofGARM. The algo–rithm starts with initialization (line 1), generating a set of chromosomes (wind farm layouts). The user defines the size of this set, i.e., the population size. Each generated layout is evaluated for quality using (4) (see line 2). Furthermore, other variables and parameters are also defined, as shown in lines 3 to 6. Following this, the number of iterations, ρ, is initialized to 1, as shown in line 7. That is, the algorithm stops after a single iteration. The rationale behind this is that if GARM observes an improvement after the first iteration, then reward iterations will be awarded to the algorithm, leading to further execution.

The main iterative process of GARM is carried out between lines 8 and 17. The 'while' loop at line 8 would carry out the selection, crossover, and mutation operati–ons until no more iterations are left for execution. The core of the rewarding mechanism is reflected in lines 13 to 16. In line 13, the best solution in the current iteration, $F_{i\ best}$, is compared with the overall best solution, $F_{o\ best}$, found so far. If $F_{i\ best} > F_{o\ best}$, then the best solution in the current iteration is designated as the overall best solution (line 14), and the reward iterations are added to the iteration count (line 15), thus increasing the number of iterations leading to further execution. In case $F_{i\ best} \leq F_{o\ best}$, the iterations count is decreased by an iteration (line 16), and the population is updated to carry out the next iteration. Once no more iterations are left, the algorithm terminates.

Algorithm GARM

```
1.  Generate initial population with layouts
2.  Evaluate the fitness of the initial population
3.  Define ρ as the number of iterations
4.  Define F_o_best as the fitness of the overall best solution
5.  Define F_i_best as the fitness of the best solution in the current iteration
6.  Define R as the number of iterations to be rewarded.
7.  ρ= 1
8.  while (ρ> 0) {
9.  Select Parents
10. Perform Crossover
11. Perform Mutation
12. Evaluate the fitness of each solution in current iteration
13. If F_i_best >F_o_best
14. S_i_best >S_o_best     // Iteration best solution becomes overall best solution
15. ρ = ρ+ R               // Add reward iterations to iteration count
16. Else  ρ = ρ- 1         // Reduce iteration count
17. Replace the current population for the next generation
18. }
```

Figure 2. Pseudo-code of GARM

As stated earlier, the purpose of the present study is to evaluate the impact of different numbers of rewarding iterations in order to maximize layout efficiency while reducing the number of extra (rather wasted) iterations. With this objective, results are analyzed in several dimensions. The first analysis is focused on the impact of different rewarding iteration values on efficiency. This is followed by analyzing the number of extra itera–tions generated for each of the five rewarding iteration values. Details of these analyses are given below.

4. RESULTS AND DISCUSSION

Simulations were carried out using real data from a potential site of Turaif, located in the northern region of Saudi Arabia. The site has an elevation of 827 meters above the mean sea level with an average wind speed of 6.94 m/s [41]. Thus, a single-speed, single-direction scenario is assumed. Furthermore, the following para–meter setup was used for the genetic algorithm: popu–lation size = 30, crossover rate = 0.6, and mutation rate = 0.1. Other parameters used in the simulations included a grid size of 10 x 10 and a number of turbines = 20. Empirical results were obtained while assuming a re–ward of 5, 10, 15, 20, and 25 iterations. Furthermore, a GE 1.5sle turbine was used in the simulations. Table 2 summarizes the turbine specifications and other rele–vant information.

Table 2. Specifications of the turbine used in the study

Turbine	GE 1.5sle turbine
Hub Height	80 m
Rotor Diameter	77 m
Cut-in Speed	3.5 m/s
Cut-out Speed	25 m/s
Rated Speed	12 m/s
Rated Power	1.5 MW

To analyze the results of iterative heuristics, 30 independent runs are carried out for each value of rewarding iterations. The best output value (in terms of efficiency as given by (4)) for each run is taken, and the average best value of the 30 runs is reported, along with the standard deviation of the 30 runs. All runs are started with the same seed solutions.

