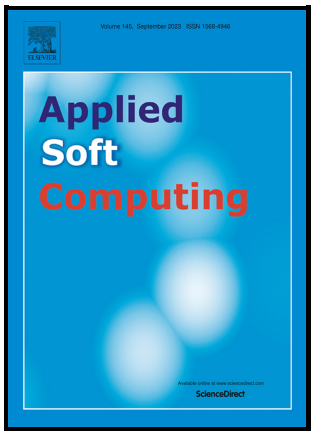
Differential Evolutionary Particle Swarm Optimization with Orthogonal Learning for Wind Integrated Optimal Power Flow

Wenlei Bai, Fanlin Meng, Ming Sun, Haoxiang Qin, Richard Allmendinger, Kwang Y Lee



PII: S1568-4946(24)00436-8

DOI: https://doi.org/10.1016/j.asoc.2024.111662

Reference: ASOC111662

To appear in: *Applied Soft Computing*

Received date: 26 May 2023
Revised date: 2 March 2024
Accepted date: 19 April 2024

Please cite this article as: Wenlei Bai, Fanlin Meng, Ming Sun, Haoxiang Qin, Richard Allmendinger and Kwang Y Lee, Differential Evolutionary Particle Swarm Optimization with Orthogonal Learning for Wind Integrated Optimal Power Flow, *Applied Soft Computing,* (2024) doi:https://doi.org/10.1016/j.asoc.2024.111662

Differential Evolutionary Particle Swarm Optimization with Orthogonal Learning for Wind Integrated Optimal Power Flow

Wenlei Bai*, Fanlin Meng, *Senior Member, IEEE*, Ming Sun, Haoxiang Qin, Richard Allmendinger, *Senior Member, IEEE*, and Kwang Y. Lee, *Fellow, IEEE*

Wenlei Bai (corresponding author)
School of Engineering and Computer Science, Baylor University, Waco, TX, USA.
wenlei_bai@baylor.edu

Fanlin Meng
Alliance Manchester Business School, University of Manchester, Booth St W, Manchester M15 6PB, UK.
University of Exeter Business School, University of Exeter, Exeter, EX4 4PU, UK.
fanlin.meng@manchester.ac.uk

Ming Sun
College of Computer and Control Engineering, Qiqihar University, Qiqihar, Heilongjiang, China.
snogisunming@163.com

Haoxiang Qin
School of Software Engineering, South China University of Technology, Guangzhou, China.
987352978@qq.com

Richard Allmendinger
Alliance Manchester Business School, University of Manchester, Booth St W, Manchester M15 6PB, UK.
richard.allmendinger@manchester.ac.uk

Kwang Y Lee
School of Engineering and Computer Science, Baylor University, Waco, TX, USA.
Kwang_Y_Lee@baylor.edu

* Corresponding author.
E-mail address: wenlei_bai@baylor.edu (W. Bai), fanlin.meng@manchester.ac.uk (F. Meng), snogisunming@163.com (M. Sun), 987352978@qq.com (H. Qin), richard.allmendinger@manchester.ac.uk (R. Allmendinger), kwang_y_lee@baylor.edu (K. Y. Lee)

Keywords:
Particle swarm optimization (PSO), differential evolution (DE), orthogonal learning (OL), wind power, optimal power flow (OPF)

Abstract—This study develops a novel variant of particle swarm optimization (PSO), which improves its balance of exploration and exploitation by modifying neighborhood topology, self-adaptive parameter strategies and deep search, namely differential evolutionary evolution PSO with orthogonal learning (OL), i.e., DEEPSO-OL in short. Evolutionary computing can explore the solution space efficiently because of its self-evolving attribute as iteration continues. The OL enhances its exploitation by focusing on deeper search for promising solutions. It utilizes the concept of orthogonal experimental design (OED) which predicts the best combination of control variables without exhaustive evaluation of all possible combinations. In addition, to avoid premature convergence in a local optimum, a stochastic star topology for particles is proposed. Such topology ensures just enough communication among the best performing particles, while encouraging them to explore other spaces. The efficacy of the algorithm is evaluated through real-world scenarios such as optimal power flow (OPF) and wind integrated OPF, which are hard to solve with classical mathematical methods. The proposed algorithm is run on a modified IEEE 30-bus test system and compared to the state-of-the-art evolutionary computing algorithms for a variety of cost objective functions

with high levels of non-linearity and non-convexity. The DEEPSO-OL demonstrates its performance to generate more accurate feasible solutions and construct promising and efficient search method for real-world complex optimization problems.

***Index Terms*—Particle swarm optimization (PSO), differential evolution (DE), orthogonal learning (OL), wind power, optimal power flow (OPF).**

1. INTRODUCTION

Particle swarm optimization (PSO) has been widely applied in continuous complex optimization domain over the past 25 years, where classical mathematical programming is no longer practical since highly non-convex and non-linear properties [1]. Even though PSO is relatively efficient, simple, and easy-to-implement, it still has some obvious drawbacks, such as lack of robustness, weak balance in exploration and exploitation causing premature convergence, weak scalability, difficulties in tuning system parameters, etc. [2][4]. The well-known *No Free Lunch Theorem* has stated that no meta-heuristic algorithm can be superior to additional algorithms for all optimization problems universally [3]. Thus, many PSO variants are developed to tackle different problems. Design of those variants can be directed by modifying neighborhood topology, evolving system parameters, executing deep search, etc. Essentially, the objective is improving the balance between exploitation and exploration. Spanning from minor improvements to the integration of advanced innovative concepts, most variations have been developed manually, with developers experimenting with new designs based on their individual knowledge and expertise [4].

The existing PSO variants can be summarized into four aspects with each PSO variant falling into only one or two aspects [4]: 1) variants on system parameters, such as self-adaptive parameters; 2) variants on the particle's search mechanism to guide some deep search; 3) variants focused on adding perturbation to position or velocity vectors to escape from local minimal; 4) variants on population topology and size. It should be noted that our proposed differential evolutionary evolution PSO with orthogonal learning (DEEPSO-OL) algorithm, aims to enhance the standard PSO in all four aspects.

Aspect 1 focuses on designing control parameters for the system, such as time-varying or adaptive/self-adaptive inertia and acceleration coefficients. Time-varying parameters are defined as functions dependent on specific iterations during algorithm runs whereas adaptive and self-adaptive parameters are functions dependent on the information during the running process to adjust their values. Because those control parameters influence the exploitation and exploration abilities of the algorithm heavily, there are numerous variations of parameter control strategies in the literature on PSO [15][16]. The weights in DEEPSO-OL undergo a mutation process where they evolve on a log distribution at each iteration. Such process improves the search behavior by balancing the exploration and exploitation because log distributions concentrate more values on the lower end and fewer on the higher end. In other words, more particles equipped with lower weights lead to a higher exploration rate while fewer particles with higher weights focus on exploiting the best solutions.

Aspect 2 focuses on improving search mechanisms, namely, regulating the distribution of all potential positions of particles. Hybrid operations with genetic algorithm [17] and differential evolution [18] fall into this category. The proposed DEEPSO-OL integrates the orthogonal learning (OL) operator to search deeper near good solutions. The OL is a technique that leverages orthogonal arrays to sample and exploit the search space more efficiently instead of exhaustively evaluating all possible feasible solutions.

Aspect 3 focuses on different mechanisms to apply perturbations to position and velocity. It improves the diversity of solutions and avoids stagnation [19]. There are *informed* or *random* perturbations. The *informed* mechanisms typically use the information of specific solution populations as the parameters (e.g., mean value) for a probability distribution, and then map random values are around them, while *random* perturbation simply introduces a stochastic value to perturbate a particle's position or the velocity. The DEEPSO-OL uses the informed perturbation on its global best solution with a normal distribution. Such mechanism equips PSO with better chance of escaping from local minimal due to the randomness introduced in the search process.

Aspect 4 includes the topology and the size of the population. Topology is crucial in achieving a balance between exploration and exploitation in search. The fully connected star, ring, and von Neumann topologies are widely recognized, and other topologies, such as hierarchical and small-world networks, have also been studied in PSO literature. Montes de Oca et al. [20] introduced a topology that reduces connectivity over time. Regarding population size, the number of particles changes dynamically in the iteration process according to some metrics [21]. In general, the population size will impact the tradeoff between solution quality and convergence speed. It is worth noting that increasing population size does not always guarantee better solution quality. In some complex cases where there are many local minima, the smaller population size outperforms the larger population size because of the prompt response of moving out of a local minimum. This paper introduces a stochastic star topology on particles, aiming to establish just about the right amount of information sharing and to avoid high computational burden. The details of DEEPSO-OL are explained in Section II.

In summary, this paper proposes a novel variant based on evolutionary PSO (EPSO). EPSO combines the idea of evolutionary computing and exploring capability of particles such that system parameters can self-evolve intelligently to adapt different problems [5][6]. Then, a differential evolution (DE) concept is adopted and hybridized into EPSO to become DEEPSO. The DE thrives on the idea of using macro-gradients to achieve progress in the search for the optimum. The DEEPSO uses the same concept, abandoning the idea of single particle memory, and replacing it with collective memory. In this work, it is further modified with a different neighborhood topology to explore the domain more efficiently. In addition, to enhance its exploitation, DEEPSO is integrated with the orthogonal learning (OL) strategy, to become DEEPSO-OL. With the assistance of orthogonal experimental design (OED), it generates promising solutions. OED is employed to identify the optimal combination levels through a relatively small number of experimental tests, thereby extracting more valuable information from past searches [7]-[9].

To evaluate the performance of DEEPSO-OL, an optimal power flow (OPF) and its variant, OPF incorporating wind power (WOPF) are proposed. The OPF aims to optimize the voltage stability, power losses, generation cost, and/or other pertinent factors in a manner that satisfies the system constraints. Such non-linear, non-convex problem is a very good real-world problem to apply modern heuristic optimization methods to search for near-optimum solution due to their efficiency [34][35]. Meanwhile, due to the increasing penetration of renewable energy sources, there is a great practical need to develop WOPF. Unlike traditional power sources, wind energy is highly uncertain and uncontrollable [10][11]. Since PSO, by nature, performs stochastic search in the solution domain, it is believed to have positive contribution to solving WOPF problem characterized by the uncertain wind power generation. The power output of wind farms fluctuates, and these fluctuations will have a large impact on the whole power system. Traditional deterministic optimization methods may not be able to account for these uncertainties and therefore result in suboptimal or even infeasible solutions. In contrast, evolutionary methods

embedded with stochastic search have been demonstrated to have the ability to counteract the uncertain nature in the optimization problem [36].

Lezama et al. [37] have formulated a competition framework for real-world complex power distribution network optimization where an aggregator tries to maximize its profit by selling and buying energy in day-ahead local market under uncertainties due to renewable energies, electric vehicles (EVs), etc. The purpose of the competition is to demonstrate the validity of stochastic optimization methods such as PSO, GA, ABC, etc., in solving the problems with reliable and satisfactory solution and much less computing time compared to traditional mathematical programming. The stochastic behavior of the system is simulated by creating scenarios with Monto Carlo simulation which can be decomposed into deterministic models [12].

In this paper, the wind power incorporated economic dispatch (ED) model by Yu and Bhattarai [13] is extended to OPF problem. The objective of the ED is to minimize generation costs while meeting the overall demand, which can be formulated as a linear programming problem. The major difference between ED and OPF is that the inequalities for OPF contains not only generators' output limits, but also the system parameters, such as transmission line capacity, bus voltage limits, transformer limits and reactive compensator limits. Instead of generating random varieties by inverse transform method and conducting Monte Carlo simulation, penalty cost is added to reflect the additional cost resulting from the uncertainty of wind power. Such mechanism avoids generating huge number of scenarios requiring high computational burden and provides penalty factors for decision maker to accurately model the wind power generating system. The cost resulting from uncertainty is comprised of two elements: surplus cost and wind power deficit cost.

To the best of our knowledge, the application of DEEPSO-OL in the power system area has not been developed yet, which is one motivation for this work to develop practical and real-world benchmark test problems. The algorithm is tested on a modified IEEE 30-bus test system. Therefore, this work has two primary objectives: 1) develop a novel variant of PSO, named DEEPSO-OL; and 2) develop real-world benchmark problems and test the proposed DEEPSO-OL on these problems. The main contributions of this paper are summarized as follows:

1) Proposed a novel PSO method based on orthogonal learning from the perspectives of evolving control parameters, deep search mechanism, perturbations on solutions and novel topology.
2) Implemented the proposed algorithm to a real-world non-linear optimization problem, OPF, and comparative study and sensitivity analysis are conducted to draw more insights of the new variant.
3) A wind energy conversion system model, WOPF, to harness wind energy efficiently is developed under the framework of the OPF problem.

The rest of the paper is as follows: Section 2 illustrates proposed DEEPSO-OL. Section 3 describes OPF and WOPF problems as well as how DEEPSO-OL is implemented. Section 4 provides detailed case studies, numerical results tested on the IEEE 30-bus system and comparison with other techniques and evaluates optimally scheduled wind power based on various penalty and reserve cost coefficients. Finally, the conclusion part is given in Section 5.

2. Proposed Methodology

In this section, to facilitate the understanding of DEEPSO-OL, standard PSO is introduced first, followed by evolutionary PSO, differential evolutionary PSO, and the OL. The DEEPSO-OL improves the search process by introducing adaptive weights, randomness on personal best, more efficient exploitation scheme, and stochastic

topology. These improvements enhance the balance between exploitation and exploration, avoid the local minimal, and simplify system parameter tuning which are the shortcomings of the standard PSO.

