IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 26, 2019, accepted February 6, 2019, date of publication February 12, 2019, date of current version February 27, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2899043

Jaya Algorithm With Self-Adaptive Multi-Population and Lévy Flights for Solving Economic Load Dispatch Problems

JIANG-TAO YU[1,2], CHANG-HWAN KIM[1], ABDUL WADOOD[1], TAHIR KHURSHAID[1], AND SANG-BONG RHEE[1]

[1]Department of Electrical Engineering, Yeungnam University, Gyeongsan 38541, South Korea
[2]Department of Electronic Information and Electrical Engineering, Anyang Institute of Technology, Anyang 455000, China

Corresponding author: Sang-Bong Rhee (rrsd@yu.ac.kr)

ABSTRACT In this paper, a recently proposed Jaya algorithm is implemented on the economic load dispatch problems (ELDPs). Different from most of the other meta-heuristics, Jaya algorithm needs no algorithm-specific parameters, and only two common parameters are required for effective execution, which makes the implementation simple and effective. Simultaneously, considering the non-convex, non-linear, and non-smooth characteristics of the ELDPs, the multi-population (MP) method is introduced to improve the population diversity. However, the introduction of the MP method adds extra parameters to the Jaya algorithm, hence a self-adaptive strategy is used to cope with the tuning problem for extra parameters. Moreover, to avoid being trapped by local optima, Lévy flights distribution is incorporated into the population iteration phase. Finally, Jaya algorithm with self-adaptive multi-population and Lévy flights (Jaya-SML) is proposed, it is evaluated by ELDPs with different constraints including power balance constraints, generating capacity limits, ramp rate limits, prohibited operating zones, valve-point effects, and multi-fuel options. The comparisons with state-of-the-art methods indicate that Jaya-SML can generate more competitive results for solving the ELDPs.

INDEX TERMS Economic load dispatch problems, Jaya, self-adaptive, multi-population, Lévy flights.

I. INTRODUCTION

Economic load dispatch problems (ELDPs) are regarded as optimization problems with high dimensional, non-convex, non-linear and non-smooth characteristics under various of constraints, which requires powerful optimization technique to handle [1], [2]. The prime requirement of ELDPs are to allocate all the committed generators so as to accomplish the total load demand in the most economical way, while satisfying physical and operational constraints imposed by generators and system limitations. Over the recent decades, great efforts of researchers across the world have been made to solve ELDPs by using meta-heuristic methods such as tabu search (TS) [3], genetic algorithm (GA) [4], particle swarm optimization (PSO) [5], artificial immune system (AIS) [6], harmony search (HS) [7], firefly algorithm [8], biogeography based optimization (BBO) [9], artificial bee colony algorithm (ABC) [10] and teaching-learning-based optimization (TLBO) [11]. Competitive results in terms of fuel cost savings and convergence rate have been achieved. However, since the ELDPs exhibit highly non-convex, non-linear and non-smooth characteristics, they tend to be easily trapped by local optima rather than at the global optimum.

To overcome the drawbacks, hybrid meta-heuristic approaches have been reported to further obtain the global optimum for ELDPs, such as new particle swarm optimization with local random search (NPSO-LRS) [12], chaos mutation firefly algorithm (CMFA) [13], hybrid of fuzzy adaptive particle swarm optimization with variable DE (FAPSO-VDE) [14], DE with chaos sequences and sequential quadratic programming (DEC-SQP) [15], DE with truncated Lévy flight and population diversity measure (DEL) [16], genetic algorithm with pattern search and sequential quadratic programming (GA-PS-SQP) [17], across neighborhood search algorithm with variable reduction strategy (ANS-VRS) [18], multi-population based chaotic JAYA algorithm (MP-CJAYA) [19], posteriori multiobjective self-adaptive multipopulation Jaya algorithm (MO-SAMP Jaya) [20], self-adaptive Jaya algorithm (SJaya) and chaotic-Jaya (CJaya) algorithm [21]. However, even though the

The associate editor coordinating the review of this manuscript and approving it for publication was Sotirios Goudos.

modified approaches have gained better performances, the combination may lead to increased number of algorithm parameters whose turning task is quite time-consuming and disturbing. Hence, we call for new algorithms with parameters as fewer as possible.

Jaya algorithm is a recently proposed method with excellent advantage of reducing the number of parameters. Except for two common parameters named maximum number of iteration $N_{\max}$ and population size N_{pop}, not a single more algorithm-specific parameter is needed [22]. This unique advantage has perfectly satisfied the calls for the number of parameters to be as fewer as possible. However, Jaya algorithm only guarantees the population to keep on getting close to the best position and getting away from the worst position, so the population converge so quickly to optima because of the strong attraction of the best position nearby, which will cause premature. Actually, for solving ELDPs, it is quite necessary to guide the population search in different regions to get local optima as many as possible, since these local optima have larger probability to become the global optima in the next iteration.

Multi-population (MP) method is proved to be a good technique to satisfy the requirements above, which works by dividing the whole population in the entire region into a certain number of sub-populations in different subregions, with the goal of enhancing population diversity and avoiding premature. However, there is a crucial question to consider for MP method, that is how many sub-populations are needed to cover the entire region? To answer this question, a self-adaptive (SA) strategy is employed to MP method, then the number of sub-populations is either decreased or increased automatically according to the strength of the environmental changes, which resolves the problem of the determination for the number of sub-populations. When sub-populations are properly created by SA strategy, they will undergo convergent process to keep exploiting the covered subregions.

In order to speed up the convergent process, as well as to increase the potential of finding global optima, we incorporate Lévy flights into the population updating phase. Finally, Jaya algorithm with self-adaptive multi-population and Lévy flights (Jaya-SML) is proposed in this study. To the best of the authors' knowledge, this is the first time for adaptive Jaya algorithm being implemented in solving the ELDPs.

The rest of this paper is constructed as follows. In Section 2, the formulations of ELDPs are presented. Related works on Jaya, MP method, SA strategy and Lévy flights are described in Section 3. The procedures of solving ELDPs are explained in Section 4. Experimental results and comparisons are provided and analyzed in Section 5. Finally, discussions and conclusion are given in Section 6.

II. PROBLEM FORMULATION

A. OBJECTIVE FUNCTION

Mathematical model for ELDPs is to add up all the fuel costs of the generating units in a power system as expressed below [11], [23]:

$$\min\ F = \sum_{i=1}^{N} F_i(P_i) \tag{1}$$

where N is the total number of committed generators, i is the index of generator where $i \in [1, N]$, P_i is the power output of generator i, $F_i(P_i)$ is the cost function of generator i with power output P_i, F is the total cost of all the generators.

Generally speaking, in classical ELDPs, the cost function of each generating unit is described by quadratic polynomial as:

$$F_i(P_i) = a_i P_i^2 + b_i P_i + c_i \tag{2}$$

where a_i, b_i, c_i is the fuel cost coefficients of generator i.

In practice, the valve-point effects on the costs of generating units must be considered. So the rectified sinusoidal components are added to the classical cost function as follows:

$$F_i(P_i) = a_i P_i^2 + b_i P_i + c_i + |e_i \sin(f_i(P_i^{\min} - P_i))| \tag{3}$$

where e_i, f_i are the fuel cost coefficients of generator i reflecting valve-point effects.

In some cases, the committed generators may be supplied by multiple fuels as natural gas, coal or oil. Then the cost function is defined with piecewise quadratic functions which reflect the effects of the fuel type changes [4]. Considering valve-point effects and multiple fuels, the objective function can be described as:

$$F_i(P_i) = \begin{cases} a_{i1}P_i^2 + b_{i1}P_i + c_{i1} + |e_{i1}\sin(f_{i1}(P_{i1}^{\min} - P_i))|, \\ \text{for fuel 1},\ P_i^{\min} \le P_i \le P_{i,1} \\ a_{i2}P_i^2 + b_{i2}P_i + c_{i2} + |e_{i2}\sin(f_{i2}(P_{i2}^{\min} - P_i))|, \\ \text{for fuel 2},\ P_{i,1} \le P_i \le P_{i,2} \\ \vdots \\ a_{im}P_i^2 + b_{im}P_i + c_{im} + |e_{im}\sin(f_{im}(P_{im}^{\min} - P_i))|, \\ \text{for fuel m},\ P_{i,m-1} \le P_i \le P_i^{\max} \end{cases} \tag{4}$$

where m is the total number of fuel types, a_{iq}, b_{iq}, c_{iq}, e_{iq}, f_{iq} are the fuel cost coefficients of generator i using fuel type q where $q \in [1, m]$.

B. CONSTRAINED FUNCTIONS

1) POWER BALANCE

The total power generated by all the committed generators must equal to the summation of the demanded power P_{demand} and the total transmission power loss P_{loss}, which can be formulated as:

$$\sum_{i=1}^{N} P_i = P_{demand} + P_{loss} \tag{5}$$

where P_{loss} is calculated by Kron's formula:

$$P_{loss} = \sum_{i=1}^{N}\sum_{j=1}^{N} P_i B_{ij} P_j + \sum_{i=1}^{N} B_{i0} P_i + B_{00} \tag{6}$$

where B_{ij}, B_{i0}, B_{00} are the B-matrix coefficients for P_{loss} which can be generally assumed to be constants under a normal operating condition.