4.1 Effect of Reward Iterations on Efficiency

The effect of five reward iterations is analyzed con–cerning the corresponding average efficiency (i.e., an average of 30 runs). Table 3 provides the average, maxi–mum, and minimum efficiency with respect to the different rewards, along with the standard deviation. From this table, it is observed that reward R = 5 gives the worst average efficiency while R values of 15, 20, and 25 have almost the same level of average efficiency. However, among these three rewarding mechanisms, R = 20 and R =25 have almost the same standard deviation (0.0075 and 0.0076), as shown in the last column of Table 3. As such, both of these rewarding values are equally good. The plots in Figure 3 further elaborate on the trends, where it is observed that as the number of rewarding iterations increased, the average, maximum, and minimum efficiencies also increased. This rate of increase is faster for small rewards (from R = 5 to R = 15). However, for larger reward values of R = 20 and R = 25, not much improvement is observed in the effi–ciency. However, in order to differentiate further bet-ween the impact of R = 20 and R = 25, further analysis is required, which is presented in the next section.

Table 3. Effect of different rewarding iterations on the average, maximum, and minimum efficiency

Reward	Average efficiency	Max efficiency	Min efficiency	Std. dev.
5	0.7533	0.7712	0.7326	0.0092
10	0.7677	0.7985	0.7425	0.0123
15	0.7830	0.8053	0.7618	0.0097
20	0.7876	0.8121	0.7745	0.0075
25	0.7874	0.8028	0.7744	0.0076

4.2 Effect of Reward Iterations on Computational Effort

Another dimension to evaluate the impact of the dif–ferent reward iterations is in terms of computational effort. This effort is measured in terms of the percentage of wasted iterations, as follows:

$$PWI = \frac{TI - IBS}{TI} \times 100 \tag{5}$$

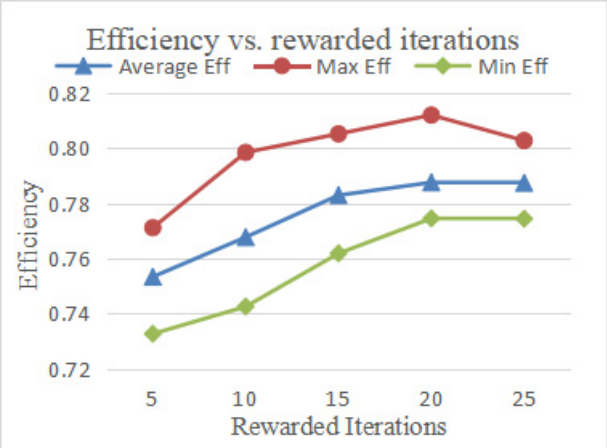


Figure 3. Efficiency versus the rewarded iterations.

In the above equation, the term TI defines the total number of iterations for which a run is executed. The term IBS identifies the iteration of the best solution during the run. This best solution is measured as efficiency given by (4). The difference in TI and IBS shows the number of extra iterations executed after the best solution is found. This difference highlights the number of *wasted* iterations. Finally,PWI denotes the percentage of wasted iterations.

The rationale behind measuring PWI is that further iterations are unnecessary once the best solution is achieved, as they add redundancy to the computational effort. As such, the additional iterations beyond the ite–ration at which the best solution is obtained should be reduced as much as possible (if not completely alle–viated). Therefore, a low value of PWI is desirable since it indicates that the simulation run should be terminated as soon as the best solution is found. Conversely, a high value of PWI shows unnecessary computational effort.

