Reinforcement Learning-Based Differential Evolution for Solving Economic Dispatch Problems

Thammarsat Visutarrom[1], Tsung-Che Chiang[1], Abdullah Konak[2], Sadan Kulturel-Konak[3]
[1]Department of Computer Science and Information Engineering, National Taiwan Normal University, Taipei, Taiwan
[2]Information Sciences and Technology, Penn State Berks, Reading, PA 19610, USA
[3]Management Information Systems, Penn State Berks, Reading, PA 19610, USA
thammarsat@gmail.com, tcchiang@ieee.org, auk3@psu.edu, sadan@psu.edu

<< This paper is included in the Proceedings of IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Singapore (virtual conference due to COVID-19), Dec. 14-17, 2020.. >>

Abstract – **In power systems, economic dispatch (ED) deals with the power allocation of power generation nits to meet the power demand and minimize the cost. Many metaheuristics have been proposed to solve the ED problem with promising results. However, the performance of these algorithms might be sensitive to their parameter settings, and parameter tuning requires considerable effort. In this paper, a reinforcement learning (RL)-based differential evolution (DE) is proposed to solve the ED problem. We develop an RL mechanism to adaptively set two critical parameters, crossover rate (*CR*) and scaling factor (*F*), of DE. The performance of the proposed RLDE is compared with the canonical DE and several algorithms in the literature using three test systems. Our algorithm shows good solution quality and strong robustness.**

Keywords - **Economic dispatch, reinforcement learning, differential evolution.**

I. INTRODUCTION

The economic dispatch (ED) problem is a constrained continuous optimization problem, which is essential for the efficient operation of power systems. In the ED problem, a power system consists of N generating units. The problem aims at allocating power output P_j of each unit j ($j = 1, \dots, N$) such that a load demand P_D is satisfied at the minimum total fuel cost under some operational constraints. In the literature [1]-[3], the objective of the problem is expressed in two different forms. Some test cases define the objective function by a convex quadratic function as given in (1), where $a_j, b_j,$ and c_j are the cost coefficients of unit j. Other test cases use a more accurate objective function that includes a sinusoidal term representing the valve-point effect as given in (2), where d_j and e_j are the coefficients of the valve-point effect. In this paper, both nonconvex and convex ED test cases are solved. The problem includes two types of constraints: operating boundary and power balance constraints. Operating boundary constraints limit the power output of each unit j to be between a lower bound P_j^{min} and an upper bound P_j^{max}, as given in (3). The power balance constraint requires that the total power output of the system is equal to the load demand P_D, as stated in (4).

$$F_{cost} = \sum_{j=1}^{N} \left(a_j + b_j P_j + c_j P_j^2\right) \tag{1}$$

$$F_{cost} = \sum_{j=1}^{N} \left(a_j + b_j P_j + c_j P_j^2\right) + |d_j\{\sin(e_j(P_j^{min} - P_j))\}| \tag{2}$$

subject to:

$$P_j^{min} \le P_j \le P_j^{max} \qquad j = 1, \dots, N \tag{3}$$

$$\sum_{j=1}^{N} P_j - P_D = 0 \tag{4}$$

Increasing attention to the ED problem has led to applications of numerous metaheuristic approaches to the problem. Parameter tuning is required for metaheuristics to generate high-quality solutions. Hybrid metaheuristics might improve performance, but they usually have more parameters. The mentioned aspect motivates us to propose a reinforcement learning-based differential evolution (RLDE). The influential parameters of DE, including crossover rate (*CR*) and scaling factor (*F*), are adaptively controlled by reinforcement learning. The rest of the paper is organized as follows. The related studies are reviewed in Section II. The details of the proposed algorithm are described in Section III. Experimental results and discussions are presented in Section IV. Conclusions and future work are lastly given in Section V.