2) GENERATING CAPACITY

The power output P_i should be within its maximum and minimum limits, as shown below:

$$P_i^{\min} \le P_i \le P_i^{\max} \tag{7}$$

where $P_i^{\max}$ and $P_i^{\min}$ are the maximum and minimum limits of the i^{th} generator.

3) RAMP RATE LIMIT

Under practical circumstances, the operating range of every generating unit is restricted by its ramp rate limit, so the output power P_i can not be adjusted instantaneously. The up-ramp and down-ramp constraints are as follows:

$$P_i - P_i^0 \le UR_i \text{ and } P_i^0 - P_i \le DR_i \tag{8}$$

where P_i is the present power output, P_i^0 is the previous power output, UR_i and DR_i is the up-ramp and down-ramp limit of generator i respectively. Considering together with the generating capacity limit, ramp rate limit can be modified as:

$$\max(P_i^{\min}, P_i^0 - DR_i) \le P_i \le \min(P_i^{\max}, P_i^0 + UR_i) \tag{9}$$

4) PROHIBITED OPERATING ZONES (POZS)

In practice, since there are physical limitations when operating the generating units, the whole operating zones are not always available. Prohibited operating zones (POZs) lead to discontinuous regions for the objective function. The output power P_i has constraints as follows:

$$P_i \in \begin{cases} P_i^{\min} \le P_i \le P_{i,1}^{lower} \\ P_{i,r-1}^{upper} \le P_i \le P_{i,r}^{lower} \\ P_{i,z_i}^{upper} \le P_i \le P_i^{\max} \end{cases} \tag{10}$$

where z_i is the total number of POZs for generator i, r is the index of POZs where $r \in [1, z_i]$, $P_{i,r}^{lower}$ and $P_{i,r}^{upper}$ are the lower and upper bounds of the r^{th} POZ of the i^{th} generator respectively.

III. RELATED WORK

A. THE STANDARD JAYA ALGORITHM

Jaya algorithm is a newly developed meta-heuristic method for solving constrained and unconstrained optimization problems [22]. Different from other heuristic algorithms requiring for algorithm-specific parameters, Jaya algorithm is free from the algorithm-specific parameters, two and only two common parameters named maximum number of iteration $N_{\max}$ and population size N_{pop} are required, whose values can be initialised without difficulties. This significant improvement makes the application of Jaya algorithm simple and efficient.

Let us assume $F(X)$ is the objective function required to be maximized or minimized, $F(X)_{bs}$ and $F(X)_{ws}$ represent the best value and the worst value of $F(X)$ among all the candidate solutions during each iteration. Suppose the design variable number is N_{var} where the index of design variable $j \in [1, N_{var}]$, suppose the population size is N_{pop} where the index of population $k \in [1, N_{pop}]$, suppose the maximum iteration number is $N_{\max}$ where the index of current iteration $t \in [1, N_{\max}]$. Let $X_{j,k,t}$ be the value of the j^{th} design variable for the k^{th} candidate population during the t^{th} iteration, then the modified value $X'_{j,k,t}$ by Jaya algorithm is calculated by:

$$X'_{j,k,t} = X_{j,k,t} + r_1 \times (X_{j,bs,t} - |X_{j,k,t}|) - r_2 \times (X_{j,ws,t} - |X_{j,k,t}|) \tag{11}$$

where $X_{j,bs,t}$ and $X_{j,ws,t}$ are the values of the j^{th} variable for $F(X)_{bs}$ and $F(X)_{ws}$ during the t^{th} iteration respectively, r_1 and r_2 are two random numbers ranged in $[0, 1]$. The term $r_1 \times (X_{j,bs,t} - |X_{j,k,t}|)$ indicates the tendency of the solution to move closer to the best position and the term $r_2 \times (X_{j,ws,t} - |X_{j,k,t}|)$ indicates the tendency of the solution to avoid the worst position. $F(X)'$ is the modified value of $F(X)$, if $F(X)'$ provides better value than $F(X)$, then $X_{j,k,t}$ is replaced by $X'_{j,k,t}$ and $F(X)$ is replaced by $F(X)'$; otherwise, keep the old value. All the values of new obtained $X_{j,k,t}$ and $F(X)$ are maintained and become the inputs to the next iteration [22]. The pseudo code of Jaya algorithm is shown in Algorithm 1:

Algorithm 1 The Standard Jaya

```
Initialize N_pop, N_var and N_max;
Generate initial population X;
Evaluate the fitness value F(X);
Set t = 1;
while t < N_max do
    Identify X_{j,bs,t} and X_{j,ws,t} according to F(X);
    for k = 1 → N_pop do
        for j = 1 → N_var do
            Generate updated solutions X'_{j,k,t} by (11);
        end
        if F(X'_{j,k,t}) is better than F(X_{j,k,t}) then
            X_{j,k,t} = X'_{j,k,t}
            F(X_{j,k,t}) = F(X'_{j,k,t})
        else
            Keep the old value;
        end
    end
    t = t + 1;
end
```

B. MULTI-POPULATION METHOD

Multi-population (MP) method is implemented with the aim of improving population diversity by scattering the entire

region with the whole population into a certain number of subregions with sub-populations. Each subregion is assigned to either intensifying or diversifying the searching process. There may be one or more local optima covered within each subregion, by keeping searching the subregions separately, changes can be monitored more effectively. Whenever a change in the solution is observed during the iteration, all the sub-populations interact with each other by means of dividing and merging process. So cooperating with MP method is an effective way to improve population diversity and to enhance the searching ability for Jaya algorithm.

As mentioned above, to represent the total number of divided sub-populations, a key parameter is introduced as K, hence the population size of each sub-population N_{sub_pop} is:

$$N_{sub\_pop} = N_{pop}/K \tag{12}$$

where N_{pop} is the population size of initially generated population.

It should be noted that, each individual is grouped to a sub-population by random way, each sub-population is assigned to explore a different region in the fitness area. If (12) has remainders, then the remaining individuals are randomly grouped to one of the sub-populations. Pseudo code of MP method is shown in Algorithm 2:

Algorithm 2 MP

```
Initialize N_pop;
Generate initial population X;
Set K;
while (Maximum number of iterations is not met) do
    Divide population X into K sub-populations as
    X_1,X_2,···,X_{K-1},X_K;
    for q = 1 → K do
        for k = 1 → N_sub_pop do
            Perform algorithm to X_{q,k} to generate X'_{q,k};
            if F(X'_{q,k}) is better than F(X_{q,k}) then
                X_{q,k} = X'_{q,k}
                F(X_{q,k}) = F(X'_{q,k})
            else
                Keep the old value;
            end
        end
    end
    Merge the sub-populations into X;
end
```

C. SELF-ADAPTIVE STRATEGY

For MP method, the disadvantage is that one more parameter K has to be introduced, and the selection of a proper value for K is quite a difficult task since it depends on the complexity of the problem. In order to address this issue, a self-adaptive (SA) strategy which modifies the value of K automatically is applied in this work. By integrating SA strategy with MP method, K value can be self-adaptively determined by the change strength of the solution without manual parameter tuning. Compared with steady-size MP, self-adaptive MP not only monitors the solution changes more effectively, but also maintains the population diversity very well [24]. The steps of self-adaptive MP are illustrated as follows, pseudo code is shown in Algorithm 3.

Algorithm 3 Self-Adaptive MP

```
Initialize N_pop;
Generate initial population X;
Initial K=2;
while (Maximum number of iterations is not met) do
    Divide population X into K sub-populations as
    X_1,X_2,···,X_{K-1},X_K;
    for q = 1 → K do
        for k = 1 → N_sub_pop do
            Perform algorithm to X_{q,k} to generate X'_{q,k};
            if F(X'_{q,k}) is better than F(X_{q,k}) then
                X_{q,k} = X'_{q,k}
                F(X_{q,k}) = F(X'_{q,k})
            else
                Keep the old value;
            end
        end
    end
    Merge the sub-populations into X;
    if F(cr_best) is better than F(fm_best) then
        K = K + 1
    else if F(cr_best) is worse than F(fm_best) then
        K = K − 1
    else
        K = K
    end
end
```

i. Generate initial population X and suppose the fitness function is $F(X)$.
ii. Divide the population X into K sub-populations (initial $K = 2$).
iii. Calculate the fitness value of each sub-population independently by the pre-defined algorithm.
iv. Compare the present fitness value with its former fitness value, if the fitness value gets better, keep the present sub-population; otherwise, keep the former sub-population.
v. Merge all the sub-populations together.
vi. Suppose $F(cr\_best)$ is the current best fitness value and $F(fm\_best)$ is the former best fitness value, if $F(cr\_best)$ is better than $F(fm\_best)$, then K is increased by 1 with the purpose to enhance the exploration ability among the entire region; if $F(cr\_best)$ is worse than $F(fm\_best)$, then K is decreased by 1 as the searching process needs to be more exploitive than explorative; if $F(cr\_best)$ is equal to $F(fm\_best)$, then keep the K value unchanged.

vii. If the maximum number of evaluations has reached, end the loop and report the best value; Otherwise, go back for re-dividing the population.