Table 4 provides the summary of results for the five reward values investigated. The table shows the maxi–mum, minimum, and average number of various metrics over 30 runs. These include the total iterations (TI) exe–cuted, the iteration in which the best solution (in terms of efficiency) was found (i.e., IBS), the difference between TI and IBS, and the corresponding PWI values. It is observed from the tables that for small values of *R*, the value of PWI is high, whereas for higher values of *R*, small PWI values are observed. More specifically, the average values of PWI for R = 5, 10, and 15 are 49.55, 46.67, and 43.82, respectively. As the values of *R* are increased to 20 and 25, the corresponding PWI values are found to be 39.19 and 38.41, respectively. This trend indicates that increasing the reward results in more efficient use of the computational resources as desired. That is, when R = 5, almost half of the iterations (49.55 %) were wasted, whereas, with$R$ = 25, the wasted iterations were considerably reduced to 38.41 %, which is a reduction of almost 11% compared to that of R = 5.

Another observation from Table 4is that for smaller values of *R*, the reduction in PWI is at a higher rate, but for small to large values of *R*, the reduction is at a much higher rate. However, for higher values of *R*, the reduction in PWI is slow. That is, between R = 5 to R = 10, the reduction in average PWI is 2.88% (i.e., 49.55 to 46.67), while from R = 10 to R = 15, the reduction is 2.85%. Furthermore, from R = 15 to R = 20, the dif–ference in average PWI is 4.63%. However, between R = 20 and R = 25, the change in average PWI is only 0.78 %. Overall, the results suggest that a change in rewarding iterations for smaller *R* values can notably impact the computational effort. However, when the rewarding iterations are increased beyond a certain point (in the case of R = 20 to R = 25), the impact on computational effort is almost negligible.

Figure 4 illustrates the stability of the five rewarding iterations for 30 runs in terms of the variation of PWI values (note that the average PWI values for different rewarding iterations are shown in Table 4). As can be seen from Figures 4(a), (b), and (c), the PWI values have a significant variation for R = 5, 10, and 15, respectively, showing a somewhat unstable behavior in terms of algorithm convergence. However, for higher rewarding iterations (R = 20 and R = 25), the graphs in Figures 4(d) and (e) show a stable behavior. The above trends indicate that GARM has relatively less variation in PWI for higher rewarding iterations compared to smaller values. In other words, for higher values of *R*, the algorithm has more or less the same behavior in all runs when it comes to finding the difference between the total number of iterations and the iteration at which the best solution was found.

5. DISCUSSION

The results in Sections (4.1) and (4.2) highlighted several important trends. On one hand, when it comes to evaluating the performance in terms of efficiency, hig–her values of *R* are desirable. On the other hand, higher values of *R*also resulted in more effective use of com–putational effort. From the engineering point of view, the WFLO problem requires the efficiency to be inc–reased (which in turn favors more power generation).

From the computational point of view, the objective is to reduce the computational effort. This computa–tional effort is reduced when the difference between the total runtime (in terms of the number of iterations) and the time at which the best solution was obtained is mi–nimized. It should also be noted that for small values of *R*, the iteration at which the best solution was obtained with respect to the total number of iterations is computa-tionally effective. However, this computational effecti–veness is negated due to low-efficiency solutions, as ob–served in Table 2. In order to get a solution that is effe–ctive both in computational as well as engineering di–mensions, a solution is desired that gives the best tradeoff of both dimensions. From the results, both R = 20 and R = 25 satisfy this requirement over other values of *R*.

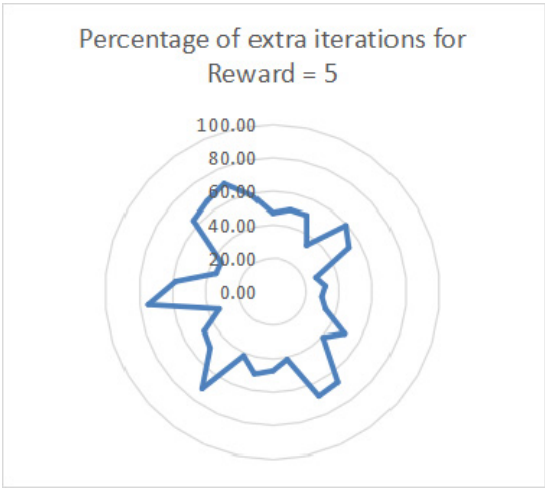
Table 4. Effect of rewarding iterations on different metrics (on averages of 30 runs)