II. LITERATURE REVIEW

In this section, the ED studies using metaheuristics, especially those with adaptively controlled parameters, were reviewed. The particle swarm optimization (PSO) was shown as an effective algorithm in several papers. In Gaing's work [1], PSO was utilized for solving small- and medium-size test cases. The inertia weight factor, controlling the inertial speed of a particle in PSO, was linearly reduced according to the number of iterations. Cai *et al*. [2] presented the chaotic particle swarm optimization (CPSO) for the ED problem. They controlled the inertia weight factor by the ratio of a particle's fitness to the average fitness of the population. This approach slightly performed better than Gaing's PSO [1] in small-size test cases, but it could not find the best-known solutions. Gholamghasemi *et al*. [3] introduced the phasor particle swarm optimization (PPSO) for solving the problem. They removed the inertia weight factor and controlled the acceleration coefficients and the step size of a particle by the cosine and sine functions, respectively. Their algorithm outperformed all of the benchmark algorithms in all test cases used in their study.

Differential evolution (DE) [4] is another promising algorithm for solving the ED problem. Coelho and Mariani [5] modified DE for handling small-size test cases. The parameter F was controlled by the sine function. Infeasible solutions were fixed by a penalty mechanism. The algorithm performed better than the CPSO [2] even using a smaller population size. Zou *et al*. [6] proposed the improved DE (IDE) algorithm, in which they modified the reproduction part of DE. Two mutation operators were randomly applied to generate new solutions. As for parameters, F was randomly generated, and the parameter CR was linearly reduced following the iteration numbers. The algorithm provided impressive results in small- and medium-size test cases.

The mutation operators of DE were introduced in other algorithms to enhance performance. Wang and Li [7] introduced a differential harmony search algorithm (DHS). The pitch adjustment was based on the DE/rand/1 mutation operator but not the bandwidth in harmony search (HS). This modification led the algorithm to perform better than some other algorithms such as GA and PSO [1]. Bhattacharya and Chattopadhyay [8] hybridized the mechanism of DE and biogeography-based optimization (BBO) to generate new solutions. The gap between the best-found solutions and the benchmark solutions was small. Xiong *et al*. [9] proposed multi-strategy ensemble biogeography-based optimization (MsEBBO) for handling the ED problem. The migration rate and emigration rate were controlled by a cosine function, which showed strong performance in some applications [10]. The DE/rand/1 mutation and crossover operators were carried out after the migration process to improve search performance. The algorithm's solution quality is quite similar to that of the IDE [6] and PPSO [3] in solving 38- and 40-unit test cases, respectively.

Moradi-Dalvand *et al*. [11] introduced the continuous quick group search optimizer (CQGSO). This algorithm found the best-known solutions in small- and medium-size test cases. However, the average-case solution quality was still slightly worse than that of PPSO [3]. Betar *et al*. [12] proposed the tournament-based harmony search algorithm (THS). The best-found solution of this algorithm had a small gap to the benchmark, but the average-case solution quality was worse than PPSO [3] and IDE [6]. Adarsh *et al*. [13] solved the ED problem by using the bat algorithm (BA). The loudness parameter and the influential parameter of BA were chaotically updated by a sine function instead of using the linear equation. This algorithm showed impressive results in test cases with different sizes.

Reinforcement learning (RL) is a technique in the field of machine learning. It is usually applied to help an agent to take proper actions in the environment based on its experience to maximize the reward. The idea of RL fits the task of parameter control for DE (choosing proper parameter values during the evolutionary process to generate high-quality solutions) but was not extensively studied in the literature. Thus, we developed an RLDE and investigated its potential in this study.

III. PROPOSED ALGORITHM (RLDE)

The proposed RLDE is introduced in this section. The pseudocode of the overall algorithm is given in Table I. The initialization and constraint handling mechanisms are explained in Subsections A and B, respectively. Reproduction and selection are presented in Subsection C. Subsection D details the proposed RL mechanism and how RL adaptively sets DE parameters F and CR.