D. LÉVY FLIGHTS

The generation of random numbers using Lévy flights consists of two steps: the choice of a random direction and the generation of *step* which obeys the chosen Lévy distribution. Lévy distribution is a simple power-law formula $L(s) \sim |s|^{-1-\beta}$ where $0 < \beta < 2$ is an index [25]. If the value of β is small, it allows the variable perform long-distance jumps in the search space and avoids being trapped in local optima; if the value of β is big, it continues to derive new values around the variable. As a result, by employing Lévy flights on updating the population, variables are able to take short jumps together with occasionally long-distance jumps towards its best value, thereby enhancing the population diversity and facilitating the algorithm to perform stronger global exploration throughout the search space.

In this study, we apply Lévy flights to each variable of the current iteration by the following equation:

$$X_{j,k,t}^{levy} = Levy(X_{j,k,t}) + r_1 \times (X_{j,bs,t} - |X_{j,k,t}|) - r_2 \times (X_{j,ws,t} - |X_{j,k,t}|) \quad (13)$$

where

$$Levy(X_{j,k,t}) = X_{j,k,t} + stepsize \times rand(size(X_{j,k,t})) \quad (14)$$

where

$$stepsize = 0.01 \times step \times (X_{j,k,t} - X_{j,bs,t}) \quad (15)$$

here *rand* is randomly generated numbers ranged in [0, 1]; the factor 0.01 comes from the fact that *step*/100 should be the typical step size of walks where *step* is the typical length scale; otherwise, Lévy flights may become so aggressive that new solutions jump outside of the domain and thus waste evaluations. $X_{j,k,t}$ and $X_{j,bs,t}$ are variables from (11).

For random walk, the value of *step* can be calculated by Mantegna's algorithm as:

$$step = \frac{u}{|v|^{1/\beta}} \quad (16)$$

here it should be noted that β parameter takes major role in Lévy distributions, by setting different values for β, the distribution situation is changed accordingly. In this study, we choose constant value 1.5 for β. The other two parameters u and v are drawn from normal distributions with standard deviation σ_u and σ_v given by:

$$u \sim N(0, \sigma_u^2), \quad v \sim N(0, \sigma_v^2) \quad (17)$$

where

$$\sigma_u = \left[\frac{\Gamma(1+\beta) \times \sin(\frac{\pi\beta}{2})}{\Gamma(\frac{1+\beta}{2}) \times \beta \times 2^{(\beta-1)/2}} \right]^{1/\beta} \quad (18)$$

$$\sigma_v = 1 \quad (19)$$

where $\Gamma(\cdot)$ is standard Gamma function [26].

Pseudo code for Lévy flights is shown in Algorithm 4.

Algorithm 4 Lévy Flights With Jaya

```
Initialize N_pop, N_var and N_max;
Generate initial population X;
Evaluate the fitness value F(X);
Set t = 1;
while t < N_max do
    Identify X_{j,bs,t} and X_{j,ws,t} according to F(X);
    for k = 1 → N_pop do
        for j = 1 → N_var do
            Generate modified solutions X^{levy}_{j,k,t} by (13);
        end
        if F(X^{levy}_{j,k,t}) is better than F(X_{j,k,t}) then
            X_{j,k,t} = X^{levy}_{j,k,t}
            F(X_{j,k,t}) = F(X^{levy}_{j,k,t})
        else
            Keep the old value;
        end
    end
    t = t + 1;
end
```

E. CONSTRAINTS HANDLING STRATEGY

1) HANDLING OF CONSTRAINED EQUATIONS

The constraints handling strategy is one of the significant concerns in solving ELDPs. Among all the techniques, penalized fuel cost function is the most commonly used one. By adding certain values to the objective function based on the constraint violations, the constrained problems are transformed into unconstrained ones. In this paper, the constraints of power balance limit and POZs are handled by adding penalty factors to the objective function as follows:

$$F_p = \sum_{i=1}^{N} F_i(P_i) + \lambda_{pb} \times (\sum_{i=1}^{N} P_i - P_{demand} - P_{loss}) + \lambda_{poz} \times \sum_{i=1}^{N} Q_i \quad (20)$$

where F_p is the value of the penalized objective function; Q_i is an indicator of falling into POZs; λ_{pb} and λ_{poz} are penalty factors used to penalize the fuel cost proportional to the amount of constraint violations. Notes that λ_{poz} equals to zero if the probability of falling into POZs is not considered [12].

2) HANDLING OF UPPER AND LOWER LIMITS

New solutions generated via supposed algorithms may violate the maximum or minimum limits, so they need to be redefined to satisfy the limits. For power output limit and ramp rate limit constraints, we adopt the following strategy to handle:

$$P_i = \begin{cases} \max(P_i^{\min}, P_i^0 - DR_i), & if\ P_i \le \max(P_i^{\min}, P_i^0 - DR_i) \\ \min(P_i^{\max}, P_i^0 + UR_i), & if\ P_i \ge \min(P_i^{\max}, P_i^0 + UR_i) \\ P_i, & \text{otherwise} \end{cases} \quad (21)$$

where $P_i^{\min}$, $P_i^{\max}$, P_i^0, DR_i and UR_i have already been illustrated before.

3) HANDLING OF POZS VIOLATION

As mentioned above, because of the exists of POZs, there are upper and lower limits for power output of the generator. If the obtained power output falls into POZs, it needs to be recalculated to satisfy the limits. For handling the POZs violation, a ''middle point'' concept is defined as follows:

$$P_{i,r}^{middle} = \frac{P_{i,r}^{lower} + P_{i,r}^{upper}}{2} \tag{22}$$

Therefore, there are two sub-POZs divided from ''middle point'' including left and right ones, then the new value of power output is re-determined as below [27]:

$$P_i^{new} = \begin{cases} P_{i,r}^{lower}, & if\ P_i \le P_{i,r}^{middle} \\ P_{i,r}^{upper}, & if\ P_i > P_{i,r}^{middle} \end{cases} \tag{23}$$

where $P_{i,r}^{lower}$ and $P_{i,r}^{upper}$ are the lower and upper bounds of the r^{th} POZ of the i^{th} generator respectively.

IV. IMPLEMENTATION OF JAYA-SML FOR ELDPS

According to the related work in previous section, Jaya algorithm with self-adaptive multi-population and Levy flights (Jaya-SML) is proposed. In Jaya-SML, three modifications are added to the standard Jaya, they are multi-population (MP) method, self-adaptive (SA) strategy and Lévy flights distribution. Pseudo code of the proposed Jaya-SML is shown in Algorithm 5.

It starts by initializing the values for common parameters. Then the initial population is created and evaluated. Next, the whole population is divided into K sub-populations. After that, each sub-population utilises Jaya algorithm with Lévy flights. If there is a change in the solution, the algorithm compares the change strength to update the K value. If the stopping condition (we set this as maximum number of iterations) has been reached, the algorithm terminates and the best solution is returned. Otherwise, the algorithm merges all the sub-populations and re-divides the whole population into K sub-populations, then starts a new iteration. The main steps are described with further details below:

i. Set parameters. Four common parameters are initialized as population size N_{pop}, maximum iteration number $N_{\max}$, number of design variables N_{var} and number of sub-populations K (initial $K = 2$).

ii. Initialization. Initial population X are generated as follows:

$$X_{j,k} = X_j^{\min} + (X_j^{\max} - X_j^{\min}) \times rand(N_{pop}, N_{var}) \tag{24}$$

here $X_{j,k}$ is the j^{th} generator in the k^{th} candidate solution where $j \in [1, N_{var}]$ and $k \in [1, N_{pop}]$, $X_j^{\min}$ and $X_j^{\max}$ are the lower and upper limits of the j^{th} generator given by (7).

Algorithm 5 Jaya-SML

```
Initialize N_pop, N_var, N_max and K;
Generate initial population X by (24);
Calculate the fuel cost F(X);
Set t = 1;
while t < N_max do
    Divide population X into K sub-populations as
    X_1,X_2,···,X_{K-1},X_K;
    for q = 1 → K do
        Confirm X_{q,best,t} and X_{q,worst,t} within X_{q,t}
        for k = 1 → N_sub_pop do
            for j = 1 → N_var do
                Generate modified solutions X^{levy}_{q,j,k,t} by
                (13);
            end
            if F(X^{levy}_{q,j,k,t}) is better than F(X_{q,j,k,t}) then
                X_{q,j,k,t} = X^{levy}_{q,j,k,t}
                F(X_{q,j,k,t}) = F(X^{levy}_{q,j,k,t})
            else
                Keep the old value
            end
        end
    end
    Merge the sub-populations into X;
    if F(cr_best) is better than F(fm_best) then
        K = K + 1
    else if F(cr_best) is worse than F(fm_best) then
        K = K − 1
    else
        K = K
    end
    t = t + 1
end
```

iii. Evaluation. Fitness values are calculated by objective function, which is problem dependent. That is to use (2) without considering valve-point effect, or to use (3) with considering valve-point effect, or to use (4) with considering multi-fuel options and valve-point effects.

iv. Dividing. Divide population X into sub-populations $[X_1, X_2, \cdots, X_{K-1}, X_K]$, as presented in Algorithm 2.

v. Assign algorithm. Each sub-population is calculated by Jaya algorithm with Lévy flights, as presented in Algorithm 4.

vi. Merging. All the sub-populations are merged together into population X.

vii. Update K. Compare the current best value of the objective function $F(cr\_best)$ with the previous best value of the objective function $F(fm\_best)$, then update the K value according to Algorithm 3.

viii. Check the stopping condition. If $N_{\max}$ is reached, stop the loop and report the best solution; otherwise set the iteration number $t = t + 1$ and go back for re-dividing the population.