No. of reward iterations	Metric	Maximum	Minimum	Average
5	Total Iterations (TI)	32	7	18 ± 6.07
	Best solution in iteration (IBS)	21	2	9.43 ± 4.85
	Difference between best and total iterations	16	4	8.57 ± 2.99
	PWI	75	27.27	49.55 ± 13.86
10	Total Iterations (TI)	79	22	50.9 ± 14.97
	Best solution in iteration (IBS)	56	6	28 ± 13.68
	Difference between best and total iterations	45	10	22.9 ± 9.08
	PWI	81.25	22.22	46.67 ± 17.01
15	Total Iterations (TI)	113	62	90.7 ± 15.38
	Best solution in iteration (IBS)	74	17	50.9 ± 11.28
	Difference between best and total iterations	61	19	39.77 ± 10.6
	PWI	72.58	24.68	43.82 ± 9.73
20	Total Iterations (TI)	136	114	125.3 ± 6.35
	Best solution in iteration (IBS)	87	65	76.3 ± 6.35
	Difference between best and total iterations	49	49	49 ± 0
	PWI	42.98	36.03	39.19 ± 1.96

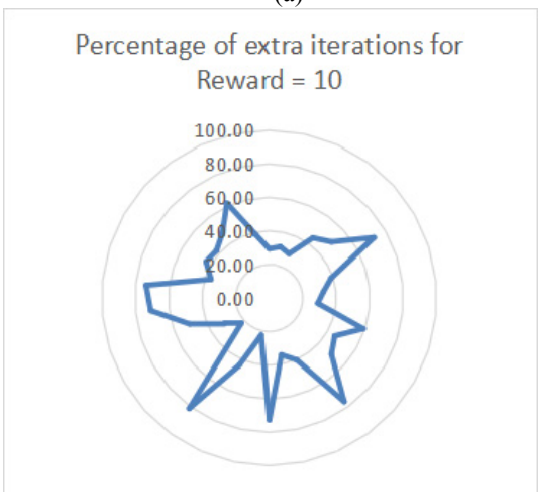
25	Total Iterations (TI)	148	114	128.1 ± 9.22
	Best solution in iteration (IBS)	99	65	79.13 ± 9.21
	Difference between best and total iterations	49	48	48.97 ± 0.18
	PWI	42.98	33.11	38.41 ± 2.71

6. CONCLUSION

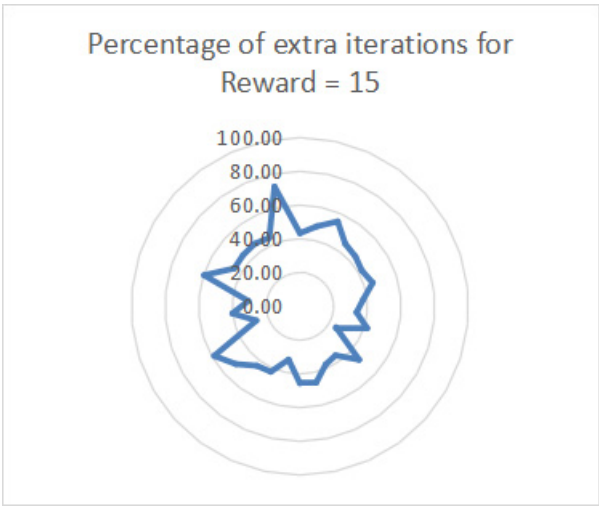
The non-polynomial complexity of the wind farm layout design (WLFO) problem solicits an efficient approach in terms of quality of solution and execution time. Pre–vious studies have shown that genetic algorithms effec–tively solve both aspects. Yet, like any other iterative nature-inspired algorithm, the genetic algorithm still has room for improvement. One area of improvement is to reduce the computational effort of the algorithm. This can be done by reducing the number of iterations once the optimal solution is obtained. In this paper, we pro–posed a modified genetic algorithm that is inspired by the rewarding mechanism of the stochastic evolution al–gorithm. The modified GA with the rewarding mecha–nism is tested using real data with respect to the solution quality and computational effort. Results indicated that for the five values of rewarding iterations tested, the higher number of rewarding iterations is more favorable than the lower number of rewarding iterations. Furthermore, a high number of rewarding iterations demonstrated a more stable behavior in terms of algorithm convergence compared to the low number of rewarding iterations.