A. Solution Encoding and Initialization

A vector $X_i = [P_{i1}, P_{i2}, \dots, P_{ij}, \dots, P_{iN}]$ represents the i^{th} solution in the population, where P_{ij} denotes the output of power generating unit j. Values of P_{ij} are initialized by random values in $[P_j^{min}, P_j^{max}]$. Infeasible solutions are fixed by our repair strategy described in the next subsection. The initial solutions are sorted in the ascending order of the fitness value.

TABLE I: THE OVERVIEW OF RLDE ALGORITHM

Notations
G_{max}, G: maximum generation and current generation number
Pop: population
NP: population size
U: trial vector
V: mutant vector
F, CR: list of scaling factor and crossover rate parameter
$QTable$: Q table of reinforcement learning
s: state of a solution

Initialize(Pop)
Repair(Pop)
Evaluate(Pop)
Sort(Pop)
Assign state s to each solution
Select a random action (F_i, CR_i) for each solution *Pop*[i]
G = 1
while $G \leq G_{max}$ **do**
    for i = 1 **to** NP **do**
        $X_i = Pop[i]$
        V_i = **Mutate**(Pop, F_i)
        U_i = **Crossover**(X_i, V_i, CR_i)
        U_i = **Repair**(U_i)
        Evaluate(U_i)
        $Pop[i]$ = **Select**(X_i, U_i)
    end
    Sort(Pop)
    Update $QTable$ by reinforcement learning
    Select an action (F_i, CR_i) for each solution *Pop*[i] by the ε-greedy method
    G= G+1
end
Output the best solution in *Pop*

B. Constraint Handling

Infeasible solutions may be created during reproduction. Infeasibility due to operating boundary constraints (3) can be easily fixed by setting P_{ij} values to the closest boundary. However, an effective and efficient constraint handling approach is required to keep the power balance constraint (4) satisfied. As handling the equality constraint (4) consumes an enormous computation time, it is relaxed to be an inequality constraint. A solution will be accepted as a feasible one when the difference $Diff = \left|\sum_{j=1}^{N} P_{ij} - P_D\right|$ is less than 10^{-10}. Two repair mechanisms, single-unit repair (SR) and all-unit repair (AR), are used. SR or AR is selected randomly to repair each infeasible solution. If *Diff* of the new solution is not less than 10^{-10} within 30 trials, the repair procedure stops and the trial solution U_i will not survive to the next iteration.

All-unit repair (AR). The average difference is calculated by $AVG_{diff} = (P_D - \sum_{j=1}^{N} P_{ij})/N$. All power outputs P_{ij} are added by AVG_{diff}. If the new value of P_{ij} violates the constraint (3), it is set to the closest boundary.

Single-unit repair (SR): The difference between the load demand and total power output is calculated by $Diff = P_D - \sum_{j=1}^{N} P_{ij}$. In an infeasible solution, any variable P_{ij} that still satisfies the boundary constraint (3) after adding *Diff* is included in a set *S*. If *S* is not empty, a P_{ij} from *S* is randomly selected; otherwise, a variable P_{ij} of the solution is randomly selected. The selected P_{ij} is added by *Diff*. If the new value of P_{ij} violates the constraint (3), it is set to the closest boundary.

C. Reproduction and Solution Selection

The DE/rand/1 mutation and the binomial crossover are employed, as shown in (5) and (6). The new solution (trial vector) replaces the old solution (target vector) if the new one is feasible and has better fitness.

$$V_i = X_{r1} + F_i(X_{r2} - X_{r3}) \tag{5}$$

$$U_i: u_{ij} = \begin{cases} v_{ij} \; if \; rand\,[0,1) \leq CR_i \; or \; j = j_{rand} \\ P_{ij} \qquad\qquad\qquad otherwise \end{cases} \tag{6}$$

D. Parameter Control based on Reinforcement Learning

RL has been receiving increasing attention in the field of optimization due to its effectiveness [14-16]. It is categorized in the area of machine learning for multi-stage decision making, which aims to find the optimal actions to reach the goal of the problem. Three components are essential in RL: states, actions, and rewards. States and actions represent the situations and activities that can happen in the problem. The reward is the learning score obtained after taking an action in a state. In this paper, we use the RL approach to set the values of *CR* and *F* in the DE adaptively. Each component is detailed in the following.