2.1 Particle Swarm Optimization (PSO)

Particle swarm optimization [1] is a stochastic search algorithm that was initially designed for continuous optimization problems. Many variants have also extended the abilities for discrete problems [40]. Each particle *i* repeatedly searches in the solution domain according to *velocity V* and *position X* (a feasible solution) update rules. The standard PSO (StdPSO) is described as follows:

$$X_i^{new} = X_i + V_i^{new} \tag{1}$$

$$V_i^{new} = w_{i0}V_i + R_1 w_{i1}(b_i - X_i) + R_2 w_{i2}(b_g - X_i) \tag{2}$$

Equation (2) consists of three components which can be intuitively interpreted as following: the first term denotes the particle's inertia, which directs it to move along its previous direction, while the inertia weight w_{i0} is used to regulate the impact of the previous velocity. The second term is the particle's memory to control the movement, which is affected by the *personal best*, b_i. The final term represents the collaboration of particles to control the movement, affected by *global best*, b_g, found by the whole swarm. The second and third terms represent the cognitive influence (CI) and social influence (SI) on the particle's movement respectively. The parameters w_{i1} and w_{i2} are the weights known as the acceleration coefficients (ACs) to regulate the impact of CI and SI, respectively; R_1, R_2 are random numbers generated from a uniform distribution in the range of [0,1], which introduce perturbation to help the algorithm avoid local minima. Variants on CI and SI to guide particles toward high-quality solutions is one of the main directions to obtain better performance. *Position X* will be assessed to obtain the fitness value by the objective function $f(\cdot)$ at each iteration.

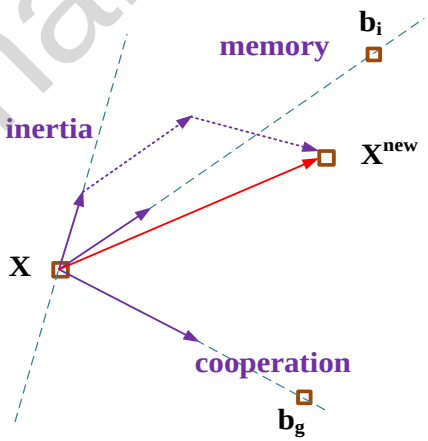


Fig. 1. Illustration of standard PSO.

Fig. 1 displays how a new particle is influenced by inertia (moving in the same direction), memory (affected by the particle's past best position), and cooperation (attracted to the global best position). Many literatures have discussed the importance of balancing exploration and exploitation [2][14], which was the motivation to design innovative variants over the years.

2.2 *Differential Evolution Evolutionary PSO (DEEPSO)*

Evolutionary PSO (EPSO) has a similar structure on velocity update compared with the StdPSO, but the parameters will undergo automatic evolution, guided by a constant mutation rate, during the search process.

$$V_i^{new} = w_{i0}^* V_i + w_{i1}^* (b_i - X_i) + w_{i2}^* (b_g^* - X_i) \tag{3}$$

$$b_g^* = b_g + w_{i3}^* N(0,1) \tag{4}$$

$$w_{ik}^* = w_{ik}[log\ N\ (0,1)]\tau \tag{5}$$

where w_{ik}^* (w_{i0}^*,w_{i1}^*…) are weights and τ is a constant mutation rate. Note that in (5) weights are not constant as opposed to the StdPSO but are under the *mutation* process dependent on a log distribution with mean 0 and variance 1. This improvement lies in the design Aspect 1 mentioned above for control parameters. The global best b_g^* is modified by adding a normally distributed (mean 0 and variance 1) random variable to the original b_g. Thus, EPSO is also guided by three components, inertia, perception (different from the StdPSO which consists of memory as the second component) and cooperation. Note that even the second term in Eq. (3) still uses particle's *personal best* and yet this term is self-evolving because of the weight w_{i1}^*. In addition, cooperation term is attracted by the approximate *global best* rather than the real one, which can help the swarm escape local optima and explore more diverse regions of space. Such modification enhances the exploration ability to cover more possible solutions [22].

Differential evolution (DE) is employed to generate a new solution, X_i, which combines the information of two random individual population (X_{r1}, X_{r2}). This process supports the diversity of a population, thereby increasing the exploration in the solution domain. Miranda and Alves proposed the DEEPSO for the first time by combining the EPSO and DE [23]. DEEPSO was also successfully implemented to address voltage stability issue by Bai, Lee, and Eke [5]. Thus, the velocity update function is now defined as:

$$V_i^{new} = w_{i0}^* V_i + w_{i1}^* (X_{r1} - X_i) + w_{i2}^* (b_g^* - X_i) \tag{6}$$

Comparing (6) with (3), the form of DEEPSO is more like EPSO. The only variation is to substitute the personal best, b_i with X_{r1} in the second term as illustrated in Fig. 2.

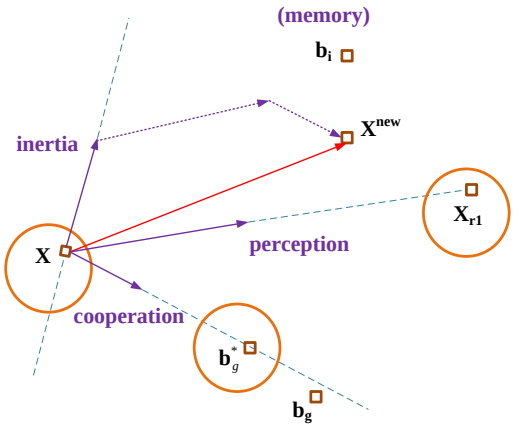


Fig. 2. Illustration of DEEPSO.

Fig. 2 illustrates a flavor of DE, where the X_{r1} information is added to the EPSO. Similarly, DEEPSO consists of inertia, perception, and cooperation terms.

As mentioned above, there are certain communication structures among particles, namely particles neighborhood topology. Classical communication structure is the *star shape* where all individuals get the same opportunities to know about the information of *global best* b_g. Another alternative is *ring shape* where each particle only knows its two neighbors' information as shown below in Fig. 3.

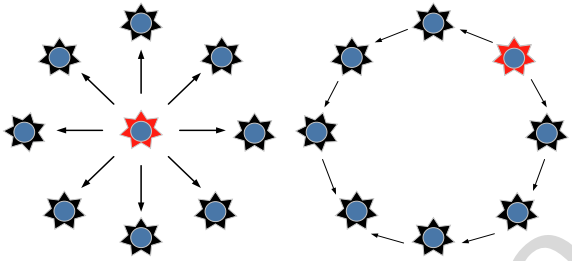


Fig. 3. Star and ring topology (red is the *global best*).

The conventional star configuration could result in premature convergence due to excessive communication that limits exploration of the search space, whereas the ring configuration runs the risk of leading the process towards a set of independent parallel searches due to insufficient information exchange. Therefore, in this paper, a stochastic star topology has been introduced to avoid traditional topologies' drawbacks where a communication probability matrix $\boldsymbol{P}$ is set to allow certain particles to access the *global best* as shown below:

$$V_i^{new} = w_{i0}^* V_i + w_{i1}^* (X_{r1} - X_i) + w_{i2}^* \boldsymbol{P}(b_g^* - X_i) \tag{7}$$

where **P** is the communication probability matrix, which is a diagonal matrix with 0 or 1 to decide if $X_i$can access the approximate *global best* b_g^* information, as the last term in (7) is expanded as:

$$[w_{1,2}^* \quad w_{2,2}^* \quad w_{3,2}^* \quad \cdots \quad w_{n,2}^*] \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} (b_g^* - X_1) \\ (b_g^* - X_2) \\ (b_g^* - X_3) \\ \vdots \\ (b_g^* - X_n) \end{bmatrix} \tag{8}$$

It is noted that the communication probability matrix **P** determines which $(b_g^* - X_i)$ term is multiplied by either one or zero, in other words, it determines which X_i can access the approximate *global best* information. Algorithm 1 illustrates a procedure for creating the communication matrix **P**:

Algorithm 1: Create communication P matrix

1. Given a probability threshold *x*
2. **for** *i* = 1 to swarm size *n*
3. **for** *k* = 1 to *n*
4. **if** $i \neq k$ then $P_{ik} = 0$
5. **else** $r = rand()$
6. **if** $r > x$, $P_{ii} = 0$ **else** $P_{ii} = 1$ **end if**
7. **end if**
8. **end for**
9. **end for**

It is important to introduce this communication probability matrix **P** to balance the exploration and exploitation because too much information from approximate *global best* b_g^* leans to exploitation in the solution domain. The impact of the various probability of exchanging information with neighbors is given in the sensitivity analysis under Section IV. In this work, rule (7) is further modified as following to gain a better performance by our empirical analysis:

$$\begin{Bmatrix} V_i^{new} = w_{i0}^* V_i + w_{i1}^*(X_{r1} - X_i) + w_{i2}^* \boldsymbol{P}(b_g^* - X_i) \\ V_i^{new} = w_{i0}^* V_i + w_{i1}^*(X_i - X_{r1}) + w_{i2}^* \boldsymbol{P}(b_g^* - X_i) \end{Bmatrix} \quad (9)$$

If the fitness value of the X_{r1} is less than that of X_i, e.g., $F(X_{r1}) < F(X_i)$, use the second equation in (9), otherwise, employ the first equation. Algorithm 2 gives the pseudocode of DEEPSO. Note that line 8 and 9 create new copies of current *V* and *X* using the mutated weights in line 6. From line 15 – 17, final population and other parameters are set after comparing the current population with the copied population.

Algorithm 2: DEEPSO

1. **Initialize** swarm (parameters, topology, population)
2. **Repeat** iteration *t*
3. **for** *i* = 1 to swarm size *n*
4. $V_i^{new} = w_{i0}^* V_i + w_{i1}^*(X_{r1} - X_i) + w_{i2}^* \boldsymbol{P}(b_g^* - X_i)$ – Equation (7)
5. $X_i^{new} = X_i + V_i^{new}$
6. Mutate weights (w_{i0}^*,w_{i1}^*,w_{i2}^*)
7. /*replicate *velocity* and *position* copies using new *w* */
8. $V_{Ci}^{new} = w_{ci0}^* V_{Ci} + w_{ci1}^*(X_{Cr1} - X_{Ci}) + w_{ci2}^* \boldsymbol{P}(b_g^* - X_{Ci})$ – Equation (7)
9. $X_{Ci}^{new} = X_{Ci} + V_{Ci}^{new}$
10. **end for**
11. Enforce X_i^{new}, V_i^{new}, X_{Ci}^{new}, V_{Ci}^{new} within feasible limits
12. **for** *i* = 1 to swarm size *n*
13. Compute $f(X_{Ci}^{new})$ and $f(X_i^{new})$ /*evaluate solutions*/
14. /*create new solution to replace the current one*/
15. **if** $f(X_{Ci}^{new}) < f(X_i^{new})$
16. $f(X_i^{new}) = f(X_{Ci}^{new})$, $X_i^{new} = X_{Ci}^{new}$, $V_i^{new} = V_{Ci}^{new}$, $w_{i0}^* = w_{ci0}^*$, $w_{i1}^* = w_{ci1}^*$, $w_{i2}^* = w_{ci2}^*$
17. **end if**
18. **end for**
19. **until** termination criterion is met
20. **return** *global best*

2.3 Orthogonal Learning

Orthogonal learning (OL) is the process based on orthogonal experimental design (OED) to get the best candidate with fewer combinations. OED is an experimental design used to study the effect of several factors simultaneously and the best combination of factor levels can be found in several tests. Table I gives an example of finding the best combination of ingredients to make bread.

TABLE I BEST BREAD EXPERIMENT

Factors / Levels	A Flour (lbs)	B Yeast (oz)	C Salt (oz)	D Water(oz)
1 *L*1	5	0.5	0.3	4
2 *L*2	4	0.6	0.4	3

In this experiment, there are four *factors* (optimization variables): flour (A), yeast (B), salt (C) and water (D) to make the bread. Each factor consists of two *levels*, e.g., the flour can be 5 or 4 lbs, salt can be 0.3 or 0.4 lbs, etc. Thus, there are total of $2^4 = 16$ combinations. Through the implementation of OL, it is possible to accurately predict the optimal combination by testing significantly fewer representative combinations, therefore OED can reduce the total testing cost. Those representative combinations are chosen according to the orthogonal array (OA). Details of OA and factor analysis (FA) are introduced as follows:

1) Orthogonal Array: firstly, '$L_N(s^d)$' is used to represent an array with *s* levels (possible values) in each factor under *d* factors (optimization variables). Then, *L* and *N* denote an array and the total combination numbers, respectively [7]-[9]. For example, $L_8(2^4)$ array given below contains 4 factors (optimization variables), 2 levels (possible values, 1 or 2) per factor, and 8 combinations.

$$L_8(2^4) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 2 \\ 1 & 2 & 2 & 1 \\ 1 & 2 & 2 & 2 \\ 2 & 1 & 2 & 1 \\ 2 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 \\ 2 & 2 & 1 & 2 \end{bmatrix} \tag{10}$$

The detailed definition and formulation of OA can be found in [32]. The idea is that instead of searching $2^4 = 16$ combinations exhaustively, OA only use 8 of them to design an experiment to predict the best combination.

OA is a predefined table for the OED method to work on, which is the fundamentals for defining representative combinations. Table II presents the eight experiments that are specified by $L_8(2^4)$. For example, the first row in (10) is [1 1 1 1], indicating that factors A (flour), B (yeast), C (salt), and D (water) are all set to the first levels (5lbs, 0.5oz, 0.3oz, 4oz) from Table I. The last column in Table II represents the evaluation of the experiments for each combination of ingredients using a fitness function that determines the level of deliciousness. The fitness function here is not a particular mathematical formula, but a professional judge who would rate combination of ingredients by outputting a fitness value based on his/her judgment. Higher value means the bread is more delicious.