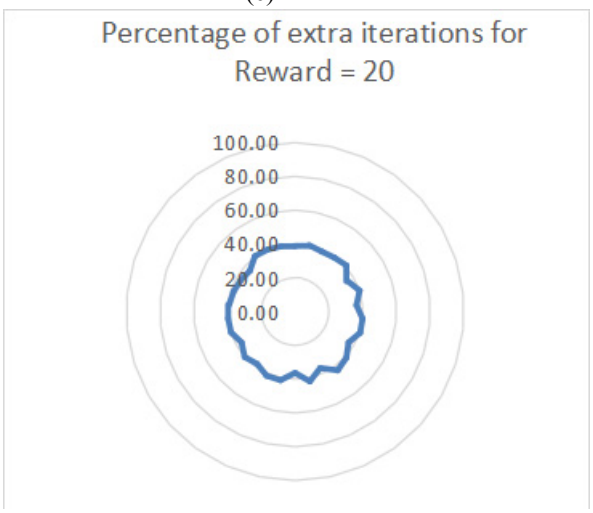
(a)



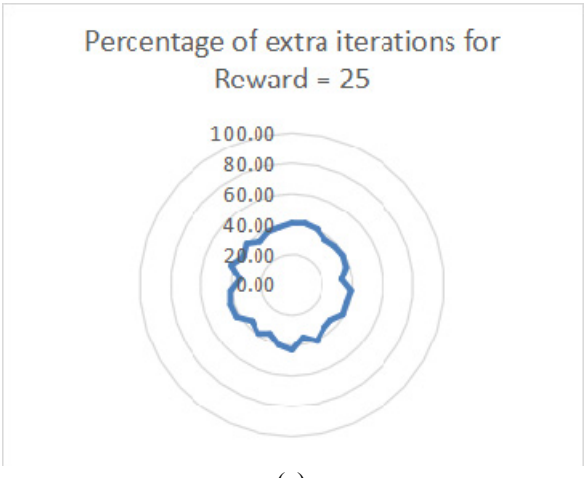
(b)



(c)



(d)



(e)

Figure 4. Percentage of extract iterations for different reward iterations (a) Reward = 5 (b) Reward = 10 (c) Reward = 15 (d) Reward = 20 and (e) Reward = 25.

The present study has the potential to expand in several dimensions, as itemized below.

- The comprehensive impact of the rewarding mechanism can be investigated with several other similar algorithms, such as swarm intel–ligence and other evolutionary computation algorithms.
- More real data from several potential sites can be used to lead to more concrete conclusions.
- Several rewarding mechanisms can be deve–loped to improve the algorithm's performance further. Particularly, a dynamic, rewarding mechanism can be developed that adjusts the rewarding iterations according to the algorithm's performance
- A variety of numerical experiments with more sets of rewarding iterations can be tried.

ACKNOWLEDGEMENT

The authors acknowledge King Fahd University of Pet–roleum & Minerals for supporting this study through Interdisciplinary Research Center grant # INRE 2217.

REFERENCES

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

[2] Khan, S. A.: Adaptation of the simulated evolution algorithm for wind farm layout optimization. FME Transact, 50(4), 664-673.