State: At the end of each generation, the solutions are sorted by the fitness value, and the population is divided into four quartiles. The state s of each solution is defined as the quartile where the solution resides, s_1 being the best 25% of the solutions, s_2 the second-best 25%, and so on. In case that the population cannot be equally divided, the number of solutions *NP*/4 in each state is rounded down. Then, the number of solutions in state s_1 is increased by one, and this step is continued for other states sequentially until the total number of solutions in all states equals the population size. For example, if the population size is ten (10/4=2.5), s_1 and s_2 include three solutions, and s_3 and s_4 include two as shown in Fig. 1 (❶).

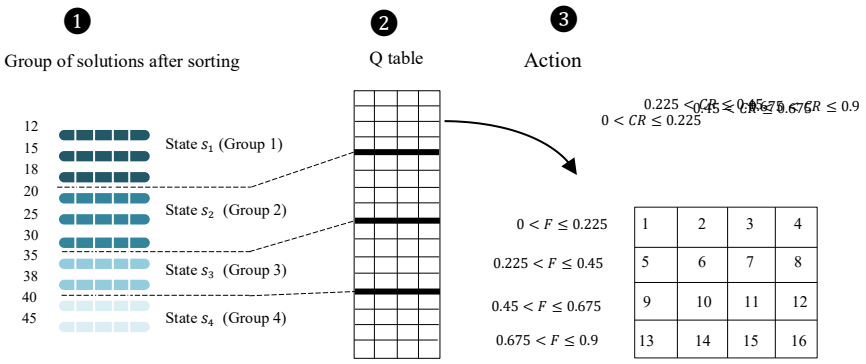

Fig. 1. Illustration of the proposed RL mechanism.

Q-table: The Q-table consists of four states, and each state has 16 actions as shown in Fig.1 (❷). The value of $Q(s,a)$ represents the favorability of selecting action a when a solution is in state s. Q-learning is applied to update the Q-table, as given in (7), where $R(s,a)$ is the immediate reward of taking action a in the current state and $Q(s,a)$ is the accumulated reward of taking action a in state s. The $max_a Q(s',a)$ is the maximum reward of the new state s' over all actions. Updating Q-value is controlled by two parameters: the learning rate α and the discount factor γ.

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha[R(s,a) + \gamma \cdot max_a Q(s',a)] \qquad (7)$$

Action: In our algorithm, an action refers to choosing the sub-ranges of *F* and *CR* for each solution. The ranges of *F* and *CR* are (0, 0.9] and are divided into four equal sub-ranges as shown in Fig. 1 (❸). There are 4×4=16 candidate actions in each state. At the first generation, each solution selects the action randomly. After that, solutions will select the action by the ε-greedy mechanism. Specifically, the solution will take the action with the maximum reward of its state in probability ε; otherwise, it will randomly select an action. After a solution X_i takes an action, the actual values of F_i and CR_i will be uniformly generated in their sub-ranges, and the mutant vector V_i and trial vector U_i are produced accordingly.

Reward: The reward of an action is defined by Table II. Let s and s' denote the states of the solution before and after selection. If the fitness of U_i is worse than that of X_i, the action receives a negative reward (line 2). We use Fig. 1 as an example, where $Numstates$ is 4. When a solution X_i at state 1 is not improved by a trial vector U_i, the reward is (1 – 4 – 1) = -4. If the fitness of U_i is better than that of X_i, the reward depends on the state change. If the state of U_i is better than that of X_i, two conditions are considered following lines 5 to 9. First, if the fitness of U_i is smaller than the minimum fitness value of the previous generation (Min), the action receives a positive reward (line 6). Second, if the fitness of U_i is worse than Min, the action receives a positive reward from the similar equation from the first condition but controlled by the linear equation of the ratio of current generation number G to maximum generation G_{max}. The reason that we add the second condition in our algorithm is to select the most appropriate action to increase the speed of convergence rate of our population in the beginning. Therefore, the action that cannot improve the quality of solutions better than the minimum fitness value of the previous generation will get a lower reward. On the other hand, towards the end of the search, when it might be hard to find the right action, the reward of the second condition will be increased similar to the first condition to help the algorithm to avoid getting stuck at the local optima. Lastly, if the state of U_i is not better than the state of X_i, the action receives a negative reward (line 11).