V. NUMERICAL EXPERIMENTS

Since the proposed Jaya-SML is the hybridization of Jaya, MP method, SA strategy and Lévy flights, it is quite necessary to observe the relative effectiveness of each constituent with application in solving ELDPs, hence four different versions of Jaya algorithm are experimented respectively:

- Jaya: The standard Jaya algorithm to compare with its variants.
- Jaya-M: Jaya algorithm with MP method.
- Jaya-SM: Jaya algorithm with SA strategy and MP method.
- Jaya-SML: Jaya algorithm with SA strategy, MP method and Lévy flights.

In the following, Jaya, Jaya-M, Jaya-SM and Jaya-SML are all applied on ELDPs with different number of generators and constraints to test their own performances. All the compared results are provided from papers where the algorithms were proposed except for Jaya, whose results have been updated from [19]. All the cases are run in Matlab 2016 under windows 7 on Intel(R) Core(TM) i5-6500 CPU 3.20GHz with 8GB RAM.

A. CASE 1

This case includes a 13-units system for load demand of 2520MW and only valve-point effect is considered. System data is provided in [28]. Common parameters including independent run time N_{run}, maximum number of iterations $N_{\max}$ and population size N_{pop} are set to 50, 3000 and 30 for fair comparison. The obtained results are minimum fuel cost $F_{\min}$, average fuel cost F_{avg}, maximum fuel cost $F_{\max}$, standard deviation of the minimum fuel cost *Std* and the execution time T.

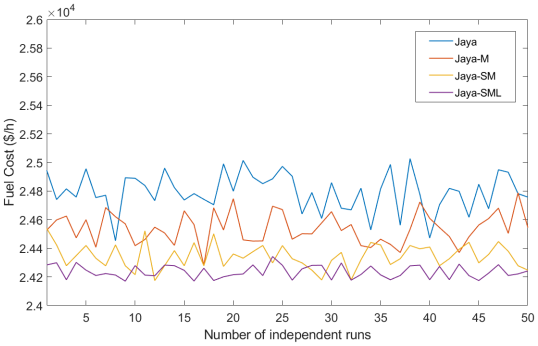

FIGURE 1. Results of independent runs for Case 1 with 13-units system.

To observe the robustness of the four versions of Jaya algorithm in terms of spacing and coverage, Fig. 1 provides the value distributions of $F_{\min}$ over 50 independently running trials and Fig. 2 shows the distribution results in box plot format. It can be clearly observed that the solution qualities obtained by Jaya, Jaya-M and Jaya-SM are all inferior to the solution quality obtained by Jaya-SML, which is more consistent, stable and reliable due to all the individuals are close to the best value. Fig. 3 shows the convergent curves chosen from 50 runs, it can be observed that Jaya-M and Jaya-SM

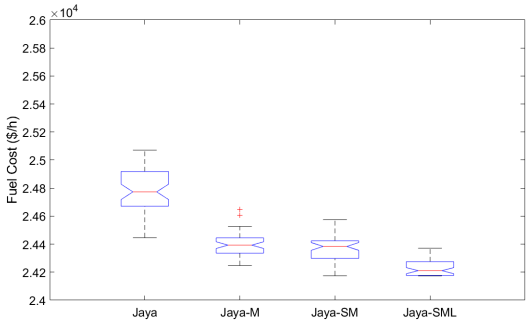

FIGURE 2. Boxplot of the results for Case 1 with 13-units system.

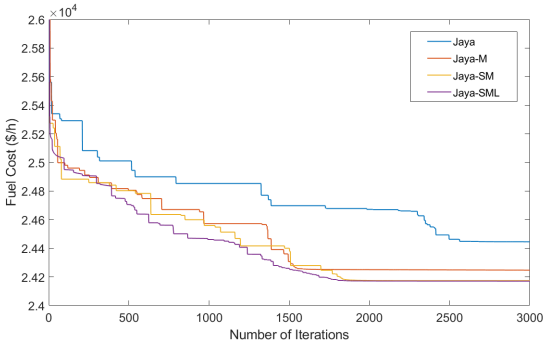

FIGURE 3. Convergence characteristic for Case 1 with 13-units system.

TABLE 1. Result comparison for Case 1 with 13-units system.

Methods	Total Fuel Cost ($/h)			Std	T(s)
	$F_{\min}$	F_{avg}	$F_{\max}$		
GA [29]	24398.23	NA	NA	NA	NA
SA [29]	24970.91	NA	NA	NA	NA
HSS [29]	24275.71	NA	NA	NA	NA
EP-SQP [30]	24266.44	NA	NA	NA	NA
PSO-SQP [30]	24261.05	NA	NA	NA	NA
CPSO [31]	24211.56	NA	NA	NA	NA
CPSO-SQP [31]	24190.97	NA	NA	NA	NA
PSO [32]	24262.73	24271.92	24277.81	NA	NA
FAPSO [32]	24170.93	24173.01	24176.40	NA	NA
FAPSO-NM [32]	24169.92	24170.00	24170.50	NA	NA
FAMPSO [33]	24169.92	**24169.92**	**24169.92**	NA	NA
UHGA [34]	24172.25	NA	NA	NA	NA
TSA [35]	24171.21	24184.06	24392.20	41.00	NA
DSPSO-TSA [35]	24169.92	24173.14	24230.80	**7.72**	NA
CJAYA [19]	24178.8040	24385.7604	NA	NA	NA
MP-CJAYA [19]	24175.5444	24228.1331	NA	NA	NA
Jaya	24177.6049	24436.3863	24577.0292	93.4513	**1.32**
Jaya-M	24171.6158	24387.4825	24486.7267	81.0386	1.98
Jaya-SM	24170.7655	24279.9223	24377.9884	69.7049	1.86
Jaya-SML	**24169.9087**	24217.0898	24285.8889	52.9052	2.45

have greatly improved the solution quality and convergence rate compared with Jaya, while Jaya-SML has outperformed Jaya-M and Jaya-SM in achieving the minimum value of fuel cost.

Result comparison with state-of-the-art methods is shown in Table 1. Obviously, the proposed Jaya-SML algorithm has obtained the best value of $F_{\min}$ among all the compared methods, which is as low as 24169.9087$\$/h$, but it has not obtained the best value of F_{avg} and $F_{\max}$. However, if we compare Jaya, Jaya-M, Jaya-SM and Jaya-SML separately, we can observe that Jaya-SML has improved all the qualities of $F_{\min}$, F_{avg}, $F_{\max}$ and *Std* than the others. The optimal

solutions by the four algorithms for this case are given in Appendix.

B. CASE 2

Jaya-SML algorithm is further evaluated by 40 generating units with load demand of 10500MW in this case to investigate its effectiveness for larger scale power system. System data is provided in [28] and only the valve-point effect is considered. Common parameters of N_{run} and $N_{\max}$ are still fixed at 50 and 3000 respectively, whereas N_{pop} is increased from 30 to 50. The obtained results include $F_{\min}$, F_{avg}, $F_{\max}$, Std and T.

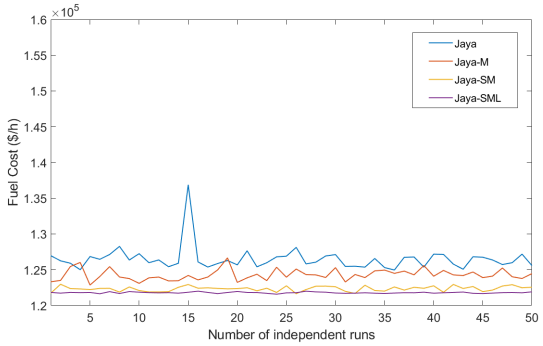

FIGURE 4. Results of independent runs for Case 2 with 40-units system.

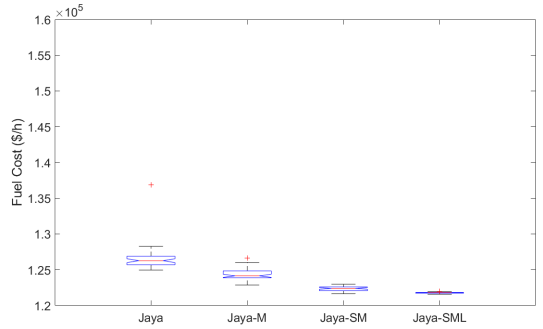

FIGURE 5. Boxplot of the results for Case 2 with 40-units system.

Fig. 4 shows the value distributions of $F_{\min}$ and Fig. 5 shows the distribution results in box plot format. Similar to Case 1, these two figures illustrate the span of $F_{\min}$ in detail. We can visually observe that Jaya-SML behaves the strongest robustness and coherence since all individuals stay more or less steady at the best value, whereas individuals by Jaya-M and Jaya-SM vary much more than Jaya-SML, while Jaya shows the worst performance. Fig. 6 provides the convergence characteristic chosen from 50 trails, it can be observed that Jaya-SML has the best solution quality and the fastest convergence rate.