[3] Rehman, S., Salman, U. T., Mohandes, M. A., Al-Sulaiman, F. A., Adetona, S., Alhems, L. M., and Baseer, M. A.: Wind speed prediction based on longshort term memory using nonlinear autoreg–ressive neural networks. FME Transactions, Vol. 50, No. 2, pp. 260-270, 2022.

[4] Dinulović, M. R., Trninić, M. R., Rašuo, B. P., and Kožović, D. V.: Methodology for aeroacoustic noise analysis of 3-bladed h-Darrieus wind tur–bine. Thermal Science, Vol. 27 (1 Part A), pp. 61-69, 2023.

[5] Rašuo, B. P., Veg, A. D.: Design, fabrication, and verification testing of the wind turbine rotor blades from composite materials, in: *Proceedings of the ICCM-16*, Japan, 2007, pp. 1-4.

[6] Parezanovic, V., Rašuo, B., Adzic, M.: Design of airfoils for wind turbine blades. in: *Proceedings of the French-Serbian European Summer University: Renewable Energy Sources And Environment-Multidisciplinary Aspect,* 2006, pp. 195-200.

[7] Rašuo, B., Dinulović, M., Veg, A., Grbović, A., and Bengin, A.: Harmonization of new wind tur–bine rotor blades development process: A re–view. Renew. Sust. Energ. Rev, Vol. 39, pp. 874-882, 2014.

[8] Khan, S. A., Rehman, S.: Iterative non-determi–nisticalgorithms in on-shore wind farmdesign: A brief survey, Renew. Sust. Energ. Rev,Vol. 19, No. 3, pp. 370–384, 2013.

[9] Grady, S.A., Hussaini, M,Y., Abdullah, M. M.: Placement of wind turbines using genetic algo–rithms. Renew. Energ., Vol 30, pp. 259–270, 2005.

[10] Mosetti, G., Poloni, C., Diviacco, B.: Optimi–zationof wind turbine positioning in large wind farms bymeans of a genetic algorithm. J. Wind Eng. Indust. Aerodyn. Vol. 51. pp.105–116. 1994.

[11] Ghoreishi, S. N., Clausen, A., Jørgensen, B. N.: Termination Criteria in Evolutionary Algorithms: A Survey. in: *Proceedings of the 9th International Joint Conference on Computational Intelligence*, Portugal, 2017, pp. 373-384.

[12] Huang, H.S.: Efficient hybrid distributed genetic algorithms for wind turbine positioning in large wind farms. in: *Proceedings ofIEEE International Symposium on Industrial Electronics*,2009, pp. 2196–2201.

[13] Emami, A., Noghreh, P.: New approach on opti–mization in placement of wind turbines withinwind farm by genetic algorithms. J. Renew.Energ.Vol. 25, pp. 1559–64, 2010.

[14] Gonzalez, J. S., Rodriguez, A. G. G., Mora, J. C., Santos, J. R., Payan, M. B.: Optimization of wind farm turbines layout using an evolutive algorithm. Renew. Energ., Vol. 35, No. 8, pp. 1671–1681, 2010.

[15] Kusiak, A., Song, Z.: Design of wind farm layout for maximum wind energy capture.Renew. Energ., Vol. 35, No. 3, pp. 685–694, 2010.

[16] Rašuo, B., Bengin, A., Veg, A.: On aerodynamic optimization of wind farm layout. PAMM, Vol. 10, No. 1, pp. 539–540, 2010.

[17] Rašuo, B., Bengin, A.: Optimization of wind farm layout. FME Trans., Vol. 38, pp. 107–114, 2010.

[18] Eroğlu, Y., Seçkiner, S.: Wind farm layout opti–mization using particle filtering approach. Renew. Energ., Vol. 58, pp. 95–107, 2013.

[19] Wang, L., Tan, A. C., Gu,Y., and Yuan, J.: A new constraint handling method for wind farm layout optimization with lands owned by different owners.Renew. Energ., Vol. 83, pp. 151–161, 2015.