TABLE II BEST COMBINATION LEVELS BY OED

Comb.	A: Flour	B: Yeast	C: Salt	D: Water	Fitness value
*Cb*1	(1) 5	(1) 0.5	(1) 0.3	(1) 4	$f_1 = 31$
*Cb*2	(1) 5	(1) 0.5	(1) 0.3	(2) 3	$f_2 = 54$
*Cb*3	(1) 5	(2) 0.6	(2) 0.4	(1) 4	$f_3 = 38$
*Cb*4	(1) 5	(2) 0.6	(2) 0.4	(2) 3	$f_4 = 53$
*Cb*5	(2) 4	(1) 0.5	(2) 0.4	(1) 4	$f_5 = 49$

*Cb*6	(2) 4	(1) 0.5	(2) 0.4	(2) 3	f_6 = 42
*Cb*7	(2) 4	(2) 0.6	(1) 0.3	(1) 4	f_7 = 57
*Cb*8	(2) 4	(2) 0.6	(1) 0.3	(2) 3	f_8 = 62
levels	Factor Analysis				
*L*1	$(f_1+f_2+f_3+f_4)/4=44$	$(f_1+f_2+f_5+f_6)/4=44$	$(f_1+f_2+f_7+f_8)/4=51$	$(f_1+f_3+f_5+f_7)/4=43.75$	
*L*2	$(f_5+f_6+f_7+f_8)/4=52.5$	$(f_3+f_4+f_7+f_8)/4=52.5$	$(f_3+f_4+f_5+f_6)/4=45.5$	$(f_5+f_6+f_7+f_8)/4=52.75$	
Results	A2	B2	C1	D2	

2) Factor Analysis: Factor analysis (FA) involves determining the optimal combination of levels (potential values). Based on the experimental results from OA with N cases, FA is performed to identify the optimal combination. Table II illustrates the FA process, with further details provided in [9]. As shown in Table II, the optimal combination determined by FA is (A2, B2, C1, and D2). It happens that the corresponding combination (4lb, 0.6oz, 0.3oz and 3oz) is *Cb8* in Table II. However, it is quite common to predict the best combination that may NOT appear in the original test table. Thus, as OL is implemented in the DEEPSO algorithm, the deeper search for the best candidate solution can be efficiently conducted by predicting the best combination of control variables as a solution vector. Constructing the best candidate is summarized in Algorithm 3:

Algorithm 3: Construct candidate solution by *OL*

1. **Generate** a transmission vector by (11)
2. **Choose** a solution vector randomly.
3. **/***The following steps are to mix and by making use of OL to construct a solution*/
4. **Generate** a 2-level OA $L_M(2^D)$, with $M = 2^{\lceil log_2(D+1) \rceil}$. ('⌈ ⌉' represents the ceiling bracket, which indicates rounding a number to the closest integer towards ∞). *D* is the number of factors/optimization variables.
5. **Obtain** *M* test solutions Z_m $(1 \le m \le M)$ with the corresponding value of T_k and X_i according to OA, where *M* denotes for the total number of combinations by the OA.
6. **Evaluate** the fitness of each solution, $f(Z_m)$, $(1 \le m \le M)$, and record the best solution Z_b in accordance with the fitness values.
7. **Obtain** the best solution Z_p and evaluate the $f(Z_p)$ by FA.
8. **Adopt** Z_p or Z_b, whichever has better fitness value, as the new candidate solution vector X_s

2.4 DEEPSO with Orthogonal Learning (DEEPSO-OL)

From the above discussion and preparation, the following DEEPSO-OL is now proposed. To construct a candidate solution by OL, a transmission solution T_k is introduced:

$$T_k = X_i + rand(0,1) \times (b_g - X_i)$$
$$k \neq i \in [1, n] \quad (11)$$

where b_g is the global best; *k* and *i* are different particle indices in the swarm. The combination of information from T_k and X_i results in an improved candidate solution X_s. Algorithm 4 outlines the framework of the DEEPSO-OL algorithm:

Algorithm 4: DEEPSO-OL

1. **Initialize** swarm (parameters, topology, population)
2. **Define** a OL threshold *p*
3. **Repeat** iteration *t*
4. **if** rand () > *p*
5. /*use DEEPSO algorithm*/

6. **for** *i* = 1 to swarm size *n*
7. $V_i^{new} = w_{i0}^* V_i + w_{i1}^* (X_{r1} - X_i) + w_{i2}^* \boldsymbol{P}(b_g^* - X_i)$ – Equation (7)
8. $X_i^{new} = X_i + V_i^{new}$
9. Mutate weights (w_{i0}^*,w_{i1}^*,w_{i2}^*)
10. /*replicate *velocity* and *position* copies using new *w**/
11. $V_{Ci}^{new} = w_{ci0}^* V_{Ci} + w_{ci1}^* (X_{Cr1} - X_{Ci}) + w_{ci2}^* \boldsymbol{P}(b_g^* - X_{Ci})$ – Equation (7)
12. $X_{Ci}^{new} = X_{Ci} + V_{Ci}^{new}$
13. **end for**
14. Enforce X_i^{new},V_i^{new},X_{Ci}^{new},V_{Ci}^{new} within feasible limits
15. **for** *i* = 1 to swarm size *n*
16. compute $f(X_{Ci}^{new})$ and $f(X_i^{new})$ /*evaluate solutions*/
17. /*create new solution to replace the current one*/
18. **if** $f(X_{Ci}^{new}) < f(X_i^{new})$
19. $f(X_i^{new}) = f(X_{Ci}^{new}),\ X_i^{new} = X_{Ci}^{new}, V_i^{new} = V_{Ci}^{new}, w_{i0}^* = w_{ci0}^*, w_{i1}^* = w_{ci1}^*, w_{i2}^* = w_{ci2}^*$
20. **end if**
21. **end for**
22. **else**
23. /*construct OL based on DEEPSO*/
24. Construct candidate solution by *OL* by Algorithm 3
25. **end if**
26. **until** termination criterion is met
27. **return** *global best*

The overall structure can be divided into two parts. Line 4 – 22, is the DEEPSO, and line 23 – 25 is the DEEPSO-OL. Note that we introduce a pre-defined OL probability *p* in line 2 to control the process such that either DEEPSO or DEEPSO-OL is executed at each iteration to reduce computation cost. Therefore, the complexity of the algorithm mainly lies on function evaluation at line 16. Depending on probability *p*, the minimal total function evaluations is between 2×*n*×*iter* (fully DEEPSO) and 2×*n*×*M*×*iter* (fully DEEPSO-OL), where *n* is the number of the population, *iter* is the iteration numbers, and *M* is the number of rows in constructed OA. In addition, *M* depends on the dimension of optimization variables. It is noticed that if we decide to use full DEEPSO-OL, the computation burden is high, therefore, to find a good probability threshold *p* is crucial when scaling the algorithm. The way to evaluate solutions ($f(X_i^{new})$) in Algorithm 4 is done with the help of *AC Power flow*, specifically, solved by *Newton-Raphson* method [5]. Essentially, control variables as solutions are fed into the *AC Power flow* solver, then high-dimensional nonlinear functions are calculated to obtain state variables. Finally, objective functions, dependent on control and/or state variables, are constructed and calculated to evaluate the fitness.

2.5 Exploration and Exploitation Balance Analysis

To illustrate the exploration and exploitation properties of algorithms, we adopted the following equations and algorithm from [42] to quantify the balance performance. The results are presented in Section 4. To calculate the increase and decrease in the distance among search agents, a diversity measurement known as the dimension-wise diversity measurement is calculated by Equations 12 - 14 in each iteration. The diversity measurement will be used to calculate the exploration and exploitation in each iteration.

$$DIV_j = \frac{1}{N}\sum_{i=1}^{N}\left|median(X^j) - X_i^j\right| \quad (12)$$

$$DIV = \frac{1}{D}\sum_{j=1}^{D} DIV_j \quad (13)$$

$$DIV_{max} = max\{DIV^1, DIV^2, \cdots, DIV^{Maxiter}\} \quad (14)$$

where median (X^j) is the median of j^{th} dimension in the whole population. X_i^j is the j^{th} dimension of particle i; N is the total number of particles in the population and D is total number of the dimension for each particle. Algorithm 5 presents the pseudo-code for calculating diversity.

Algorithm 5: DIVERSITY CALCULATION

Input: *population X, N, D*
Repeat iteration *t*
while iter < *Maxiter*
  for *i* = 1 to *D*
    for *j* = 1 to *N*
      Calculate the diversity in each dimension DIV_j, by (12);
    end for
    Calculate the diversity of the entire population DIV by (13);
  end for
  iter = iter + 1;
end while
Calculate the highest diversity value DIV_{max} by (14);
return *DIV*, DIV_{max}

The exploration is calculated from (15), which is the ratio between the diversity in each iteration and the maximum attainable diversity. On the other hand, exploitation from (16) is just the complementary percentage to exploration, which reflects the difference between the maximum diversity and the current diversity of an iteration.

$$Exploration\% = \frac{DIV}{DIV_{max}} \times 100 \quad (15)$$

$$Exploitation\% = \frac{|DIV_{max} - DIV|}{DIV_{max}} \times 100 \quad (16)$$

3. Formulation of Problems

In this section, the OPF and WOPF problems are developed and then the process of applying DEEPSO-OL is described.

3.1 OPF and WOPF

The goal of conventional OPF is to optimize a power system's objective function by selecting control variable settings that satisfy network constraints and operational requirements. The mathematical formulation of this objective is:

$$min \quad f(x,u) \quad (17)$$

$$S.T. \quad g(x,u) = 0 \quad (18)$$

$$h(x,u) \leq 0 \quad (19)$$

where the control variable vector u comprises generator bus voltage, transformer setpoint, generator real power, and shunt compensator at specified buses. The state variable vector x encompasses real power at the slack bus, reactive power at the generator bus, voltage at the load bus, and transmission line capacity. Note that some of the state variables are used to construct objective functions. The set of equality constraints g includes power flow balance equations at each node, while the set of inequality constraints h comprises limits on generator real and reactive power, transformer setpoint, and shunt capacitor.

Usually, two distinct objective functions for fuel cost are considered, namely quadratic cost functions with and without valve point loading in equations (20) and (21), respectively.

$$\sum_{i=1}^{N_G} a_i + b_i P_{Gi} + c_i P_{Gi}^2 \tag{20}$$

$$\sum_{i=1}^{N_G} a_i + b_i P_{Gi} + c_i P_{Gi}^2 + \left| d_i \sin(e_i (P_{Gi,min} - P_{Gi})) \right| \tag{21}$$

where P_{Gi} denotes the active power on the *i*-th unit. Fig. 4 illustrates the impact of valve point loading on a quadratic cost function. In a power plant, steam is regulated by valves to enter the turbine through separate nozzle groups. Optimal efficiency is attained when each nozzle group operates at full output [24]. To achieve the maximum possible efficiency, valves must open in sequence, leading to a fluctuating cost curve as depicted in Fig. 4.

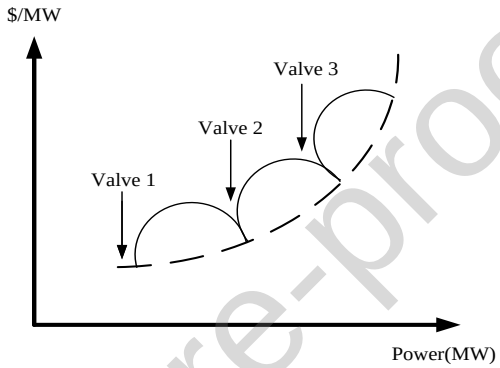


Fig. 4. Effect of valve point loading on a quadratic cost function.

The third objective function minimizes the total power loss as current flows along the transmission lines. It is defined as the total power loss:

$$\sum_{k=1}^{N_l} \frac{r_k}{r_k^2 + x_k^2} \left[V_i^2 + V_j^2 - 2 V_i V_j \cos(\delta_i - \delta_j) \right] \tag{22}$$

The fourth objective function minimizes the *L*-index which indicates the system's voltage stability. This index denotes the proximity of the system to collapse. The *L*-index varies between 0 (no load condition) and 1 (voltage collapse), and the bus with the highest *L*-index is the most susceptible bus. In other words, each bus has its *L*-index, and the one with the largest value needs to be minimized. The *L*-index for *j*-th bus is given as:

$$L_j = \left| 1 - \sum_{i=1}^{N_g} F_{ji} \frac{V_i}{V_j} \angle(\theta_{ji} + \delta_i - \delta_j) \right| \tag{23}$$

The control variables for OPF consist of the real power output on all generating buses except the slack bus, the voltages at all generating buses, the transformer taps, and the shunt capacitors, which are denoted as:

$$\left[P_{G,2} \cdots P_{G,i}, V_{G,1} \cdots V_{G,j}, T_1 \cdots T_i, Q_{C,1} \cdots Q_{C,i} \right] \tag{24}$$

The equality constraints *g* in equation (18) are the AC power flow balance equations at each bus, which state that the power flowing into a specific bus equals the power flowing out it. This is defined as:

$$P_i = V_i \sum_{j=1}^{N} V_j Y_{ij} \cos(\delta_i - \delta_j - \theta_{ij})$$

$$Q_i = V_i \sum_{j=1}^{N} V_j Y_{ij} \sin(\delta_i - \delta_j - \theta_{ij}) \quad \forall i, \forall j \tag{25}$$

The inequality constraints h (19) encompasses security constraints, transformer tap positions, generator limits, shunt capacitor limitations, and voltage and transmission line flow restrictions for the load buses.

Generator limits:

$$\begin{aligned} P_{Gi,min} &\le P_{Gi} \le P_{Gi,max} \\ Q_{Gi,min} &\le Q_{Gi} \le Q_{Gi,max} \\ V_{Gi,min} &\le V_{Gi} \le V_{Gi,max} \qquad i \in N_G \end{aligned} \tag{26}$$

Tap positions of transformers:

$$TP_{i,min} \le TP_i \le TP_{i,max} \qquad i \in N_T \tag{27}$$

Shunt capacitors constraints:

$$Q_{ci,min} \le Q_{ci} \le Q_{ci,max} \qquad i \in N_C \tag{28}$$

Security constraints on the bus voltage and transmission line flows:

$$\begin{aligned} V_{Li,min} &\le V_{Li} \le V_{Li,max} \qquad i \in N_{pq} \\ S_{Li} &\le S_{Li,max} \qquad i \in N_l \end{aligned} \tag{29}$$

where the fuel cost coefficients of the i-th unit are represented by a_i, b_i, c_i, d_i, e_i; P_{Gi}, which denotes the real power of the i-th unit; V_i symbolizes the voltage magnitude at bus i; r_k and x_k indicate the resistance and reactance of the transmission line k that connects bus i and j; V_i, V_j, δ_i and δ_j, which represent the voltages and angles at bus i and j, respectively; ω is the weighting factor; Y_{ij} and θ_{ij} are the Y-bus admittance matrix elements between bus i and j; N_{pq} denotes the number of PQ buses; N_l represents the total number of transmission lines; The variables in this equation include: N_G represents the number of generators; N_T denotes the number of tap-changing transformers; $P_{Gi,min}/P_{Gi,max}$, $Q_{Gi,min}/Q_{Gi,max}$ and $V_{Gi,min}/V_{Gi,max}$, which signify the minimum/maximum real power, reactive power, and voltage limits of generating unit i; $T_{Pi,min}/T_{Pi,max}$ and $Q_{ci,min}/Q_{ci,max}$ represent the limits for transformers and shunt capacitors, respectively; $V_{Li,min}$ and $V_{Li,max}$, which indicate the limits of load bus voltage; and $S_{Li,max}$ denotes the maximum line capacity of transmission line i.