The result comparison is shown in Table 2. It can be seen that the proposed Jaya-SML has obtained the best value of $F_{\min}$ among all the compared methods, which is as low as 121476.3977$/h. Meanwhile, Jaya-SML has also obtained the best value of F_{avg} and $F_{\max}$, as well as Std, which has obviously verified that Jaya-SML has the strongest

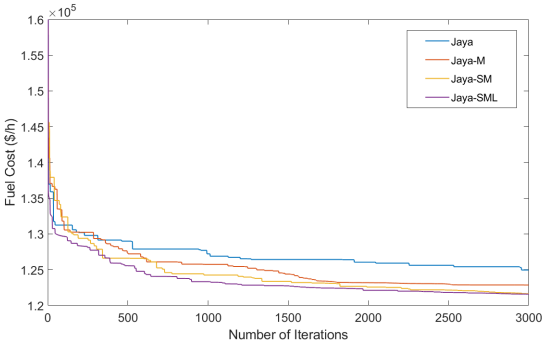

FIGURE 6. Convergence characteristic for Case 2 with 40-units system.

TABLE 2. Result comparison for Case 2 with 40-units system.

Methods	Total Fuel Costs ($/h)			Std	T(s)
	$F_{\min}$	F_{avg}	$F_{\max}$		
PSO-LRS [12]	122035.7946	122558.4565	123461.6794	NA	15.75
NPSO [12]	121704.7391	122221.3697	122995.0976	NA	**3.52**
NPSO-LRS [12]	121664.4308	122209.3185	122981.5913	NA	3.93
SPSO [36]	122049.66	NA	NA	NA	NA
PC-PSO [36]	121767.89	NA	NA	NA	NA
SOH-PSO [36]	121501.14	121853.57	122446.30	NA	NA
EP-SQP [30]	122323.97	122379.63	NA	NA	997.73
PSO-SQP [30]	122094.67	122245.25	NA	NA	733.97
MPSO [5]	122252.2650	NA	NA	NA	NA
PSO [37]	121735.47	122513.92	123467.41	NA	NA
APSO(1) [37]	121704.74	122221.37	122995.10	NA	NA
APSO(2) [37]	121663.52	122153.67	122912.40	NA	NA
CPSO [38]	121885.11	122469.64	123767.36	307.15	NA
PSO-GM [38]	121845.98	122398.38	123219.22	258.44	NA
CBPSO-RVM [38]	121555.32	122281.14	123094.98	259.99	NA
HDE [39]	121813.26	122705.66	NA	NA	6.92
ST-HDE [39]	121698.51	122304.30	NA	NA	6.07
DEC-SQP [15]	121741.9793	122295.1278	122839.2941	386.1809	14.26
TLBO [11]	124517.27	126581.56	128207.06	1060	NA
CTLBO [11]	121553.83	121790.23	122116.18	150	NA
CJAYA [19]	121516.97	121926.77	NA	NA	NA
MP-CJAYA [19]	121480.10	121861.08	NA	NA	NA
Jaya	121733.5492	122279.1504	122707.1277	243.6377	9.89
Jaya-M	121516.9603	121814.4651	122269.0088	186.9668	11.77
Jaya-SM	121485.0974	121801.9415	122150.9126	177.6231	10.38
Jaya-SML	**121476.3977**	**121689.0773**	**122039.8731**	**147.8901**	12.89

capabilities for handling ELDPs with large number of generators. Optimal solutions for this case are given in Appendix.

C. CASE 3

In this case, Jaya-SML algorithm is applied to 6-units system with constraints of ramp rate limit, transmission loss and prohibited operating zones (POZs). System data and B-coefficients are taken from [40]. There are two POZs in every generator with increasing number of non-convex decision spaces, this situation causes challenging complexity to find global optimum. Common parameters of N_{run}, $N_{\max}$ and N_{pop} are set to 50, 2000 and 30 for fair comparison. Obtained results include $F_{\min}$, F_{avg}, $F_{\max}$, Std and T.

Fig. 7 provides the value distributions of $F_{\min}$ over 50 independently running trials and Fig. 8 shows the distribution results in box plot format. It can be seen that within 50 independent runs, Jaya-M has lower fuel cost and higher robustness than Jaya because of the extra MP method, while Jaya-SM obtained even lower fuel cost and higher robustness than Jaya-M because of SA strategy, while Jaya-SML

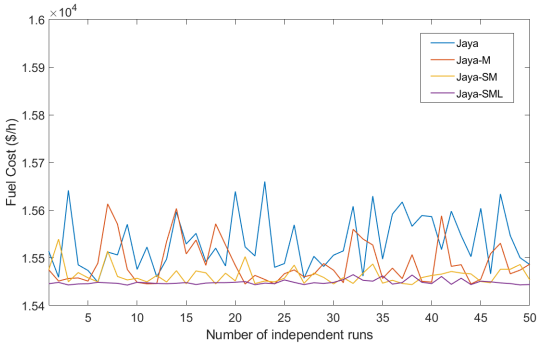

FIGURE 7. Results of independent runs for Case 3 with 6-units system.

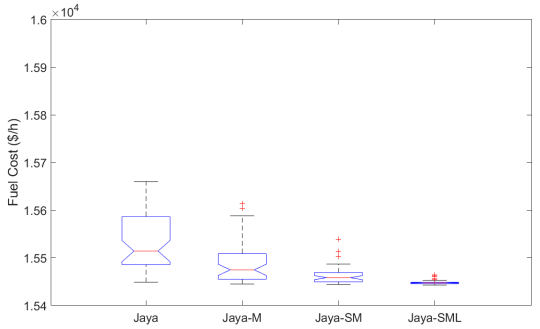

FIGURE 8. Boxplot of the results for Case 3 with 6-units system.

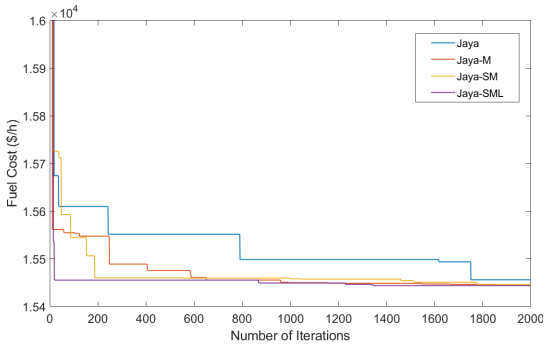

FIGURE 9. Convergence characteristic for Case 3 with 6-units system.

achieved the lowest value of fuel cost and maintained the highest robustness because of the Lévy distribution in population updating phase. From the convergence curve in Fig. 9, we can observe that Jaya-SML approaches optimum value within no more than 50 iterations, which is the fastest speed in convergence rate among the four algorithms.

Table 3 illustrates that the proposed Jaya-SML algorithm has obtained the best value of $F_{\min}$ and F_{avg} as 15445.1682\$/h and 15447.2910\$/h. The best value of $F_{\max}$ is achieved by DE as 15449.777\$/h [42], which actually is very close to 15450.6497\$/h that obtained by Jaya-SML. Optimal solutions for this case are given in Appendix.

D. CASE 4

A larger size of 15-units system with the same constraints as in Case 3 is experimented in this case. System data and

TABLE 3. Result comparison for Case 3 with 6-units system.

Methods	Total Fuel Costs ($/h)			Std	T(s)
	$F_{\min}$	F_{avg}	$F_{\max}$		
SA [41]	15461.10	15488.98	15545.50	28.3678	50.36
GA [41]	15457.96	15477.71	15524.69	17.4072	46.60
TS [41]	15454.89	15472.56	15498.05	13.7195	20.55
PSO [41]	15450.14	15465.83	15491.71	10.1502	6.82
MTS [41]	15450.06	15451.17	15453.64	0.9287	1.29
PSO-LRS [12]	15450.00	15454.00	15455.00	NA	NA
NPSO [12]	15450.00	15452.00	15454.00	NA	NA
NPSO-LRS [12]	15450.00	15450.50	15452.00	NA	NA
SPSO [36]	15466.63	15523.64	15642.68	NA	**0.0602**
PC-PSO [36]	15453.09	15514.98	15633.30	NA	0.0643
SOH-PSO [36]	15446.02	15497.35	15609.64	NA	0.0633
AIS [6]	15448.00	15472.00	15459.70	6.252	NA
DE [42]	15449.766	15449.874	**15449.777**	NA	NA
FA [13]	15450.5090	15452.5310	15458.4427	2.048	1.965
CMFA [13]	15449.8994	15449.8994	15449.8994	**8.96E-06**	2.724
CJAYA [19]	15446.71	15461.62	15484.34	NA	NA
MP-CJAYA [19]	15446.17	15449.23	15451.68	NA	NA
Jaya	15446.5675	15489.7034	15573.5151	13.3122	1.30
Jaya-M	15446.0196	15464.8797	15498.5242	10.9328	1.80
Jaya-SM	15445.8001	15459.4744	15468.9230	8.4657	1.87
Jaya-SML	**15445.1682**	**15447.2910**	15450.6497	6.2214	2.11

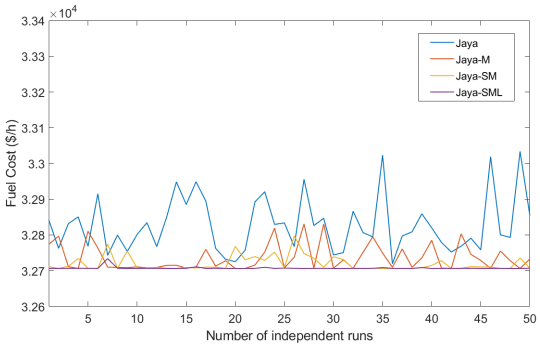

FIGURE 10. Results of independent runs for Case 4 with 15-units system.