TABLE II: REWARD CALCULATION

Notations
$NumStates$: number of states
s, s': states of the solution before and after section
G_{max}, G: maximum generation and generation number
Min: minimum fitness value in each generation
X_i, U_i: target and trial vector

if $f(U_i) > f(X_i)$ **then**
    $R(s,a) = s - NumStates - 1$
else
    if $s' \leq s$ **then**
        if $Min > f(U_i)$
            $R(s,a) = (Numstates - s') + 1$
        else
            $R(s,a) = \big((Numstates - s') + 1\big) * (G/G_{max})$
        end
    else
        $R(s,a) = s - s'$
    end
end

IV. EXPERIMENTS AND RESULTS

A. Test Cases and Parameter Setting

Three ED test cases were chosen to investigate the performance of the proposed RLDE, including 13-, 38-, and 40-unit test cases. Details of these test cases can be found in [6] and [13]. Two load demands, 1800 MW and 2520 MW, were tested for the 13-unit test case, but here we only showed the results of one of them since the results are similar. The load demands are 6000 MW and 10500 MW in 38- and 40-unit cases, respectively. The 13- and 40-unit test cases are non-convex ED problems (i.e., using (2) as the objective function), and the 38-unit test case is a convex ED problem (i.e., using (1) as the objective function).

The RLDE was run with the population sizes 30, 30, and 50 and with the maximum number of generations 550, 700, 1000 for 13- 38-, and 40-unit test cases, respectively. For the proposed RL mechanism, the learning rate α, discount factor γ, and ε probability were set to 0.2 0.6, and 0.7, respectively, as it achieves the best solution quality in all test cases. We ran the RLDE 50 times to solve each test case.

B. Analysis of Solution Quality

Tables III-V summarize the results obtained by our RLDE, canonical DE with random parameters (Random DE), canonical DE with the most effective parameter setting of 81 combinations of values of *CR* and *F* (DE81), and several algorithms in the literature. The second, third, and fourth columns present the minimum, average, and maximum cost obtained by each algorithm. Our RLDE found the best-known solutions for the 13-unit cases and the 40-unit case. For the 38-units case, the deviation percentage from the best-known solution was lower than 0.01%. The RLDE performed better than Random DE in all cases, which shows that RLDE can select parameter values appropriately. The RLDE outperformed DE81 for seven out of nine metrics. Although DE81 had lower average and maximum costs than RLDE in the 40-unit case, it required a lot of effort on parameter tuning.

TABLE III: RESULTS OF 13-UNIT SYSTEM (1800 MW)

Algo.	Min	Avg.	Max	Ref.
IDE	17960.37	17961.47	17969.49	[6]
DHS	17960.37	17968.36	17969.57	[7]
RLDE	17960.37	17969.1	17970.89	
THS	17960.37	17977.6	-	[12]
SADE	17960.41	17966.35	17969.39	[6]
DE81(0.5, 0.5)	17975.46	17978.26	17980.59	
Random DE	17991.88	18014.67	18147.35	
MBDE	18024.88	18172.98	18452.96	[6]

TABLE IV: RESULTS OF 38-UNIT SYSTEM (6000 MW)

Algo.	Min	Avg.	Max	Ref.
MsEBBO	9417235.78	9417235.78	9417235.78	[9]
DE/BBO	9417235.78	-	-	[8]
IDE	9417235.79	9417235.79	9417235.79	[6]
MBDE	9417235.79	9417235.79	9417235.79	[6]
RLDE	9417235.79	9417235.83	9417235.9	
DE81(0.5, 0.8)	9417235.79	9417245	9417273	
Random DE	9417241.39	9417368.59	9417396.96	
SADE	9417241.93	9417252.85	9417274.92	[6]