The problem considers the real power generation of PV buses, voltage at all generator buses, transformer tap settings, and shunt compensators as control variables, which are initially assigned random values within the feasible domain. A penalty function is introduced to ensure that the dependent/state variables are also within the feasible domain, and to handle the inequality constraints. Specifically, the penalty function is defined and utilized as follows.

$$Pen(x_i) = \begin{Bmatrix} (x_i - x_{i,max})^2 \\ (x_{i,min} - x_i)^2 \\ 0 \end{Bmatrix} \qquad \begin{matrix} \text{if } x_i > x_{i,max} \\ \text{if } x_i < x_{i,min} \\ \text{if } x_{i,max} \le x_i \le x_{i,max} \end{matrix} \tag{30}$$

where $p(x_i)$ indicates that the penalty function of dependent variable x_i at bus i. The penalty cost increases quadratically when the dependent variables exceed their respective limits and zero, otherwise. Therefore, the augmented objective function is described by adding the penalty function for the slack bus, reactive power generation, PQ bus voltage, and transmission line capacity as follows:

$$F = f + C_p Pen(P_{G1}) + C_q \sum_{i=1}^{N_G} Pen(Q_{Gi}) + C_v \sum_{i=1}^{N_{pq}} Pen(V_{Li}) + C_s \sum_{i=1}^{N_l} Pen(S_{Li}) \quad (31)$$

where f is the original cost function (17), C_p, C_q, C_v and C_s are normally large values and they represent penalty factors for the real power generation of the slack bus, reactive power output of the generator buses, PQ bus voltage, and transmission line capacity, respectively. Obviously, if variables violate inequality limits, their corresponding cost function value are to be penalized to a large value, then the solution is more likely to be abandoned.

For those who are not in power systems background, readers can simply consider the objective functions as the evaluation functions where the outputs are the fitness values that justify the quality of input *position* vectors in DEEPSO-OL.

3.2 OPF Incorporating Wind Power (WOPF)

The essence of WOPF is to optimize the objective function (minimal loss, cost, stable voltage profile, etc.) when wind farms are connected at nodes in a power system grid. It raises another difficult question for system operators to operate grid effectively and reliably, because wind power is an intermittent source with uncontrollable nature. A common approach to tackle the stochastic behavior of the wind is to create scenarios and each scenario is considered as deterministic programming by Monte Carlo simulation [12]. Yet such an approach requires a lot of computations. However, this paper considers wind power uncertainty as a random variable and introduces an additional penalty cost in the unknown future of the wind power. This section incorporates wind power generators into the classical OPF problems to formulate WOPF.

The model was designed from the viewpoint of system operators (SOs). It is common that the SOs own assets such as conventional plants and/or wind farms. However, this paper assumes the SOs do not own any assets but merely manage power in the market. Since wind power is uncertain, there will be overestimation or underestimation compared to the power committed for next day. The overestimation (reserve cost) is when the actual wind power generation is short of the scheduled reference estimated, in which case, reserve power will be purchased from other sources to meet the deficiency, and otherwise load will be shed. Those activities lead to incremental costs for the SOs. When the actual wind power generation exceeds the expected planned reference generation, an underestimation (penalty cost) occurs, meaning that the SO has already purchased additional electricity that would not have been purchased from the wind farm, but must handle the actual remaining wind power. It is note that if SOs own wind farms, the cost of underestimating penalty will not exist. SOs usually sell additional wind power to adjacent power grids through re-dispatching. If none of the above methods can be achieved, excess energy must be released through a pseudo load resistor. In summary, these activities can be modeled by overestimating and underestimating the penalty cost function, and increase to power generation costs while supporting load demand and adhering to system constraints, as shown below:

$$F_{total} = \sum_{i}^{M} f_i(P_i) + \sum_{i}^{N} f_{\omega,i}(\omega_i) + \sum_{i}^{N} f_{p,\omega,i}(W_{i,av} - \omega_i) + \sum_{i}^{N} f_{r,\omega,i}(\omega_i - W_{i,av})$$
$$0 \le \omega_i \le \omega_{r,i} \quad (32)$$

where F_{total} is the objective function for WOPF; there are M thermal plants, and N wind farms; ω_i and $\omega_{r,i}$ are the *scheduled wind power* and *rated power* of the i-th wind generator, respectively; $W_{i,av}$ is a random variable with

probabilities varying with a Weibull probability density function (PDF) with values $0 \le W_{i,av} \le \omega_{r,i}$. The objective function includes four aspects:

1) The first term represents fuel cost for thermal plants defined in (20) and (21).
2) The second term represents direct cost of wind plants, in which $f_{\omega,_i}(\cdot)$is the cost function depending on ω_i. Here, Assume a value of zero for the sake of simplicity.
3) The third term represents the penalty for underestimating wind power, where $f_{p,\omega,_i}(\cdot)$is the penalty cost function depending on $W_{i,av}$ and ω_i.
4) The fourth term represents the penalty for overestimating wind power, where $f_{r,\omega,_i}(\cdot)$is the cost function for reserve cost. The derivation of those penalty functions is presented in the following.

In summary, the control vector *u* of WOPF consists of scheduled wind power ω_i, power generation at PV bus $P_{G,i}$, voltage at generator buses $V_{G,i}$, transformer taps T_i, and shunt capacitors $Q_{C,i}$, which is expressed as:

$$\left[\omega_1 \cdots \omega_i, P_{G,2} \cdots P_{G,i}, V_{G,1} \cdots V_{G,j}, T_1 \cdots T_i, Q_{C,1} \cdots Q_{C,i}\right] \tag{33}$$

The equality and inequality functions will remain unchanged except for the inclusion of an additional constraint for wind power, $0 \le \omega_i \le \omega_{r,i}$.

It is assumed that the underestimation penalty cost and the overestimation reserve cost have linear relationships with the gap between the actual and scheduled wind generation [13]. Then the penalty and reserve cost functions, respectively, can be calculated as:

$$f_{p,\omega,i}(W_{i,av} - \omega_i) = k_{p,i}(W_{i,av} - \omega_i) = k_{p,i}\int_{\omega_i}^{\omega_{r,i}}(\omega - \omega_i)f_W(\omega)d\omega \tag{34}$$

$$f_{r,\omega,i}(W_{i,av} - \omega_i) = k_{r,i}(\omega_i - W_{i,av}) = k_{r,i}\int_{0}^{\omega_i}(\omega_i - \omega)f_W(\omega)d\omega \tag{35}$$

where $k_{p,i}$ and $k_{r,i}$ are the cost coefficients for penalty and reserve, respectively, and $f_W(\omega)$ is the PDF of wind power. Note that the unit of cost coefficients is '$/h•MW'.

To assess both the reserve and penalty costs numerically, the PDF for the wind power output needs to be known. In general, the PDF of wind speed is following Weibull distribution, but to calculate the cost functions in WOPF, the wind power PDF $f_W(\omega)$ is derived from wind speed PDF, which is presented in Appendix B.

The wind integrated model developed is implemented in the IEEE 30-bus system, where generators at bus 2 and 5 are replaced by wind power generators to conduct numerical assessment of the WOPF, as illustrated in Fig. 5. Details of this case study will be described in Section IV.

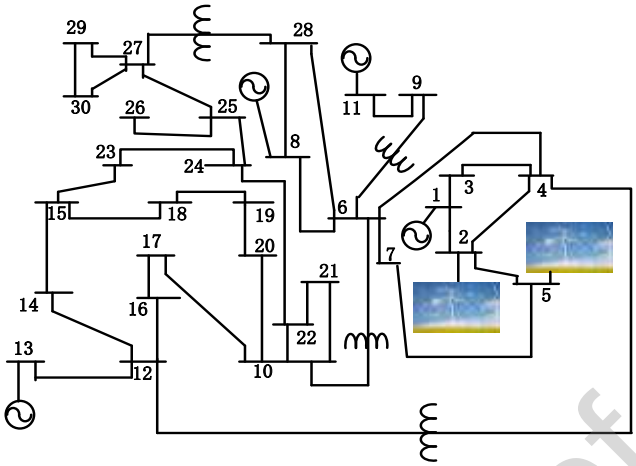


Fig. 5. Modified IEEE 30-bus.

4 Case Studies

The DEEPSO-OL has been implemented in a modified IEEE 30-bus system to demonstrate its performance. Comparisons are made with other modern heuristics, and sensitivity analysis as well as statistical analysis are performed. The computer used for simulation work has a 2.4 GHz Intel core i9 Processor and 64 GB RAM. The power flow was computed by the MATPOWER package [25]. Two cases will be studied, Case 1 for OPF and Case 2 for WOPF.

4.1 Case 1: OPF on IEEE 30-bus system

Case 1 is a standard OPF problem which consists of four scenarios corresponding to four objective functions denoted in (20) - (23). The IEEE 30-bus system data, control variables, and cost coefficients can be found in [26]. Lines 4-12, 6-9, 6-10, and 28-27 are equipped with tap-changing transformers to adjust voltage, and buses (10, 12, 15, 17, 20, 21, 23, 24 and 29) are installed with shunt capacitors to support reactive power. The system operates on a 100 MVA base and has an active power demand of 2.834 p.u. and a reactive power demand of 1.262 p.u. Further details are provided in Table III.

Table III IEEE 30-bus System Characteristics

Variables	Values	Details
Buses	30	Ref [27]
Branches	41	Ref [27]
Generators	6	Bus 1, 2, 5, 8, 11 and 13
Load voltage limits	24	[0.95 – 1.05]
Shunt Cap	9	Bus 10, 12, 15, 17, 20, 21, 23, 24 and 29
Xfmr Tap	4	Branches 4-12, 6-9, 6-10, 28-27
Control variables	24	N/A

Simulations are run 30 times to obtain statistical conclusion. Results from other methods were also presented to compare the results, such as basic artificial bee colony (ABC), differential evolution (DE), PSO, DEEPSO, adaptive constraint differential evolution (ACDE), improved NSGA-III, adaptive group search optimization (AGSO), sine-cosine algorithm (SCA), and modified sine-cosine algorithm (MSCA) [28]-[32]. Especially, DE is known for its effectiveness on continuous optimization problem [32] and thus there are four DE related algorithms

chosen for comparison. Those are the state-of-the-art algorithms published within the last five years in high-quality journals. To have fair comparison, the max iteration is set to 200. The comparison including execution time and the number of function evaluations are given for four scenarios in Table V. Cost coefficients of Case 1 and Case 2 are listed in Table IV [28]. Note that for Case 1, only coefficients a, b and c are used.

Table IV Cost coefficients for Cases 1 and 2

	a($/h)	b($/MWh)	c($/MW^{2}h)	d($/h)	e(rad/MW)
Bus 1	0	2	0.00375	18	0.037
Bus 2	0	1.75	0.0175	16	0.038
Bus 5	0	1	0.0625	14	0.04
Bus 8	0	3.25	0.00834	12	0.045
Bus 11	0	3	0.025	13	0.042
Bus 13	0	3	0.025	13.5	0.041

Scenario 1 is for the minimization with the quadratic fuel cost (20). Table V shows the comparisons of various algorithms for four scenarios respectively. From Table V, it is found that the minimum cost of DEEPSO-OL is 800.411 $/h, with 0 standard deviation. In Scenario 2, all buses with generating units have used the fuel cost function (21) with valve point loading. The minimum total fuel cost from DEEPSO-OL is 830.391$/h, with 0.02 standard deviation. Scenario 3 is for the minimization of the total transmission loss (22). Table V lists that the minimal line loss found by the proposed method is 3.021 MW, with 0 standard deviation. Scenario 4 is for the minimization of the *L*-index (23). Table V shows that the minimal *L*-index is 0.111, with 0.005 standard deviation.

It is interesting to note that results on some published papers seem to be promising at the first glance, and yet there are either some control variables or state variables violating their limits [31][32]. The common reason is that the papers failed to consider voltages constraints (0.95 – 1.05) on each PQ buses. In Appendix A, results of DEEPSO-OL from four scenarios are posted for cross check, and it is shown that the control variables are within constraints.