[20] Rehman, S., Ali, S.S., Khan, S.A.: Wind farm layout design using cuckoo search algorithms.Appl. Artif. Intell., Vol. 30, No. 10, pp.899-922, 2016

[21] Afanasyeva, S., Saari, J., Pyrhönen,O., Partanen, J.: Cuckoo search for wind farm optimization with auxiliary infrastructure. Wind Energy, Vol. 21, No. 10, pp. 855–875, 2018.

[22] Charhouni, N., Sallaou, M., Mansouri, K.: Realistic wind farm design layout optimization with different wind turbines types. Int. J.Energ.Env. Eng., Vol. 10, No. 3, pp. 307–318, 2019.

[23] Wang, L.: Comparative study of wind turbine pla–cement methods for flat wind farm layout opti–mization with irregular boundary. Appl. Sci., Vol. 9, No. 4, p.639, 2019.

[24] Jug, X., and Liu, F.: Wind farm layout optimization using self-informed genetic algorithm with

information guided exploitation. Apple. Energy. Vol. 248, pp. 429–445, 2019.

[25] Ago, X., Li, Y., Zhao, F., Sun, H.: Comparisons of the accuracy of different wake models in wind farm layout optimization. Energy Explor. Exploit., Vol. 38, No. 5, pp. 1725–1741, 2020.

[26] Wu, X., Hu, W., Huang, Q.,Chen, C., Jacobson,M. Z., Chen, Z.: Optimizing the layout of onshore wind farms to minimize noise. Appl. Energ., Vol. 267, p. 114896, 2020.

[27] Rehman, S., Khan, S. A., Alhems, L. M.: The effect of acceleration coefficients in particle swarm optimization algorithm with application to wind farm layout design. FME Trans., Vol. 48, No. 4, pp. 922-930. 2020.

[28] Shin, J., Baek, S., Rhee, Y.: Wind farm layout opti–mization using a meta model and ea/pso algorithm in Korea offshore. Energies, Vol. 14, No. 1, pp. 146, 2020.

[29] Aggarwal, S. K., Saini, L. M., Sood,V.:Large wind farm layout optimization using nature inspired meta-heuristic algorithms. IETE J.Res., pp. 1–18, 2021.

[30] Al Shereiqi, A., Mohandes, B., Al-Hinai, A., Bakhtvar, M., Al-Abri, R., El Moursi, M., Albadi, M.: Co-optimisation of wind farm micrositing and cabling layouts. IET Renew. Power Gener., Vol. 15, No. 8, pp. 1848–1860, 2021.

[31] Kirchner-Bossi, N., Port-Agel, F.: Wind farm area shape optimization using newly developed multi-objective evolutionary algorithms. Energies, Vol. 14, No. 14, p. 4185, 2021.

[32] Asfour, R., Brahimi, T., El-Amin, M.: Wind farm layout: modeling and optimization using genetic algorithm. in: *Proceedings of the IOP conference Series: Earth and Environmental Science,* Vol. 1008, 2022, p. 012004. IOP Publishing.

[33] Guoqing, H., Zhang, S., Li, K.: Optimization of wind farm regular layout based on grid coordinate genetic algorithm. in: *Proceedings of the World Congress on Advances in Civil, Environmental, & Materials Research (ACEM22)*, Seoul, Korea, 2022.

[34] Huang, X., Wang, Z., Li, C., and Zhang, M.: A low-complexity evolutionary algorithm for wind farm layout optimization. Energy Rep., Vol. 9, pp. 5752-5761, 2023

[35] Holland, J. H.: Genetic algorithms and adapta–tion. Adaptive control of ill-defined systems, Vol. 16, pp. 317-333, 1984.

[36] Saab, Y.,and Rao, V.: Stochastic Evolution: A fast effective heuristic for some generic layout problems. In: *Proceedings of the 27th ACM/IEEE Design Automation Conference*, 1991, pp. 26-31.