TABLE VI: RESULTS OF 40-UNIT SYSTEM (10500 MW)

Algo.	Min	Avg.	Max	Ref.
MsEBBO	121412.53	121417.19	121450	[9]
RLDE	121412.53	121441.76	121506.69	
PPSO	121412.54	121412.59	121413.95	[3]
CBA	121412.55	121418.98	121436.15	[13]
CQGSO	121412.55	121423.52	121438.69	[11]
DE81(0.1, 0.1)	121415.07	121432.76	121473.98	
THS	121425.15	121528.65	-	[12]
Random DE	121429.62	121566.88	121823.58	

C. Analysis of Algorithm Robustness

This research aims to propose an algorithm that is able to find good solutions with little effort on parameter tuning. We proposed the RL mechanism to adaptively control the values of F and CR of DE. Since our RL introduced some hyper-parameters, we compared the canonical DE and the proposed RLDE in terms of their performance sensitivity to parameter setting. The canonical DE used fixed values of F and CR through the whole evolutionary process. We tested nine values $\{0.1, 0.2, 0.3, \ldots, 0.9\}$ for F and CR of the canonical DE and for α and γ of the RLDE. The population size and the number of maximum generations were the same for both algorithms. Fig. 2 shows the average-case solution quality of 9×9=81 parameter settings for both algorithms by heat maps. The darker the color is, the better the solution quality is. Only a small portion of parameter settings can lead the canonical DE to find high-quality solutions, and different parameter values are required for solving different problems. On the contrary, the RLDE can find good solutions under most parameter settings. These results show the strong robustness of the RLDE with respect to parameter settings.

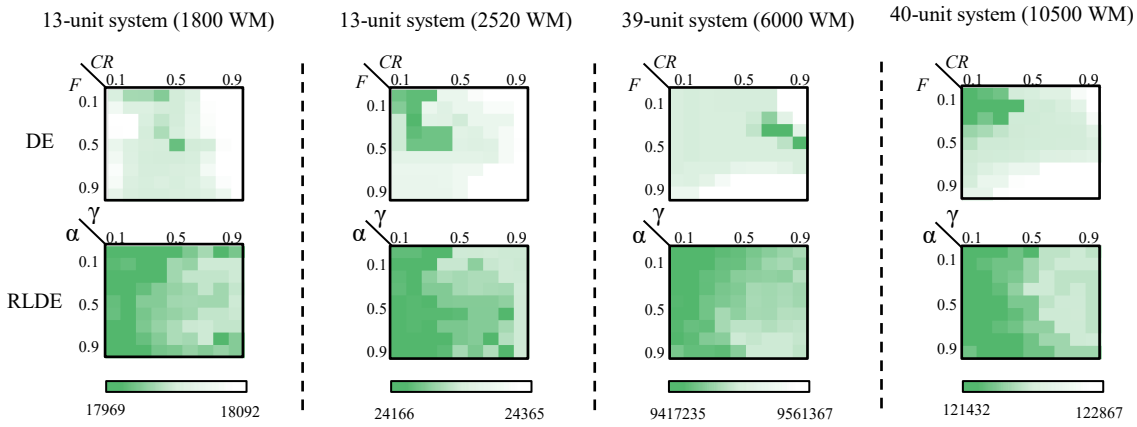

Fig. 2. The average performance of DE and RLDE using different parameter settings.

V. CONCLUSIONS AND FUTURE WORK

Economic dispatch is an important optimization problem in the power industry. Although many metaheuristics algorithms were able to find high-quality solutions, their performance could be sensitive to parameter settings, and parameter tuning could be a time-consuming process. In this paper, we proposed the RLDE, in which we utilized the RL technique to adaptively control the critical parameters of DE. We tested the RLDE by three test cases and compared its performance with many existing algorithms. Our RLDE showed competitive results. Moreover, we tested the sensitivity of RLDE to the parameter setting and compared the sensitivity with the canonical DE. Unlike the canonical DE, our RLDE can find good solutions under a wide range of parameter settings, hence increasing the robustness of DE.