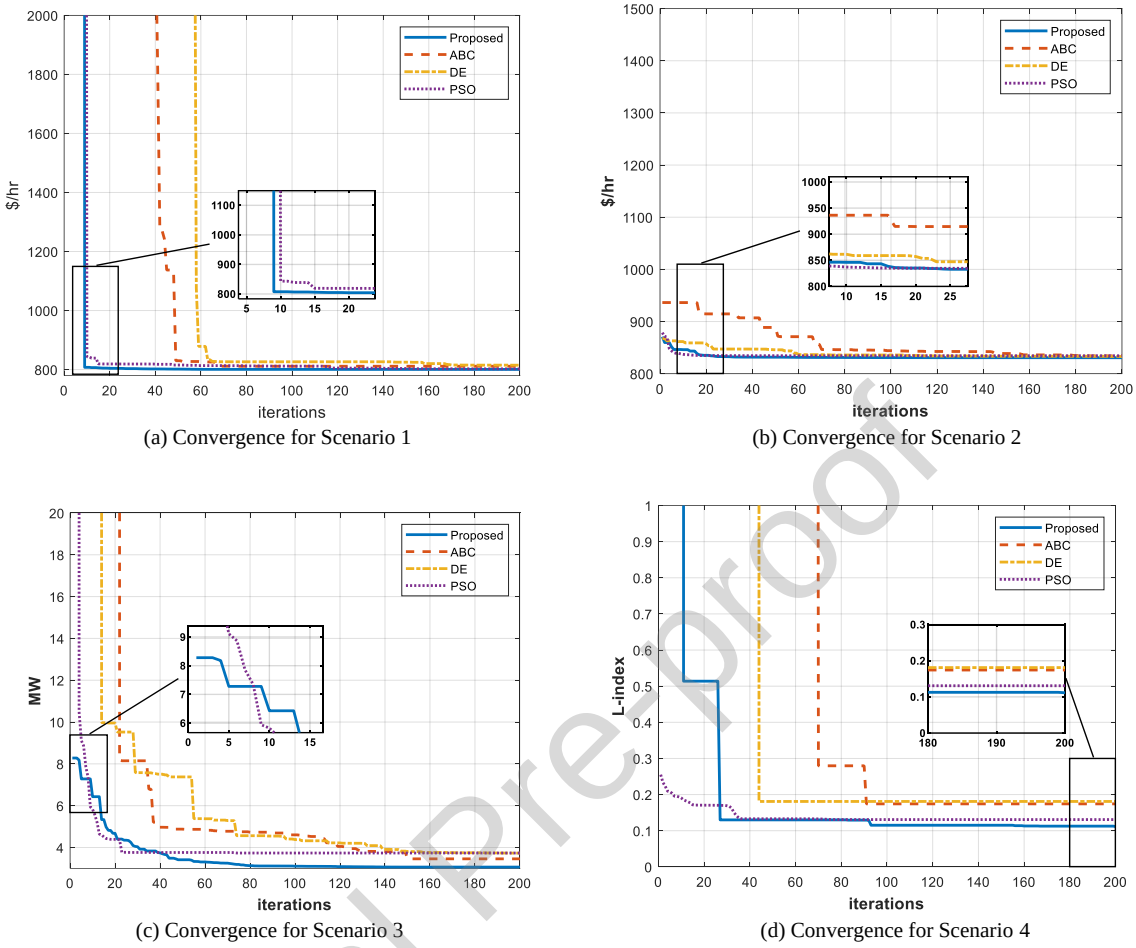
Table V Comparison by algorithms of 4 scenarios in IEEE 30-bus

Scenario 1	Fuel cost ($/h)					
Method	Min	Avg.	Max	Standard deviation	T(s)	Function evaluated
DEEPSO-OL	800.411	**800.413**	800.418	**0.00**	**33.3**	**15900**
DEEPSO	800.501	801.742	803.218	0.77	67.2	27900
ABC	800.707	802.262	803.411	0.82	42.2	20065
DE	802.629	803.031	803.509	0.25	39.8	20000
PSO	800.648	802.101	804.201	1.11	41.6	20100
ACDE [28]	800.411	**800.413**	800.418	**0.00**	83.2	N/A
NSGA-III [29]	802.173	803.632	804.459	N/A	10.8	N/A
AGSO [30]	801.287	801.750	802.509	N/A	N/A	N/A
SCA [31] [a]	N/A	800.102	N/A	N/A	N/A	N/A
MSCA [31] [a]	N/A	799.310	N/A	N/A	N/A	N/A
SF-DE [32]	800.413	800.415	800.419	0.00	133.1	N/A
SP-DE [32]	800.429	800.468	800.441	0.01	120.1	N/A
MSA [33]	N/A	800.510	N/A	N/A	14.9	N/A
Scenario 2	Fuel cost considering valve effect ($/h)					
Method	Min	Avg.	Max	Standard deviation	T(s)	Function evaluated
DEEPSO-OL	830.391	**830.396**	830.483	**0.02**	**42.6**	**16860**
DEEPSO	830.469	831.558	833.433	0.58	70.1	28100

ABC	831.125	834.081	838.326	1.71	50.4	20050
DE	832.483	832.483	832.483	0.26	45.9	20000
PSO	832.582	832.753	835.383	0.83	44.6	20100
ACDE [28]	832.072	832.096	832.394	0.06	81.5	N/A
SF-DE [32]	832.088	832.106	832.129	0.02	137.6	N/A
SP-DE [32]	832.481	832.655	832.876	0.09	141.7	N/A
Scenario 3		Total loss (MW)				
Method	Min	Avg.	Max	Standard deviation	T(s)	Function evaluated
DEEPSO-OL	3.021	**3.021**	3.032	**0**	**38.8**	**14704**
DEEPSO	3.024	3.035	3.138	0.023	70.8	27667
ABC	3.112	3.322	3.595	0.112	51.9	20077
DE	3.276	3.311	3.371	0.03	46.5	20000
PSO	3.051	3.072	3.331	0.051	54.7	20100
MSA [33]	N/A	3.101	N/A	N/A	N/A	N/A
ACDE [28]	3.084	3.085	3.086	0	82.3	N/A
SF-DE [32]	3.084	3.086	3.086	0.003	84.5	N/A
SP-DE [32]	3.084	3.085	3.086	0.003	136.4	N/A
Scenario 4		Minimize *L*-index				
Method	Min	Avg.	Max	Standard deviation	T(s)	Function evaluated
DEEPSO-OL	0.097	**0.111**	0.121	0.005	**44.3**	**15100**
DEEPSO	0.106	0.165	0.222	0.029	89.9	28500
ABC	0.108	0.165	0.255	0.038	83.6	20090
DE	0.146	0.152	0.156	**0.003**	59.2	20000
PSO	0.106	0.131	0.181	0.016	80.3	20100
SF-DE [32]	N/A	0.137	N/A	N/A	136.5	N/A
SP-DE [32]	N/A	0.137	N/A	N/A	130.7	N/A

[a] infeasible solutions due to the violation of state variables (exceeding load bus voltage limits)

Table V presents various statistics of algorithms including minimum, average, maximum, standard deviation, computation time and total number of functions evaluated. Since all scenarios are minimization problems and thus the smaller value they found, the better performance. A solution is feasible means that not only this solution is in its feasible domain, but other state variables depending on the solution are within limits as well. Some algorithms listed in Table V were infeasible according to literature. In all, we consider that if the average fitness values over 30-run is less, the solution is more accurate; if the standard deviation is less, the algorithm is more robust and if computation time and total function evaluated are less, the algorithm is more efficient.



(a) Convergence for Scenario 1

(b) Convergence for Scenario 2

(c) Convergence for Scenario 3

(d) Convergence for Scenario 4

Fig. 6. Convergence properties for case 1 on 4 scenarios.

Results in Scenario 1 show that the proposed method found the least minimal cost compared to other approaches in the article and especially the '0' standard deviation proves its robustness. Function evaluation times denotes that the proposed algorithm can find the best solution in a much less function evaluation process, which demonstrates its effectiveness.

Results in Scenarios 2, 3 and 4 show that the proposed method outperforms other methods with least standard deviation (except Scenario 4), time, and function evaluations. Especially in Scenario 3, a significant improvement in average value by 9.06% was found compared with the original ABC, which demonstrated the exploitation power of the proposed method in a more complex solution domain.

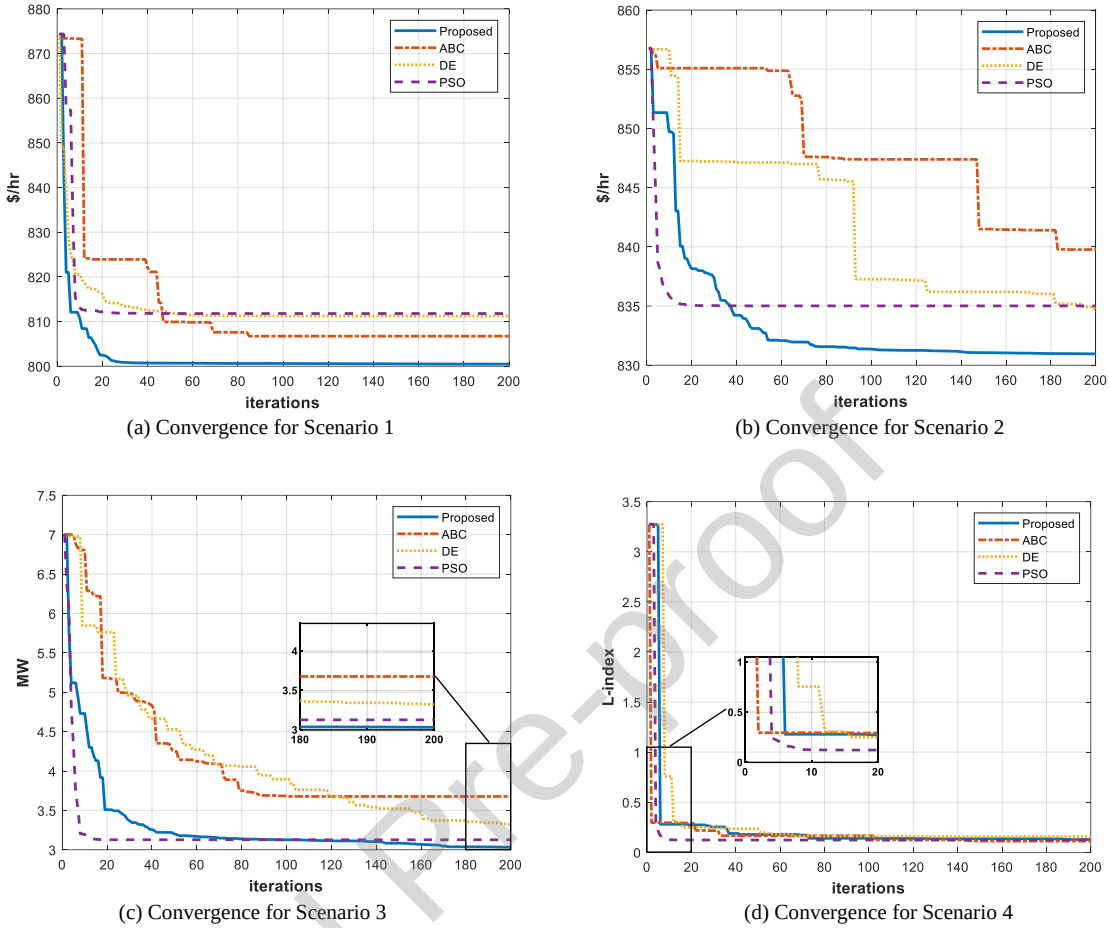
Fig. 6 shows the convergence properties of ABC, DE, PSO and DEEPSO-OL. For all scenarios, in total 200 iterations the proposed algorithm not only outperforms others in the convergence speed, but also the initial solution shows higher quality than others in Scenarios 2 and 3. This is because perturbation has been added to the acceleration coefficients ω_1 and ω_2 and global best particle, b_g. This is to increase the varieties of initial solutions such that they may contain more feasible information.

To further compare and obtain more information on the performance of the algorithms, we ensure the same initial solution as a starting point for all algorithms, as shown in convergence plot Fig. 7. Also, we purposely set the number of function evaluation to be around 10000 such that it terminates at an early stage as shown in Table VI. By doing so, we can have a fair comparison on the performance and see which algorithm can find the best

solution efficiently while performing relatively low function evaluation. Note that the number of function evaluation is not exactly 10000 for all, but close to 10000.

Table VI Comparison by algorithms for 4 scenarios in IEEE 30-bus

Scenario 1	Fuel cost ($/h)					
Method	Min	Avg.	Max	Standard deviation	T(s)	Function evaluated
DEEPSO-OL	800.423	**800.437**	800.472	**0.01**	21.2	**9780**
DEEPSO	800.501	801.312	802.218	0.72	20.2	10100
ABC	803.701	806.893	811.018	2.07	23.4	10050
DE	801.182	802.694	807.026	1.23	**18.9**	10000
PSO	801.991	802.141	802.142	0.03	21.8	9999
Scenario 2	Fuel cost considering valve effect ($/h)					
Method	Min	Avg.	Max	Standard deviation	T(s)	Function evaluated
DEEPSO-OL	830.391	**830.485**	832.724	**0.42**	23.2	**9850**
DEEPSO	830.969	832.578	834.433	0.79	**20.5**	10100
ABC	832.146	836.706	845.471	3.41	25.4	9950
DE	832.788	833.483	838.401	1.11	24.9	10000
PSO	832.582	833.122	836.973	1.14	21.9	10000
Scenario 3	Total loss (MW)					
Method	Min	Avg.	Max	Standard deviation	T(s)	Function evaluated
DEEPSO-OL	3.025	**3.024**	3.067	**0.050**	24.6	**9925**
DEEPSO	3.074	3.095	3.188	0.087	23.8	10100
ABC	3.281	3.856	4.776	0.348	25.8	10055
DE	3.344	3.487	3.576	0.060	**22.6**	10000
PSO	3.051	3.083	3.271	0.062	26.7	10100
Scenario 4	Minimize *L*-index					
Method	Min	Avg.	Max	Standard deviation	T(s)	Function evaluated
DEEPSO-OL	0.108	0.152	0.226	0.030	32.3	**9772**
DEEPSO	0.111	0.195	0.282	0.079	**30.9**	10100
ABC	0.100	**0.135**	0.173	**0.014**	39.9	10053
DE	0.117	0.171	0.241	0.030	32.2	10000
PSO	0.116	**0.135**	0.191	**0.014**	38.5	9999



(a) Convergence for Scenario 1

(b) Convergence for Scenario 2

(c) Convergence for Scenario 3

(d) Convergence for Scenario 4

Fig. 7. Convergence properties for case 1 on 4 scenarios with similar initial solution.

From Table VI, we observed that our proposed algorithm outperformed the rest of the algorithms in average and stand deviation metrics except for scenario 4. It appears that the proposed algorithm requires more function evaluations to achieve stable result for scenario 4. Fig.7 presents the convergence properties for all scenarios with similar initial solution. We can see that the proposed algorithm shows fast and stable convergence for scenario 1; For scenarios 2 and 3, it's not the fastest one, yet it was able to achieve the best objective value in the end; the proposed algorithm's convergence in scenario 4 was not able to stand out. Note that the function evaluation is still around 10,000 for each algorithm even though the total iteration is 200, which was achieved by adjusting the number of populations.

4.2 Sensitivity Analysis to DEEPSO-OL Control Parameters

Since two control parameters, the population size *N* and communication probability *p*, are essential in DEEPSO-OL, the effects of these two parameters will be discussed. The following figures illustrate the impacts of the two control parameters by heatmaps.