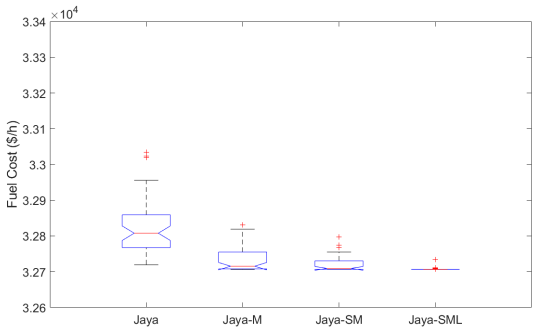

FIGURE 11. Boxplot of the results for Case 4 with 15-units system.

B-coefficients are taken from [40]. There are four generators having POZs. Generators 2, 5 and 6 have three POZs and generator 12 has two POZs. N_{run} is still fixed at 50, whereas $N_{\max}$ and N_{pop} are modified to 3000 and 50 to cope with the increased number of non-convex decision spaces resulted by increased number of generators as well as the POZs. Obtained results include $F_{\min}$, F_{avg}, $F_{\max}$, Std and T.

Fig. 10 provides the value distributions of $F_{\min}$ and Fig. 11 shows the distribution results in box plot format. Convergence curve of $F_{\min}$ is shown in Fig. 12. We can observe from these figures that Jaya-M performs better than Jaya in convergence

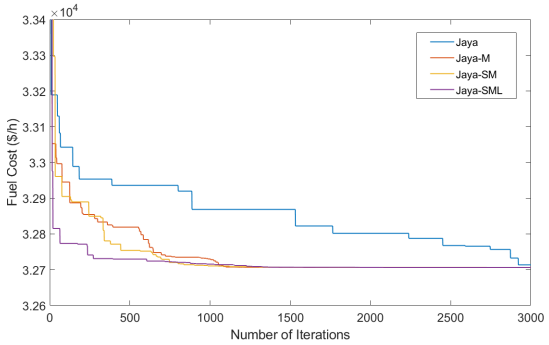

FIGURE 12. Convergence characteristic of Case 4 with 15-units system.

TABLE 4. Result comparison for Case 4 with 15-units system.

Methods	Total Fuel Costs ($/h)			Std	T(s)
	F_{min}	F_{avg}	F_{max}		
SA [41]	32786.40	32869.51	33028.95	112.32	71.25
GA [41]	32779.81	32841.21	33041.64	81.22	48.17
TS [41]	32762.12	32822.84	32942.71	60.59	26.41
PSO [41]	32724.17	32807.45	32841.38	21.24	13.25
MTS [41]	32716.87	32767.21	32796.15	17.51	3.65
TSA [35]	32917.87	33066.76	33245.54	66.82	25.75
DSPSO-TSA [35]	32715.06	32724.63	32730.39	8.40	**2.30**
AIS [6]	32854.00	32873.25	32892.00	10.8079	NA
CJAYA [19]	32710.0768	32740.0719	32828.6554	NA	NA
MP-CJAYA [19]	32706.5158	32706.7150	32708.8736	NA	NA
Jaya	32712.6458	32743.4613	32822.9993	47.0256	3.80
Jaya-M	32707.0312	32714.4386	32743.6808	12.0972	5.07
Jaya-SM	32706.9830	32709.0463	32728.2292	8.7817	4.40
Jaya-SML	**32706.3578**	**32706.6764**	**32707.2925**	**2.3244**	5.14

quality and consistency due to the enhanced population diversity provided by MP method, Jaya-SM shows superiority over Jaya-M due to the adaptive value of K provided by SA strategy, while Jaya-SML has made the biggest improvements in accelerating the convergence rate and maintaining the lowest cost among 50 independent runs.

Result comparisons are shown in Table 4. Obviously, the proposed Jaya-SML algorithm has obtained the best value of $F_{\min}$, F_{avg}, $F_{\max}$ and *Std* among all the compared methods. Especially for *Std*, which is as least as 2.3244, means Jaya-SML has extremely high precision in achieving the global optimum. Conclusion is that Jaya-SML has powerful capabilities of handling larger size of ELDPs with complex constrained conditions. Optimal solutions are given in Appendix.

E. CASE 5

In the last case, a 10-units system with multi-fuel options and valve-point effect for load demand of 2700MW is experimented. Different from aforementioned cases, the objective function for each generator in this case consists of two or three piecewise-quadratic cost functions representing different fuel types. System data is taken from [4]. Common parameters of N_{run} is still fixed at 50, whereas $N_{\max}$ and N_{pop} are set to 1000 and 30 respectively. The obtained results include $F_{\min}$, F_{avg}, $F_{\max}$, *Std* and T.

Fig. 13 and Fig. 14 illustrate the span of $F_{\min}$ in detail. We can observe that Jaya, Jaya-M and Jaya-SM are able to give continuously decreasing values of $F_{\min}$ while Jaya-SML

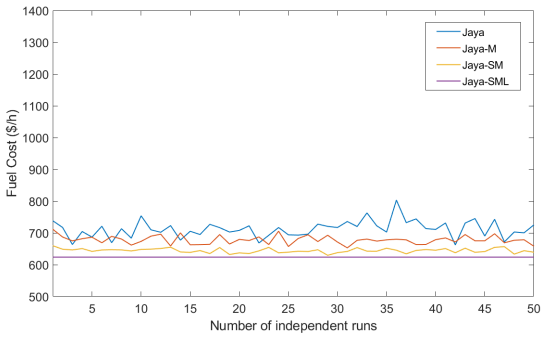

FIGURE 13. Results of independent runs for Case 5 with 10-units system.

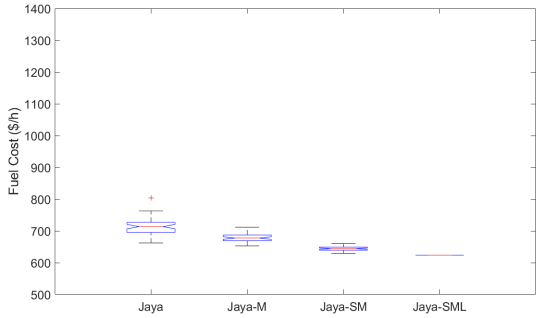

FIGURE 14. Boxplot of the results for Case 5 with 10-units system.

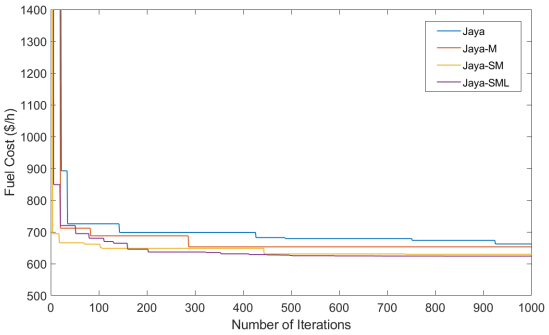

FIGURE 15. Convergence characteristic for Case 5 with 10-units system.

has achieved the lowest fuel cost. Moreover, Jaya-SML not only achieves the lowest fuel cost but also maintains almost the same lowest value during all the independent running times, which confirms its powerful capability of convergence and robustness. Fig. 15 shows the convergence characteristic chosen from 50 independent runs, it can be concluded that Jaya-SML has the best solution quality and the fastest convergence rate compared with Jaya, Jaya-M and Jaya-SM.

Comparisons are shown in Table 5. It is observed that CBPSO-RVM [38] has reached the best value of $F_{\min}$ as 623.9588$/$h, the proposed Jaya-SML ranked the second place as 623.9738$/$h. However when we analysis the values of F_{avg} and $F_{\max}$, we can see that Jaya-SML has achieved the first place in ranking for both of them, which is as low as 624.0468$/$h and 624.1300$/$h respectively. Moreover, the best value of *Std* is also achieved by Jaya-SML, which is as least as 0.0327, it means almost all the 50 independent solutions have converged to the global optimum.

TABLE 5. Result comparison for Case 5 with 10-units system.

Methods	Total Fuel Costs ($/h)			Std	T(s)
	F_{min}	F_{avg}	F_{max}		
PSO-LRS [12]	624.2297	625.7887	628.3214	NA	0.88
NPSO [12]	624.1624	625.2180	627.4237	NA	**0.35**
NPSO-LRS [12]	624.1273	624.9985	626.9981	NA	0.52
CGA-MU [4]	624.7193	627.6087	633.8652	NA	26.64
IGA-MU [4]	624.5178	625.8692	630.8705	NA	7.32
CPSO [38]	624.1715	624.5493	624.7844	0.1278	NA
PSO-GM [38]	624.3050	624.6749	625.0854	0.1580	NA
CBPSO-RVM [38]	**623.9588**	624.0816	624.2930	0.0576	NA
PSO [37]	624.3506	625.8198	629.1073	NA	NA
APSO(1) [37]	624.1624	625.2180	627.4237	NA	NA
APSO(2) [37]	624.0145	624.8185	627.3049	NA	NA
TSA [35]	624.3078	624.8285	635.0623	1.1593	9.71
GA [35]	624.5050	624.7419	624.8169	0.1005	18.37
PSO [35]	624.3045	624.5054	625.9252	0.1749	11.04
Jaya	624.6819	626.1531	637.5108	1.6584	5.29
Jaya-M	624.4959	625.9222	630.7652	0.8578	4.88
Jaya-SM	624.0850	624.2788	624.9105	0.1139	4.86
Jaya-SML	623.9738	**624.0468**	**624.1300**	**0.0327**	7.31

TABLE 6. Result comparison for Case 4 with different population size N_{pop} and K value.