[37] Youssef, H., Sait, S. M., and Khan, S. A.: Fuzzy evolutionary hybrid metaheuristic for network topology design. In: *Proceedings of Evolutionary Multi-Criterion Optimization: First International Conference*, Zurich, Switzerland, 2001, pp. 400-415.

[38] Kling, R., Banerjee, P.: ESP: Placement by simulated evolution. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., Vol. 8, No. 3, pp. 245-256, 1989.

[39] Khan, S. A., Engelbrecht, A.P.:Assessment of the "Evaluation" function in the simulatedevolution algorithm. in: *Proceedings of IEEE 7th International Conference on Natural Computation*, China, 2011, pp. 1062-1066.

[40] Mahmood, A., Khan, S. A., Albalooshi, F., Awwad, N.: Energy-aware real-time task scheduling in multiprocessor systems using a hybrid genetic algorithm. Electronics, Vol. 6, No. 2, p. 40, 2017.

[41] Rehman, S., Khan, S. A., and Alhems, L. M.: Application of TOPSIS approach to multi-criteria selection of wind turbines for on-shore sites. Appl. Sci., Vol. 10, No. 21, pp. 7595, 2020.

NOMENCLATURE

$v_{t,j}$	Wind speed at turbine i under the wake of turbine j
v_0	Mean wind speed (prevailing wind)
v_i	Wind speed at turbine i
RR_j	Rotor radius of turbine j
r_j	Wake radius
α	Entrainment factor
$d_{i,j}$	Distance downstream from turbine j to turbine i (i.e., distance between the current turbine and the turbine creating wake effect on it)
θ	The angle of prevailing wind to the front of the farm
N	Total number of turbines
C	Number of cells in the layout grid
$P_{current}$	Total power generated by turbines
P_{ideal}	Ideal power generated by turbines
ρ	Number of iterations in GARM
F_{o_best}	Fitness of overall best solution
F_{i_best}	Fitness of best solution (current iteration)
S_{i_best}	Iteration best solution
S_{o_best}	Overall best solution
GA	Genetic algorithms
GARM	Genetic algorithm with the rewarding mechanism
StocE	Stochastic Evolution algorithm
PWI	Biogeography based optimization
TI	Percentage of wasted iterations
IBS	Total iterations
	Iteration at which the best solution is found

ГАРМ: ГЕНЕТСКИ АЛГОРИТАМ ЗАСНОВАН НА СТОХАСТИЧКОЈ ЕВОЛУЦИЈИ СА МЕХАНИЗМОМ НАГРАЂИВАЊА ЗА ОПТИМИЗАЦИЈУ РАСПОРЕДА ВЕТРОПАРКА

М. Мохандес, С.А. Кан, Ш. Рехман, А. Ал-Шаики Б. Лиу, К. Икбал

Енергија ветра се појавила као потенцијална алтернатива традиционалним изворима енергије за

економичну и чисту производњу електричне енер–гије. Један важан аспект производње енергије ветра је дизајн ветропарка како би се искористила мак–симална енергија. Због своје инхерентне рачунске сложености, проблем дизајна ветропарка традици–онално се решава коришћењем алгоритама инспи–рисаних природом. Важно питање у алгоритмима инспирисаним природом је услов завршетка, који регулише време извршења алгоритма. Да би се оптимизирало време извршења, требало би при–менити одговарајуће услове раскида. Ова студија предлаже концепт механизма награђивања за пос–тизање оптимизације у условима завршетка уз одр–жавање квалитета решења. Предложени механизам награђивања, усвојен из алгоритма стохастичке ево–луције, уграђен је у генетски алгоритам. Предло–жени генетски алгоритам са механизмом награ–ђивања (ГАРМ) је емпиријски тестиран коришће–њем стварних података са потенцијалне локације ветропарка са различитим итерацијама награђивања.