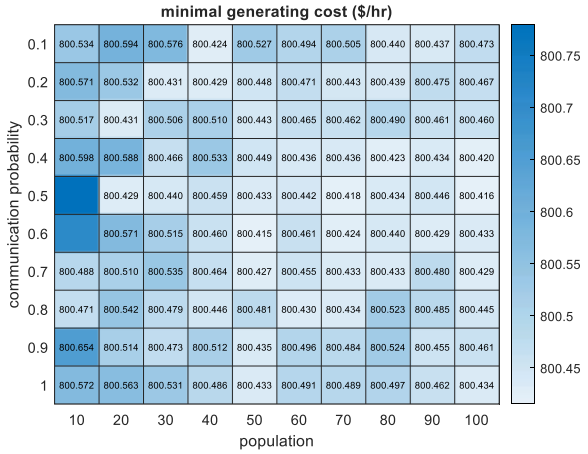


Fig. 8. Sensitivity heatmap on size *N* and probability *p* for Scenario 1.

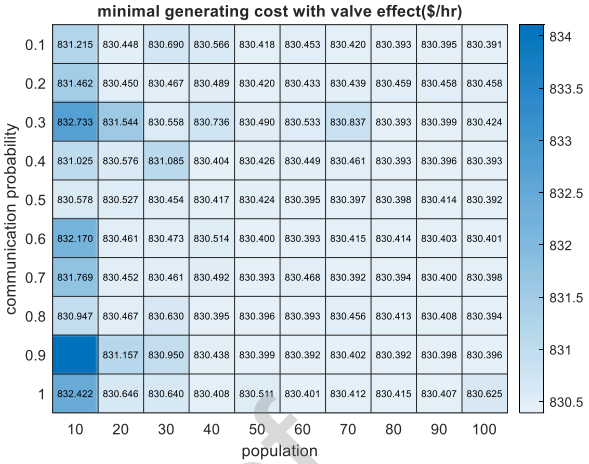


Fig. 9. Sensitivity heatmap on size *N* and probability *p* for Scenario 2.

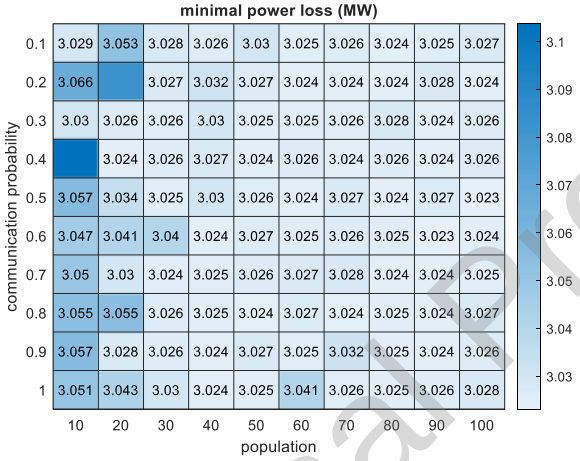


Fig. 10. Sensitivity heatmap on size *N* and probability *p* for Scenario 3.

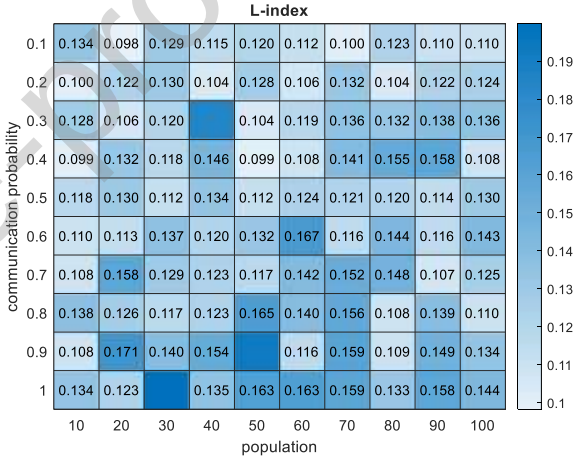


Fig. 11. Sensitivity heatmap on size *N* and probability *p* for Scenario 4.

Fig.8 shows the sensitivity heatmap for Scenario 1. In all, as population size and communication probability vary, the results are stable. The best solution (minimum generation cost) can be found as 800.416 $/h at N = 100 and p = 0.5. Fig. 9 illustrates the proposed algorithm is consistent with results of Scenario 1 as population and communication probability vary. The best solution can be found as 830.391 $/h at N = 100 and p = 0.1. Fig. 10 shows that the minimal power loss 3.023 MW is the best result found by (N = 90, p = 0.6) and (N = 100, p = 0.5). It is also noticed that when N = 20 and p = 0.4, the best solution is 3.024 which offers a good trade-off between computational cost and best result. When population size increases, the performance would improve in general, as the color becomes lighter. Fig. 11 shows that as population and communication probability vary, the results also vary considerably. The best solution can be found as 0.098 at N = 20 and p = 0.1.

4.3 Balance Analysis

Recall that diversity, exploration, and exploitation are defined in section 2.5, and we plot the balance curves below as shown in Fig. 12. The population size is set to 20 and a total of 200 iterations is chosen for each algorithm for four scenarios.

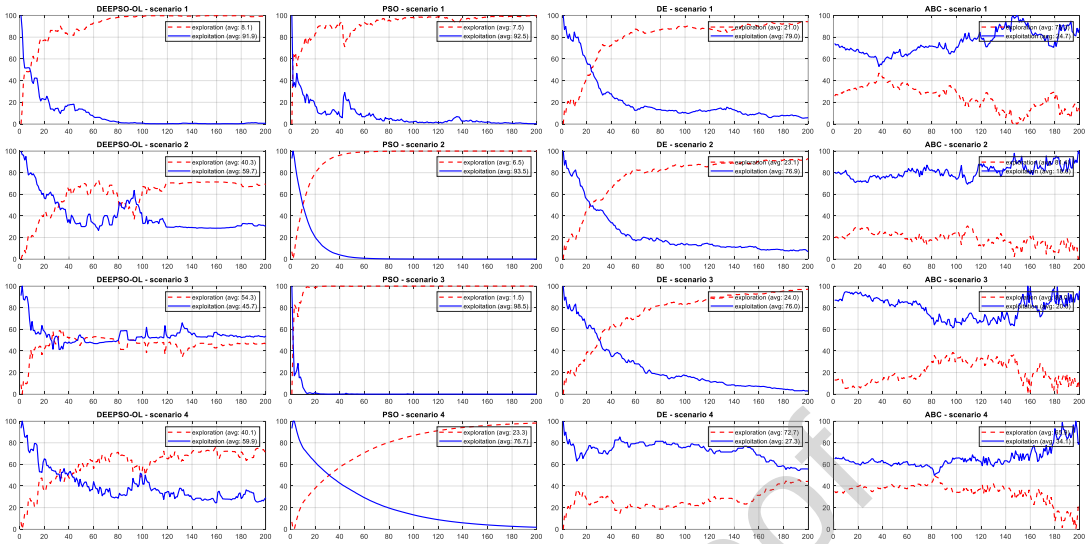


Fig. 12. Balance Analysis

It is observed from Fig.12 that the proposed method achieves a better balance on scenario 2, 3 and 4 compared with the rest algorithms indicated by the average exploration and exploitation values. The proposed method for scenario 1 does not seem to achieve a good balance since the average exploration is 8.1% and exploitation is 91.9%, yet it still outperforms the rest of the algorithms from Table V and VI. Even though researchers have identified the importance of balancing exploration and exploitation, and yet the relationship between diversity and exploration and exploitation is still unclear and more research is needed [41]. In other words, there is not always a positive correlation between diversity measures and fitness. Note that exploration and exploitation equations in this work are also derived based on diversity, to achieve a better understanding of this, we will conduct more research and theoretical analysis in our future word.

4.4 Statistical Analysis

Statistical analysis was conducted to draw more insights on the comparisons over 30 times run. Figs. 13 – 16 display the box plot for the comparison and Table VII shows the one-tail paired *t*-test to determine if the proposed algorithm is considered improvement statistically. Box plot is a statistical tool to visually show the mean, variance, 1st and 3rd percentile and maximum or minimum of a group of values. One-tail paired *t*-test is to test if one group's mean is larger or less than the other group's mean statistically.

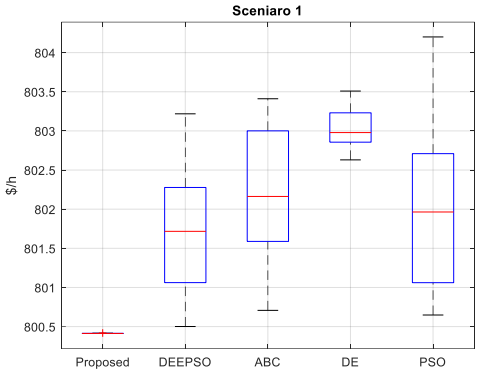


Fig. 13. Boxplot for Scenario 1.

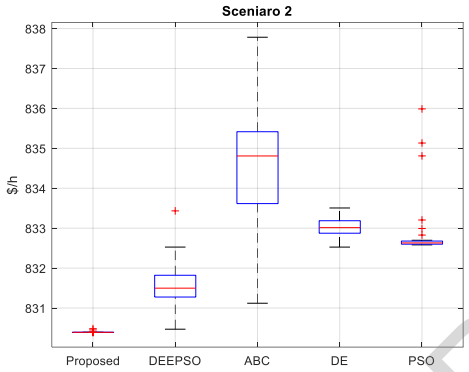


Fig. 14. Boxplot for Scenario 2.

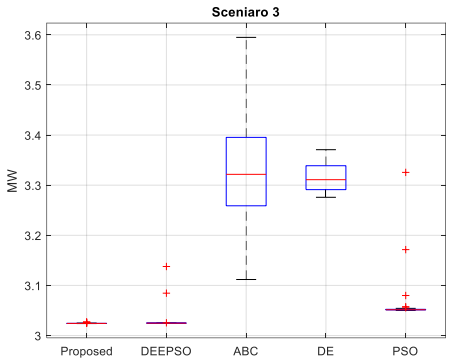


Fig. 15. Boxplot for Scenario 3.

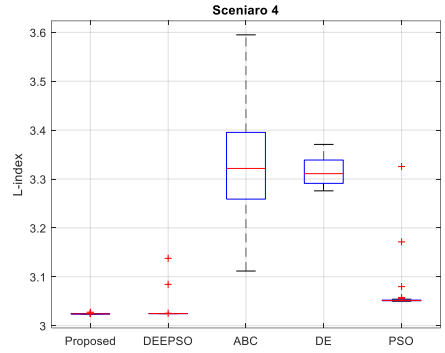


Fig. 16. Boxplot for Scenario 4.

The first three scenarios' boxplots have demonstrated that the proposed method in the first column is the most robust with almost zero standard deviation and efficient with least median values. The second-best algorithm is DEEPSO. Another observation is that even though results from the original PSO for Scenarios 2, 3 and 4 seem to have good mean value and small standard deviation and yet they have multiple outliers which degrades the whole quality. A paired statistical *T* test is conducted to draw statistical conclusion shown in Table VII based on the comparison between proposed method and the second-best methods for each scenario. Note that the *T* test is conducted based on 30 simulations, and the best and average solution are listed for reader's references. There are only DEEPSO, ABC, PSO, DE developed by authors and thus available to conduct 30 simulations and yet other results from references do not provide 30 simulations. Therefore, the second-best methods are only chosen from the above ones. The null hypothesis (H_0) is defined such that the proposed method does NOT have significant improvement over the second-best method statistically. The alternative hypothesis (H_1) is the opposite statement.

TABLE VII PAIRED STATISTICAL T TEST

Cases	DEEPSO-OL		Second-best		P-value
	Best	Avg.	Best	Avg.	
1	800.411	800.413	800.501	801.742	1.03e-10
2	830.391	830.396	830.469	831.558	2.58e-12
3	3.021	3.021	3.024	3.035	1.94e-10
4	0.097	0.113	0.106	0.131	9.08e-11

The P-value is a measure of the probability of observing the given test statistics assuming that the null hypothesis is true. The P-value for each scenario is much less than 0.01, which means there is strong evidence at

99% confidence level to reject the null hypothesis. In other words, the proposed algorithm has significant improvement over the second-best method statistically.

4.5 Case 2: Wind Power Integrated OPF (WOPF)

In this case, wind generators were added to buses 2 and 5 in the IEEE 30-bus system. Two wind scheduled power are added in the control variables. The assumption is made that wind farms will consistently harness the maximum amount of available wind energy. Various reserve and penalty cost coefficients were selected to prove the system's performance. The unit for cost coefficients is $/h•MW and the economic impacts of reserve and penalty cost coefficients on power scheduling have been investigated in Fig. 17 and Fig. 18, respectively. Table VII gives the comparison in this case for the cost coefficients up to 0.2.

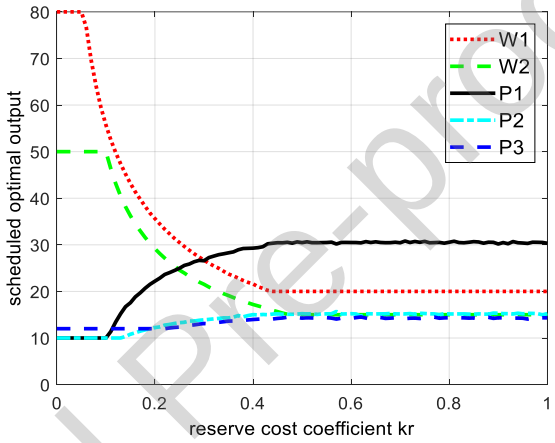


Fig. 17. Power output with respect to reserve cost coefficient k_r.

Fig. 17 shows the optimal scheduled power output for wind power W1 and W2, and conventional power P1, P2 and P3, on bus 2, 5, 8, 11, and 13, respectively. Note that the W1 and W2 have power output limits of 20-80 and 15-50 (MW), respectively. When penalty cost coefficient k_p is set to 0.1, while increasing k_r from 0 to 1, the output power from W1 and W2 decreases to their limits, because if the SO overestimates the wind power, it will be fined a big amount of expense due to the high reserve cost coefficient. Thus, as k_r goes high, the SO prefers to schedule less wind power.