N_{pop}	Minimum Fuel Cost $F_{\min}$ ($/h)				
	K=1	K=2	K=5	K=10	Adaptive K
10	32726.4036	32720.3048	32721.7049	32727.5490	**32719.9215**
20	32721.5161	32717.6211	32715.6761	32721.6341	**32711.4402**
30	32715.6830	32712.3094	32710.6970	32711.8766	**32707.2889**
40	32714.8774	32710.4459	32707.8741	32707.8627	**32707.1126**
50	32712.6458	32709.6691	32707.0312	32707.3474	**32706.9830**

VI. DISCUSSION AND CONCLUSION

In this study, Jaya algorithm with self-adaptive multi-population and Levy flights (Jaya-SML) is proposed and experimented in solving economic load dispatch problems (ELDPs) with different constrained conditions. There are three advantages for the implementation of Jaya-SML algorithm. Firstly, compared with the standard Jaya, there is not a single one parameter has been added throughout the whole procedure, except for the K value which can be initialised as 2. Secondly, it is simple to use, because no complicated techniques are introduced, only MP method, SA strategy and Lévy flights are involved in the algorithm. Thirdly, it can be easily combined with different evolutionary algorithms (EAs), such as PSO, DE and GA, this is one of the author's interests for future work.

By summarizing the ELDP cases, we can get two conclusions. The first one is, Jaya-SML algorithm is the best performer in aspects of solution quality, convergence rate and stability among Jaya, Jaya-M, Jaya-SM and Jaya-SML, regardless of the generator numbers and constrained conditions. The second one is, compared with the other hybrid-heuristic algorithms such as CPSO-SQP, NPSO-LRS, CBPSO-RVM, MP-CJAYA and so on, Jaya-SML achieves the best or nearly best results in reducing the total fuel cost of $F_{\min}$, F_{avg} and $F_{\max}$.

However, it needs to be mentioned that in steady-size MP method, different K value results in different solution quality. If K is too small, it can not contribute to the population diversity; if K is too big, the number of individuals in each sub-population will be too small to perform comprehensive search within the subregions. To illustrate these two situations, Table 6 provides the comparisons for $F_{\min}$ in Case 4 with different population size N_{pop} and steady K value, as well as adaptive K. We can observe that for different N_{pop}, the best value of $F_{\min}$ is not always achieved by the same K value, actually it depends on the complexity of the problem to be solved. However, by using adaptive K value, we do not have to test the perfect value for K, since it is able to adapt

TABLE 7. The optimal solution for Case 1 with 13-units system.

Unit	Jaya	Jaya-M	Jaya-SM	Jaya-SML
1	628.2987	628.1543	628.3097	628.3185
2	298.9136	299.1229	299.1872	299.1993
3	298.5001	299.1932	298.8792	299.1993
4	159.6981	159.4463	159.6752	159.7331
5	159.5153	159.7002	159.6668	159.7331
6	159.0483	159.6522	159.6952	159.7331
7	159.5602	159.6385	159.5858	159.7331
8	159.4420	159.7012	159.7286	159.7331
9	159.6187	159.5676	159.6722	159.7331
10	113.7344	77.1030	77.3290	77.3999
11	76.8710	76.8889	77.3559	77.3999
12	91.7618	89.8069	92.2491	92.3999
13	55.0158	92.0428	88.7024	87.6711
P_{total}(MW)	2519.98	2520.00	2520.00	2519.99
F_{cost}($/h)	24177.6049	24171.6158	24170.7655	24169.9087

TABLE 8. The optimal solution for Case 2 with 40-units system.

Unit	Jaya	Jaya-M	Jaya-SM	Jaya-SML
1	114.0000	113.5252	110.7999	110.7998
2	113.0278	110.7998	110.7998	110.7998
3	99.9503	120.0000	97.3999	97.3999
4	179.8226	179.7331	179.7331	179.7331
5	96.2457	97.0000	97.0000	96.3199
6	140.0000	140.0000	140.0000	140.0000
7	299.7289	300.0000	259.5997	300.0000
8	285.6087	284.5997	284.5997	284.5997
9	286.6349	284.5997	284.5997	284.5997
10	130.2800	130.0000	130.0000	130.0000
11	94.2775	94.0000	168.7998	94.0000
12	94.0287	94.0000	94.0000	94.0000
13	125.0160	125.0000	304.5196	125.0000
14	484.6313	394.2794	394.2794	394.2794
15	304.4157	394.2794	304.5196	394.2794
16	394.2211	394.2794	304.5196	394.2794
17	489.6003	489.2794	489.2794	489.2794
18	489.8317	489.2794	489.2794	489.2794
19	511.8611	511.2794	511.2794	511.2794
20	511.3126	511.2794	511.2794	511.2794
21	524.0884	523.2794	523.2794	523.2794
22	523.8282	523.2794	523.2794	523.2794
23	523.6836	523.2794	523.2794	523.2794
24	524.0062	523.2794	523.2794	523.2794
25	524.5391	523.2794	523.2794	523.2794
26	526.4563	523.2794	523.2794	523.2794
27	10.0789	10.0000	10.0000	10.0000
28	10.2151	10.0000	10.0000	10.0000
29	10.6094	10.0000	10.0000	10.0000
30	96.5283	97.0000	87.9271	87.7999
31	189.8846	190.0000	190.0000	190.0000
32	190.0000	190.0000	190.0000	190.0000
33	189.7319	190.0000	190.0000	190.0000
34	200.0000	164.7999	200.0000	200.0000
35	170.1334	200.0000	200.0000	200.0000
36	199.9047	200.0000	164.7998	200.0000
37	109.4948	110.0000	110.0000	110.0000
38	109.9924	110.0000	110.0000	110.0000
39	110.0000	110.0000	110.0000	110.0000
40	512.2693	511.2794	511.2794	511.2794
P_{total}(MW)	10500.05	10499.97	10499.97	10499.97
F_{cost}($/h)	121733.5492	121516.9603	121485.0974	121476.3977

TABLE 9. The optimal solution for Case 3 with 6-units system.

Unit	Jaya	Jaya-M	Jaya-SM	Jaya-SML
1	429.3031	456.7211	453.5270	447.8944
2	177.0305	163.9166	171.7646	182.9939
3	267.5871	269.6730	261.4402	264.0069
4	143.0923	140.3796	147.3180	139.9736
5	165.3239	155.0737	154.1639	154.7018
6	93.0325	89.6049	87.0225	85.7651
P_{total}(MW)	1275.3694	1275.3686	1275.2361	1275.3357
P_{loss}(MW)	12.3605	12.3686	12.2160	12.3356
F_{cost}(\$/h)	15446.5675	15446.0196	15445.8001	15445.1682

TABLE 10. The optimal solution for Case 4 with 15-units system.

Unit	Jaya	Jaya-M	Jaya-SM	Jaya-SML
1	455.0000	455.0000	455.0000	454.9999
2	380.0000	380.0000	380.0000	380.0000
3	130.0000	130.0000	130.0000	130.0000
4	130.0000	130.0000	130.0000	130.0000
5	170.0000	170.0000	170.0000	170.0000
6	459.4440	460.0000	460.0000	460.0000
7	430.0000	430.0000	430.0000	430.0000
8	108.4288	60.0000	60.9089	71.4456
9	25.0000	70.8256	69.9168	59.3587
10	158.3287	160.0000	160.0000	160.0000
11	80.0000	80.0000	80.0000	79.9997
12	80.0000	80.0000	80.0000	80.0000
13	25.0000	25.0000	25.0000	25.0000
14	15.0000	15.0000	15.0000	15.0000
15	15.0000	15.0000	15.0000	15.0000
P_{total}(MW)	2661.2015	2660.8256	2660.8257	2660.8039
P_{loss}(MW)	31.1936	30.8280	30.8288	30.8039
F_{cost}(\$/h)	32712.6458	32707.0312	32706.9830	32706.3578

himself to the complexity of the problem and always can achieve the best solutions, which is the benefits of adopting self-adaptive MP method.

It should also be addressed that, when using MP method, two issues need to be considered. One is how to guide the sub-populations to move toward subregions in a more promising way except for the "*random way*" used in this paper. This issue is crucial since if the sub-populations cannot move to different subregions, Jaya cannot search and track the local optima in subregions. The other issue is concerned on how to define suitable space size for each subregion. If the subregion is too small, the population might converge fast and the population diversity will be lost, then no progress can be made during the iteration. However if the subregion is too large, there is a potential that more than one local optima are covered within the area, due to the updating mechanism only one local optima is recorded, so the useful information of the other local optima cannot be kept, which is also a bad loss of diversity. These two issues will be further investigated by the author.

As long as Jaya-SML algorithm has gained outstanding superiority in solving the ELDPs, it is supposed to be applied to other optimization problems such as micro grid power dispatch problems and dynamic optimization problems, to broaden its applications in power system in the future.

APPENDIX

See Tables 7–11.

TABLE 11. The optimal solution for Case 5 with 10-units system.