We will continue this research in several directions for future work. First, we will investigate the reward mechanism for a better selection of actions. Second, we will improve the RLDE to solve large-size test cases such as 110- and 140-unit test systems. Third, we will apply the RLDE to other continuous optimization problems.

REFERENCES

[1] Z. L. Gaing, "Particle swarm optimization to solving the economic dispatch considering the generator constraints," *IEEE Transactions on Power Systems*, vol. 18, pp. 1187-1195, 2003.

[2] J. Cai, X. Ma, L. Li, and H. Peng, "Chaotic particle swarm optimization for economic dispatch considering the generator constraints," *Energy Conversion and Management*, vol. 48, pp. 645-653, 2007.

[3] M. Gholamghasemi, E. Akbari, M. B. Asadpoor, and M. Ghasemi, "A new solution to the non-convex economic load dispatch problems using phasor particle swarm optimization," *Applied Soft Computing*, vol. 79, pp. 111-124, 2019.

[4] 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.

[5] L. D. S. Coelho and V. C. Mariani, "Improved differential evolution algorithms for handling economic dispatch optimization with generator constraints," *Energy Conversion and Management*, vol. 48, pp. 1631-1639, 2007.

[6] D. Zou, S. Li, G. G. Wang, Z. Li, and H. Ouyang, "An improved differential evolution algorithm for the economic load dispatch problems with or without valve-point effects," *Applied Energy*, vol. 181, pp. 375-390, 2016.

[7] L. Wang and L. P. Li, "An effective differential harmony search algorithm for the solving non-convex," *Electrical Power and Energy Systems*, vol. 44, pp. 832-843, 2013.

[8] A. Bhattacharya and P. K. Chattopadhyay, "Hybrid differential evolution with biogeography-based optimization for solution of economic load dispatch," *IEEE Transactions on Power Systems*, vol. 25, pp. 1955-1964, 2010.

[9] G. Xiong, D. Shi, and X. Duan, "Multi-strategy ensemble biogeography-based optimization for economic dispatch problems," *Applied Energy*, vol. 111, pp. 801-811, 2013.

[10] H. Ma, "An analysis of the equilibrium of migration models for biogeography-based optimization," *Information Sciences*, vol. 180, pp. 3444-3464, 2010.

[11] M. Moradi-Dalvand, B. Mohammadi-Ivatloo, A. Najafi, and A. Rabiee, "Continuous quick group search optimizer for solving non-convex economic dispatch problems," *Electric Power Systems Research*, vol. 93, pp. 93-105, 2012.

[12] M. A. A. Betar, M. A. Awadallah, A. T. Khader, and A. L. Bolaji, "Tournament-based harmony search algorithm for non-convex economic load dispatch problem," *Applied Soft Computing*, vol. 47, pp. 449–459, 2016.

[13] B. R. Adarsh, T. Raghunathan, T. Jayabarathi, and X. S. Yang, "Economic dispatch using chaotic bat algorithm," *Energy*, vol. 96, pp. 666-675, 2016.

[14] Y. Z. Hsieh and M. C. Su, "A Q-learning-based swarm optimization algorithm for economic dispatch problem," *Neural Computing & Applications*, vol. 27, pp. 2333-2350, 2016.

[15] Y. Liu, H. Lu, S. Cheng, and Y. Shi, "An adaptive online parameter control algorithm for particle swarm optimization based on reinforcement learning," *in Proceedings of IEEE Congress Evolution Computation*, pp. 815–822, 2019.

[16] M. Sharma, A. Komninos, M. Lopez Ibanez, and D. Kazakov, "Deep reinforcement learning based parameter control in differential evolution," *in Proceedings of Genetic and Evolutionary Computation Conference*, pp. 709–717, 2019.