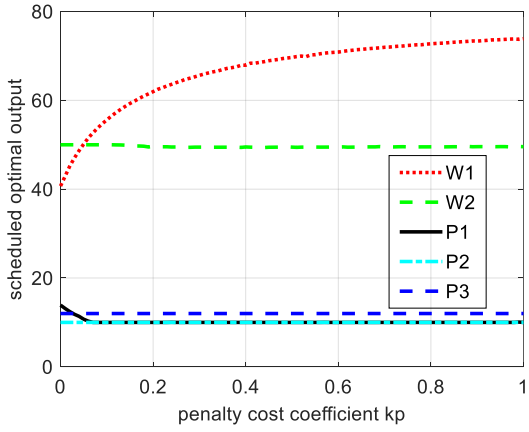


Fig. 18. Power output with respect to penalty cost coefficient k_p.

Fig. 18 shows the optimal scheduled power output for wind power W1, W2, and conventional power P1, P2 and P3. When reserve cost coefficient k_r is set to 0.1, while increasing k_p from 0 to 1, the output power from W1 increases gradually and W2 remains at its maximum power limit (50MW). Because if the SO underestimates the wind power, it will be fined a big amount of expense due to the high penalty cost coefficient; thus, as k_p increases, the SO prefers to schedule more wind power. Thus, the relationships between the optimal scheduled power output and cost coefficients can be used as references for the SO in making decision. It is notable that the values of cost coefficients depend on the local power market, wind source and reserved power cost, the criteria of choosing the values can be a further research topic and thus is not the focus of this paper. To further demonstrate the economic impact of WOPF, several different combinations of cost coefficients are presented, observing the power output, as shown in Table VIII.

TABLE VIII COMPARISON FOR CASE 2 WITH DIFFERENT PENALTY FACTORS

reserve factor ($/h•MW)	penalty factor ($/h•MW)	W1 (MW)	W2 (MW)	Fuel cost ($/h)				
				Min	Avg.	Max	Std.	T(s)
0.01	0.01	80	50	456.224	456.744	457.012	0.93	31.2
0.01	0.10	80	50	456.219	457.019	457.819	0.51	32.3
0.01	0.15	80	50	457.023	457.436	458.163	0.65	39.7
0.01	0.20	80	50	456.810	457.119	457.921	0.78	35.6
0.10	0.01	43.69	49.96	715.856	716.488	716.488	0.95	39.9
0.15	0.01	29.47	32.19	775.012	775.564	776.132	0.43	35.4
0.20	0.01	22.27	23.85	805.553	806.058	807.253	0.77	36.1

There are several observations on Table VIII. (1) When reserve factor remains 0.01 ($/h•MW), as noted in the first four rows, and penalty factor was being increased from 0.01-0.20 ($/h•MW), maximum wind power (80 and 50 MW) was scheduled by DEEPSO-OL to W1 and W2. This is because the actual wind power cannot align with the scheduled one, the cost of purchasing power from other source remains low (low reserve factor value). Therefore, regardless of what the penalty factor is, maximum wind power was scheduled in this situation. (2) When reserve factor starts to increase to higher values, for instance, 0.20 ($/h•MW) in the last row, the scheduled wind power output for both W1 and W2, as determined by DEEPSO-OL, was reduced to avoid overestimation, and associated high costs resulting from the implementation of high reserve factors. (3) Overall, the integration of wind power into the system will lead to a substantial reduction in total costs, being dependent on factors such as penalty and reserve coefficients, as well as the available wind resources.

Note that although there are various solutions for OPF with renewable energy resources, there is no standard model and test framework across the literature yet. In other words, each existing study may propose a unique model to incorporate wind energy, unique test circuits, etc. Thus, it is difficult to have apple to apple comparison. For instance, many works do not consider wind power output as control/optimization variables, but consider it as negative load, which is a known information obtained from forecast [38][39]. The wind power forecast also varies greatly depending on methodologies. Wind power forecast has two major approaches in terms of output: point forecast and probabilistic forecast. Point forecasting gives a single future wind power output. On the contrary, probabilistic forecast gives a conditional distribution of future wind power output [39], so that system operators and traders can utilize a much broader set of information. The wind power distribution can be further represented as quantiles, interval forecasts, PDFs and scenarios generated by Monte Carlo simulation [11][36][37]. The main advantage of WOPF model in this work is that the wind output, as part of control variables, is derived from wind speed PDF to have its analytic expression as described in Appendix B, which not only takes forecast error into

account but also requires much less computing power compared with Monte Carlo simulation method.

5 Discussion and conclusion

This paper summarized several categories by which PSO variants are developed and proposed a novel differential evolutionary PSO integrated with orthogonal learning (OL). It has proved to be promising in the balance between exploration and exploitation. The DEEPSO-OL is improved from aspects of improving control parameters by self-evolving on CI and SI coefficients, searching mechanism with OL, perturbation mechanism by adding noise on global best, and population topology with stochastic star shape. To evaluate its performance on a real-world problem, OPF and WOPF are developed accordingly. In OPF, there are four objective functions tested and compared with other state-of-the-art evolutionary computation algorithms published in recent years. The proposed algorithm outperformed all other algorithms in less computing time, less objective values, less standard deviation (exception on Scenario 4), and less function evaluations.

Also, by running 30 simulations, statistical analysis and *T* test were conducted to show that the DEEPSO-OL has significantly improved the performance compared with the second-best algorithm, especially on complex solution domains. Parameter sensitivity analysis is presented via heatmap which vividly demonstrates that even using less population, sound results can be obtained. WOPF was developed such that network constraints were also considered and DEEPSO-OL proves its effectiveness on the complex stochastic optimization problem by generating robust solutions in a reasonable time. Such a tool can be used to assess wind power integration and provide good insights for decision makers.

The trend of evolutionary computation research is to solve real-world complex problems. For example, the proposed algorithm can be potentially used to solve complex supply chain management problems where optimal production plan, allocation of distribution centers, and vehicle routes are to be determined under network and supply-demand balance constraints. This is a very complicated NP hard problem. This work not only shows the importance of balancing exploration and exploitation but also provides improvement guidance from four aspects mentioned in introduction for researchers. Yet, this work does not intend to imply that the more aspects algorithm gets tuned simultaneously, the better performance it will achieve, because algorithm should be tuned case by case to fit for specific problems. Therefore, further investigation can be done into the individual contribution of these four aspects. In addition, for all evolutionary computation-based solutions, they suffer from non-consistent results of each run, and computational burden. Computational burden may be compensated by stronger computing power and/or parallel computing. Therefore, for future works, authors aim to improve the robustness of this algorithm and use a larger system with more control variables test. Moreover, researchers in this community can think of more practical applications where they are too hard, if not impossible, to be solved by traditional methods and do not require consistent solution repeatedly.

CRediT authorship contribution statement

Wenlei Bai: Conceptualization, Methodology, Software, Validation, Investigation, Writing – original draft, Writing – review & editing. Fanlin Meng: Conceptualization, Investigation, Writing – review & editing. Ming Sun: Investigation, Writing – review & editing. Haoxiang Qin: Software, Writing – review & editing. Richard Allmendinger: Writing – review & editing. Kwang Y. Lee: Writing – review & editing, Supervision.

Declaration of competing interest

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

6 REFERENCES

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," *IEEE Proceedings of International Conference on Neural Networks*, Nov. 1995.

[2] T. M. Shami, A. A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M. A. Summakieh and S. Mirjalili, "Particle Swarm Optimization: A Comprehensive Study and Analysis," *IEEE Access*, vol. 10, pp. 10031 – 10061, 2022.

[3] D. H. Wolpert and W. G. Macready, "No free lunch theorem for optimization," *IEEE Trans. Evolutionary Computation*, vol.1, no.1, pp.67-82, April 1997.

[4] C. L. Camacho-Villalón, M. Dorigo, and T. Stützle, "PSO-X: A Component-Based Framework for the Automatic Design of Particle Swarm Optimization Algorithms," *IEEE Trans. Evolutionary*, vol.26, no.3, pp. 402-416, June 2022.

[5] W. Bai, K. Y. Lee, and I. Eke, "Optimal power flow considering global voltage stability based on a hybrid modern heuristic technique," 11th *IFAC Symposium on CPES 2022*, vol. 55, no. 9, pp. 413-418.

[6] V. Miranda, L. M. Carvalho, M. A. Rosa, A. M. L. Silva, and C. Singh, "Improving power system reliability calculation efficiency with EPSO variants," *IEEE Trans. Power System*, vol. 24, no. 4, pp. 1772 – 1779, 2009.

[7] W. F. Gao, S. Y. Liu, and L. L. Huang, "A novel artificial bee colony algorithm based on modified search equation and orthogonal learning," *IEEE Trans. Cybern.* vol. 43, no. 3, Jun. 2013.

[8] W. Bai, I. Eke, and K. Y. Lee, "An improved artificial bee colony optimization algorithm based on orthogonal learning for optimal power flow problem," *Control Engineering Practice*, vol. 61, pp. 163 – 172, 2017.

[9] Z. H. Zhan, J. Zhang, Y. Li, and Y. H. Shi, "Orthogonal learning particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 15, no. 6, pp. 832-847, Dec. 2011.

[10] A. K. Khamees, A. Y. Abdelaziz, M. R. Eskaros, M. A. Attia and M. A. Sameh, "Optimal power flow with stochastic renewable energy using three mixture component distribution functions," *Sustainability*, vol. 15, pp. 334-355, 2023.

[11] W. Bai, D. Lee, and K. Y. Lee, "Stochastic dynamic AC optimal power flow based on a multivariate short-term wind power scenario forecasting model," *Energies.*, vol. 10, pp.2138-2157, Dec. 2017.

[12] J. Wang, A. Botterud, R. Bessa, H. Keko, L. Carvalho, D. Issicaba, J. Sumaili, and V. Miranda, "Wind power forecasting uncertainty and unit commitment," *Applied Energy*, vol. 88, pp. 4014-4023, Apr. 2014.

[13] J. Hetzer, D. C. Yu, and K. Bhattarai, "An economic dispatch model incorporating wind power," *IEEE Trans. Energy Convers*., vol. 23, no. 2, pp. 603-611 Jun. 2008.

[14] J. C. Bansal, P. K. Singh, and N. R. Pal, *Evolutionary and Swarm Intelligence Algorithms, 1st ed.* New York, NY, USA: Springer, 2019.

[15] K. R. Harrison, A. P. Engelbrecht, and B. M. Ombuki-Berman, "Inertia weight control strategies for particle swarm optimization," *Swarm Intell.*, vol. 10, no. 4, pp. 267–305, 2016.

[16] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73, Feb. 2002.

[17] X. Zhang, P. Guo, H. Zhang, and J. Yao, "Hybrid particle swarm optimization algorithm for process planning," *Mathematics*, vol. 8, pp. 1745 – 1767, 2020.

[18] V. Miranda and N. Fonseca, "EPSO – best-of-two-worlds meta-heuristic applied to power system problems," *IEEE Congress on Evolutionary Computation*, vol. 2, pp. 1080-1085, 2002.

[19] P. K. Lehre and C. Witt, "Finite first hitting time versus stochastic convergence in particle swarm optimization," in *Advances in Metaheuristics*. New York, NY, USA: Springer, 2013, pp. 1–20.

[20] M. A. Montes de Oca, T. Stützle, M. Birattari, and M. Dorigo, "Frankenstein's PSO: A composite particle swarm optimization algorithm," *IEEE Trans. Evol. Comput*., vol. 13, no. 5, pp. 1120–1132, Oct. 2009.

[21] M. A. Montes de Oca, T. Stützle, K. Van den Enden, and M. Dorigo, "Incremental social learning in particle swarms," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 2, pp. 368–384, Apr. 2011.

[22] R. Storn and K. Price, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, pp. 341-359, 1997.

[23] V. Miranda and R. Alves, "Differential evolutionary particle swarm optimization (DEEPSO): a successful hybrid," *BRICS Congress on Computational Intelligence*, Lpojuca, Brazil, 2013.

[24] G. L Decker and A. D. Brooks, "Valve point loading turbines", *Electrical Engineering*, vol. 77, issue 6, pp. 501

[25] MATPOWER. Available: http://www.pserc.cornell.edu/matpower/

[26] K. Y. Lee, Y. M. Park, and J. Ortiz, "A united approach to optimal real and reactive power dispatch," *IEEE Trans. Power Apparatus Syst.* vol. 104, pp. 1147-53, May 1985.

[27] O. Alsac and B. Stott, "Optimal load flow with steady-state security," *IEEE Trans. Power Apparatus Syst.*, vol. 93, pp. 745-51, May 1974.

[28] S. Li, W. Gong, C. Hu, X. Yan, L. Wang, and Q. Gu, "Adaptive constraint differential evolution for optimal power flow," *Energy*, vol. 235, pp. 121362-121375, 2021.

[29] J. Zhang, S. Wang, Q. Tang, Y. Zhou, and T. Zeng, "An improved NSGA-III integrating adaptive elimination strategy to solution of many-objective optimal power flow problems," *Energy*, vol. 172, pp. 945-957, 2019.

[30] N. Daryani, M. T. Hagh, and S. Teimourzadeh, "Adaptive group search optimization algorithm for multi-objective optimal power flow problem," *Applied Soft Computing*, vol. 38, pp. 1012-1024, 2016.

[31] A. Attia, R. El Sehiemy, and H. M. Hasanien, "Optimal power flow solution in power systems using a novel Sine-Cosine algorithm," *Electrical Power and Energy Systems*, vol. 99, pp. 331-343, 2018.

[32] P. P.Biswas, P. N. Suganthan, R. Mallipeddi, and G. Amaratunga, "Optimal power flow solutions using differential evolution algorithm integrated with effective constraint handling techniques," *Electric Power Systems Research* vol. 142, pp. 190-206, 2017.

[33] A. A. Mohamed, Y. S. Mohamed, and A. El-Gaafary, "Optimal power flow using moth swarm algorithm," *Electric Power Systems Research* vol. 142, pp 190-206, 2017.

[34] W. Warid, "Optimal power flow using the AMTPG-Jaya algorithm," *Applied Soft Computing* vol. 91, pp. 106252 – 106261, 2020.

[35] Y. Muhammad, M. Raja, M. Altaf, F. Ullah, N. Chaudhary, and C. Shu, "Design of fractional comprehensive learning PSO strategy for optimal power flow problems," *Applied Soft Computing* vol. 130, pp. 109638 – 109655, 2022.

[36] A. Y. Rodríguez-González, F. Lezama, Y. Martínez-López, J. Madera, J. Soares, and Z. Vale, "WCCI/GECCO 2020 Competition on evolutionary computation in the energy domain: an overview from the winner perspective," *Applied Soft Computing* vol. 125, pp.109162 – 109174, 2022.

[37] F. Lezama, J. Soares, Z. Vale, J. Rueda, S. Rivera, and I. Elrich, "2017 IEEE competition on modern heuristic optimizers for smart grid operation: Testbeds and results. Swarm and evolutionary computation," vol. 44, pp. 420-427, 2019.

[38] W. Bai, X. Zhu, and K. Y. Lee, "Dynamic optimal power flow based on spatio-temporal wind speed forecast model," *IEEE Congress on Evolutionary Computation*, Krakow, Poland (Virtual), pp. 136-143, 2021.

[39] S. Li, W. Gong, L. Wang, and Q. Gu, "Multi-objective optimal power flow with stochastic wind and solar power," *Applied Soft Computing*, vol. 114, pp. 108045 – 108062, 2022.

[40] R. A. Gomez-Montoya, J. A. Cano, P. Cortes, and F. Salazar, "A Discrete Particle Swarm Optimization to Solve the Put-Away Routing Problem in Distribution Centres," *Computation*, vol. 8(4), pp. 99 – 116, 2020.

[41] E. Burke, S. Gustafson, and G. Kendall, "Diversity in genetic programming: An analysis of measures and correlation with fitness," *IEEE Trans. Evol. Comput*. vol. 8(1), pp. 47-62.

[42] Z. Chen, P. Xuan, A. A. Heidari, L, Liu, C. Wu, H. Chen, J. Escorcia-Gutierrez, and R. F. Mansour, "An artificial bee bare-bone hunger games search for global optimization and high-dimensional feature selection," iScience, vol. 26(5), pp. 106679 – 106717.

APPENDIX A: OPTIMAL SOLUTION OBTAINED BY DEEPSO-OL FOR IEEE-30 SYSTEM

Control Variables	Min	Max	Scenario 1	Scenario 2	Scenario 3	Scenario 4
P_{G2} (MW)	20	80	48.7869	43.8497	80.0000	31.0681
P_{G5} (MW)	15	50	21.3968	17.9695	50.0000	31.0547
P_{G8} (MW)	10	35	21.1653	10.0000	35.0000	22.2462
P_{G11} (MW)	10	30	11.9035	10.0000	30.0000	21.2139
P_{G13} (MW)	12	40	12.0001	12.0000	40.0000	18.0070
V_1 (p.u)	0.95	1.10	1.0815	1.0999	1.0700	1.0700
V_2 (p.u)	0.95	1.10	1.0626	1.0797	1.0660	1.0318
V_5 (p.u)	0.95	1.10	1.0313	1.0477	1.0470	0.9754
V_8 (p.u)	0.95	1.10	1.0355	1.0509	1.0531	0.9844
V_{11} (p.u)	0.95	1.10	1.0781	1.0977	1.0869	1.0894

V_{13} (p.u)	0.95	1.10	1.0566	1.0593	1.0667	0.9557
$T_{4\text{-}12}$ (p.u)	0.90	1.10	1.0154	1.0544	1.0103	0.9653
$T_{6\text{-}9}$ (p.u)	0.90	1.10	0.9618	0.9280	0.9631	0.9241
$T_{6\text{-}10}$ (p.u)	0.90	1.10	0.9806	0.9739	1.0017	0.9963
$T_{28\text{-}27}$ (p.u)	0.90	1.10	0.9707	0.9778	0.9760	0.9007
Q_{c10} (MVar)	0	5	4.8912	0.9427	0.6809	2.6581
Q_{c12} (MVar)	0	5	0.1059	0.1293	3.9682	1.6465
Q_{c15} (MVar)	0	5	2.2623	4.0402	4.2123	2.3399
Q_{c17} (MVar)	0	5	4.9342	4.9993	5.0000	1.6338
Q_{c20} (MVar)	0	5	4.4215	3.8652	3.6695	1.8388
Q_{c21} (MVar)	0	5	5.0000	4.9998	5.0000	2.3321
Q_{c23} (MVar)	0	5	2.9204	2.7587	2.5803	3.8374
Q_{c24} (MVar)	0	5	4.9777	4.9999	4.9998	0.5975
Q_{c29} (MVar)	0	5	2.4285	2.2443	2.1775	3.1007
Result			800.44	830.42	3.03	0.113

APPENDIX B: DERIVATION OF WIND POWER PDF

The PDF of wind speed is considered as Weibull distribution [13]:

$$f(v) = \left(\frac{k}{c}\right)\left(\frac{v}{c}\right)^{k-1}\left(e^{-(v/c)^k}\right) \qquad 0 < v < \infty \tag{C1}$$

where Weibull distribution can characterize the wind speed random variable by using different factor values *k* and *c*. Figs. C1 and C2 give the Weibull PDF functions for *k* = 1 and 2, respectively, with *c* = 10, 15, and 20.

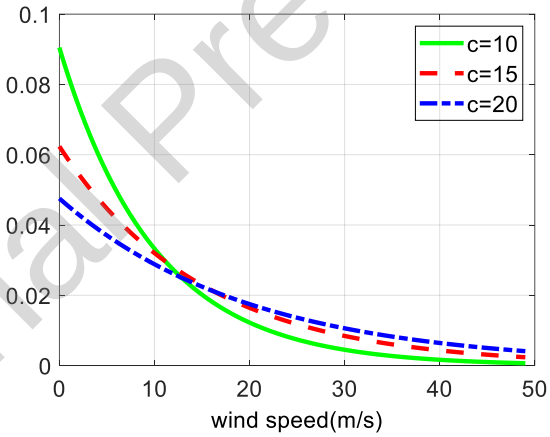


Fig. C1. Weibull PDF with *k* = 1.

The power output of wind farms is subject to randomness and can be derived through a transformation from wind speed. The relationship between wind turbine power and wind speed is expressed as [6]:

$$\omega = \begin{cases} 0, & v < v_n \text{ or } v > v_0 \\ \omega_v \frac{(v - v_0)}{(v_r - v_n)}, & v_n \le v \le v_r \\ \omega_{r,} & v_r \le v \le v_0 \end{cases} \tag{C2}$$

where v_r is the rated wind speed, and v_n and v_0 are cut-in and cut-out speeds, respectively. Given a Weibull distribution for a specific wind speed, Hetzer, Yu, and Bhattarai [13] provide a comprehensive guide for transforming the wind speed distribution into a wind power distribution:

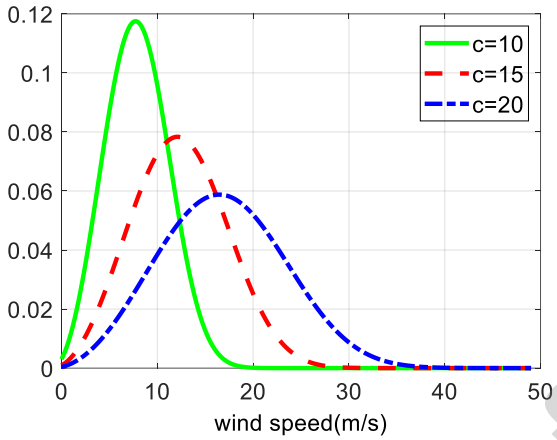


Fig. C2. Weibull PDF with k = 2.

$$\omega = g(v)$$
$$g: R \to R$$
$$f_W(\omega) = f_V(v)\left|\frac{d}{d_\omega} g^{-1}(\omega)\right| \quad \text{(C3)}$$

where v and ω are respectively the wind speed and power random variables, $g$is the function that maps v to ω.

Given g, the wind speed PDF $f_V(v)$ can be transformed to the wind power PDF $f_W(\omega)$ by (C3). It is worth mentioning that the wind speed PDF $f_V(v)$ can be obtained by historical meteorological data of a specific site, and we assume that it can be used for determining the expected values of wind speed and wind power. The expected wind power is considered as the predicted available power.

Fig. C3 proves the PDF of wind power, which has been normalized to correspond to the given wind speed PDF with a shape factor k of 2 and scale factors c of 10, 15, and 20.

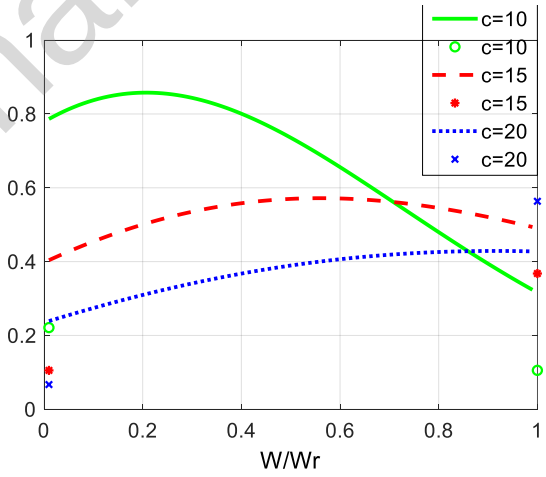


Fig. C3. Wind power PDF with k = 2 (discrete at 0 and 1; continuous between 0 and 1).

Note that the PDF of the wind power output comprises both continuous random and discrete random variables (at 0 and 1).

Appendix C: Acronyms

ABC	Artificial bee colony	AC	Acceleration coefficient
AGSO	Adaptive group search optimization	CI	cognitive influence
DE	Differential evolution	DEEPSO	Differential evolutionary evolution PSO
DEEPSO-OL	Differential evolutionary evolution PSO with orthogonal learning (OL)	ED	Economic dispatch
EPSO	Evolutionary PSO	EV	Electrical vehicle

FA	Factor analysis	GA	Genetic algorithm
MSCA	Modified sine-cosine algorithm	OA	Orthogonal array
OED	Orthogonal experimental design	OL	Orthogonal learning
OPF	Optimal power flow	PDF	Probability density function
PSO	Particle swarm optimization	SCA	Sine-cosine algorithm
SI	social influence	SO	System operator
StdPSO	Standard PSO	WOPF	OPF incorporating wind power

APPENDIX D: DETAILED IMPLEMENTATION OF DEEPSO-OL ON OPF AND WOPF

The overall pseudo code is shown in Algorithm 4. Here authors expand the implementation with certain details hoping to provide good instruction for readers to code by themselves. Meanwhile, we've uploaded all the source code to Github repository: https://github.com/wbai123/matlab-code-of-evolutionary-algorithms-for-optimal-power-flow for reference.

Algorithm: DEEPSO-OL

1. **Initialize** swarm (parameters, topology, population)
2. **Define** a OL threshold *p*
3. **Repeat** iteration *t*
4. **if** rand () > *p*
5. /*use DEEPSO algorithm*/
6. **for** *i* = 1 to swarm size *n*
7. $V_i^{new} = w_{i0}^* V_i + w_{i1}^*(X_{r1} - X_i) + w_{i2}^* \boldsymbol{P}(b_g^* - X_i)$
8. $X_i^{new} = X_i + V_i^{new}$
9. Mutate weights (w_{i0}^*,w_{i1}^*,w_{i2}^*)
10. /*replicate *velocity* and *position* copies using new *w**/
11. $V_{Ci}^{new} = w_{ci0}^* V_{Ci} + w_{ci1}^*(X_{Cr1} - X_{Ci}) + w_{ci2}^* \boldsymbol{P}(b_g^* - X_{Ci})$
12. $X_{Ci}^{new} = X_{Ci} + V_{Ci}^{new}$
13. **end for**
14. Enforce X_i^{new},V_i^{new},X_{Ci}^{new},V_{Ci}^{new} within feasible limits
15. **for** *i* = 1 to swarm size *n*
16. Compute $f(X_{Ci}^{new})$ and $f(X_i^{new})$ /*evaluate solutions*/
17. /*create new solution to replace the current one*/
18. **if** $f(X_{Ci}^{new}) < f(X_i^{new})$
19. $f(X_i^{new}) = f(X_{Ci}^{new})$, $X_i^{new} = X_{Ci}^{new}$, $V_i^{new} = V_{Ci}^{new}$, $w_{i0}^* = w_{ci0}^*$,$w_{i1}^* = w_{ci1}^*$,$w_{i2}^* = w_{ci2}^*$
20. **end if**
21. **end for**
22. **else**
23. /*construct OL based on DEEPSO*/
24. Construct candidate solution by *OL* by Algorithm 3
25. **end if**
26. **until** termination criterion is met
27. **return** *global best*

For OPF, the control variables $X_i = [P_{G,2} \cdots P_{G,i}, V_{G,1} \cdots V_{G,j}, T_1 \cdots T_i, Q_{C,1} \cdots Q_{C,i}]$ is described from (24). The overall structure can be divided into two parts. Line 4 – 22, is the DEEPSO, and line 23 – 25 is the DEEPSO-OL. Only one part is executed at each iteration based on the pre-defined OL probability *p*. Mutate weights in line 9 is based on (5). Line 16 is to compute the fitness values of X_{Ci}^{new}(computed in line 12) and X_i^{new}(computed in line 8). The equation to evaluate solutions is based on (31). For WOPF, the main difference is that the control variables become $X_i = [\omega_1 \cdots \omega_i, P_{G,2} \cdots P_{G,i}, V_{G,1} \cdots V_{G,j}, T_1 \cdots T_i, Q_{C,1} \cdots Q_{C,i}]$ and the computation of fitness value is based on (32).

Declaration of interests

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

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Highlights

1) Propose a novel differential evolutionary PSO variant method based on orthogonal learning to balance exploration and exploitation.
2) Apply to a real-world non-linear optimization OPF problem.
3) Develop a wind energy conversion system model WOPF for wind integrated optimal power flow.