Unit	Fuel Type	Jaya	Jaya-M	Jaya-SM	Jaya-SML
1	2	215.9857	214.3788	218.2575	217.7758
2	1	215.0708	214.0979	212.6408	211.6629
3	1	282.8981	277.3448	283.4240	279.4465
4	3	237.7716	242.3610	238.7602	240.3575
5	1	278.6049	279.3389	282.2614	279.9952
6	3	235.8957	241.0124	239.1227	240.0703
7	1	281.2280	287.9884	287.2395	285.9509
8	3	240.9229	246.1329	240.2153	240.3611
9	3	440.0000	421.7795	424.8005	429.0488
10	1	271.6440	275.5739	273.2647	275.3385
P_{total}(MW)		2700.02	2700.01	2700.00	2700.01
F_{cost}(\$/h)		624.6819	624.4959	624.0850	623.9738

ACKNOWLEDGMENT

The authors would like to thank Yeungnam University for all the supports in terms of fellowship to Jiang-Tao Yu.

REFERENCES

[1] G. Abbas, J. Gu, U. Farooq, M. U. Asad, and M. El-Hawary, "Solution of an economic dispatch problem through particle swarm optimization: A detailed survey—Part I," *IEEE Access*, vol. 5, pp. 15105–15141, 2017.
[2] G. Abbas, J. Gu, U. Farooq, A. Raza, M. U. Asad, and M. E. El-Hawary, "Solution of an economic dispatch problem through particle swarm optimization: A detailed survey—Part II," *IEEE ACCESS*, vol. 5, pp. 24426–24445, 2017.
[3] W.-M. Lin, F.-S. Cheng, and M.-T. Tsay, "An improved tabu search for economic dispatch with multiple minima," *IEEE Trans. Power Syst.*, vol. 17, no. 1, pp. 108–112, Feb. 2002.
[4] C.-L. Chiang, "Improved genetic algorithm for power economic dispatch of units with valve-point effects and multiple fuels," *IEEE Trans. Power Syst.*, vol. 20, no. 4, pp. 1690–1699, Nov. 2005.
[5] J.-B. Park, K.-S. Lee, J.-R. Shin, and K. Y. Lee, "A particle swarm optimization for economic dispatch with nonsmooth cost functions," *IEEE Trans. Power Syst.*, vol. 20, no. 1, pp. 34–42, Feb. 2005.
[6] B. K. Panigrahi, S. R. Yadav, S. Agrawal, and M. K. Tiwari, "A clonal algorithm to solve economic load dispatch," *Electr. Power Syst. Res.*, vol. 77, no. 10, pp. 1381–1389, Aug. 2007.
[7] L. S. Coelho and V. C. Mariani, "An improved harmony search algorithm for power economic load dispatch," *Energy Convers. Manage.*, vol. 50, no. 10, pp. 2522–2526, Oct. 2009.
[8] X.-S. Yang, S. S. S. Hosseini, and A. H. Gandomi, "Firefly algorithm for solving non-convex economic dispatch problems with valve loading effect," *Appl. Soft Comput.*, vol. 12, no. 3, pp. 1180–1186, Mar. 2012.
[9] A. Bhattacharya and P. K. Chattopadhyay, "Biogeography-based optimization for different economic load dispatch problems," *IEEE Trans. Power Syst.*, vol. 25, no. 2, pp. 1064–1077, May 2010.
[10] D. C. Secui, "A new modified artificial bee colony algorithm for the economic dispatch problem," *Energy Convers. Manage.*, vol. 89, no. 1, pp. 43–62, Jan. 2015.
[11] X. He, Y. Rao, and J. Huang, "A novel algorithm for economic load dispatch of power systems," *Neurocomputing*, vol. 171, no. 1, pp. 1454–1461, Jan. 2016.
[12] A. I. Selvakumar and K. Thanushkodi, "A new particle swarm optimization solution to nonconvex economic dispatch problems," *IEEE Trans. Power Syst.*, vol. 22, no. 1, pp. 42–51, Feb. 2007.
[13] Y. Yang, B. Wei, H. Liu, Y. Zhang, J. Zhao, and E. Manla, "Chaos firefly algorithm with self-adaptation mutation mechanism for solving large-scaleeconomic dispatch with valve-point effects and multiple fuel options," *IEEE Access*, vol. 6, pp. 45907–45922, 2018.
[14] T. Niknam, H. D. Mojarrad, and H. Z. Meymand, "A novel hybrid particle swarm optimization for economic dispatch with valve-point loading effects," *Energy Convers. Manage.*, vol. 52, no. 4, pp. 1800–1809, Apr. 2011.
[15] L. dos Santos Coelho and V. C. Mariani, "Combining of chaotic differential evolution and quadratic programming for economic dispatch optimization with valve-point effect," *IEEE Trans. Power Syst.*, vol. 21, no. 2, pp. 989–996, May 2006.
[16] L. S. Coelho, T. C. Bora, and V. C. Mariani, "Differential evolution based on truncated Lévy-type flights and population diversity measure to solve economic load dispatch problems," *Int. J. Elect. Power Energy Syst.*, vol. 57, pp. 178–188, May 2014.

[17] J. S. Alsumait, J. K. Sykulski, and A. K. Al-Othman, "A hybrid GA–PS–SQP method to solve power system valve-point economic dispatch problems," *Appl. Energy*, vol. 87, no. 5, pp. 1773–1781, May 2010.
[18] X. Shen, G. Wu, R. Wang, H. Chen, H. Li, and J. Shi, "A self-adapted across neighborhood search algorithm with variable reduction strategy for solving non-Convex static and dynamic economic dispatch problems," *IEEE Access*, vol. 6, pp. 41314–41324, 2018.
[19] J. Yu, C.-H. Kim, A. Wadood, T. Khurshiad, and S.-B. Rhee, "A novel multi-population based chaotic JAYA algorithm with application in solving economic load dispatch problems," *Energies*, vol. 11, p. 1946, Jul. 2018.
[20] R. V. Rao, A. Saroj, P. Oclon, J. Taler, and J. Lakshmi, "A posteriori multiobjective self-adaptive multipopulation Jaya algorithm for optimization of thermal devices and cycles," *IEEE Access*, vol. 7, pp. 4113–4134, Dec. 2018.
[21] J. L. Ravipudi and M. Neebha, "Synthesis of linear antenna arrays using Jaya, self-adaptive Jaya and chaotic Jaya algorithms," *AEU—International journal of electronics and communications*, vol. 92, pp. 54–63, Aug. 2018.
[22] R. V. Rao, "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems," *Int. J. Ind. Eng. Comput.*, vol. 7, no. 1, pp. 19–34, 2016.
[23] P. Li and J. Hu, "An ADMM based distributed finite-time algorithm for economic dispatch problems," *IEEE Access*, vol. 6, pp. 30969–30976, 2018.
[24] R. V. Rao and A. Saroj, "A self-adaptive multi-population based Jaya algorithm for engineering optimization," *Swarm Evol. Comput.*, vol. 37, pp. 1–26, Dec. 2017.
[25] R. Jensi and G. W. Jiji, "An enhanced particle swarm optimization with levy flight for global optimization," *Appl. Soft Comput.*, vol. 43, pp. 248–261, Jun. 2016.
[26] H. Hakli and H. Uğuz, "A novel particle swarm optimization algorithm with Levy flight," *Appl. Soft Comput.*, vol. 23, pp. 333–345, Oct. 2014.
[27] T. T. Nguyen and D. N. Vo, "The application of one rank cuckoo search algorithm for solving economic load dispatch problems," *Appl. Soft Comput.*, vol. 37, pp. 763–773, Dec. 2015.
[28] N. Sinha, R. Chakrabarti, and P. K. Chattopadhyay, "Evolutionary programming techniques for economic load dispatch," *IEEE Trans. Evol. Comput.*, vol. 7, no. 1, pp. 83–94, Feb. 2003.
[29] D. B. Das and C. Patvardhan, "Solution of economic load dispatch using real coded hybrid stochastic search," *Int. J. Elect. Power Energy Syst.*, vol. 21, no. 3, pp. 165–170, Mar. 1999.
[30] T. A. A. Victoire and A. E. Jeyakumar, "Hybrid PSO–SQP for economic dispatch with valve-point effect," *Electr. Power Syst. Res.*, vol. 71, no. 1, pp. 51–59, Sep. 2004.
[31] J. Cai, Q. Li, L. Li, H. Peng, and Y. Yang, "A hybrid CPSO–SQP method for economic dispatch considering the valve-point effects," *Energy Convers. Manage.*, vol. 53, no. 1, pp. 175–181, Jan. 2012.
[32] T. Niknam, "A new fuzzy adaptive hybrid particle swarm optimization algorithm for non-linear, non-smooth and non-convex economic dispatch problem," *Appl. Energy*, vol. 87, no. 1, pp. 327–339, Jan. 2010.
[33] T. Niknam, H. D. Mojarrad, and M. Nayeripour, "A new fuzzy adaptive particle swarm optimization for non-smooth economic dispatch," *Energy*, vol. 35, no. 4, pp. 1764–1778, Apr. 2010.
[34] H. Da-kuo, W. Fu-li, and M. Zhi-Zhong, "Hybrid genetic algorithm for economic dispatch with valve-point effect," *Electr. Power Syst. Res.*, vol. 78, no. 4, pp. 626–633, Apr. 2008.
[35] S. Khamsawang and S. Jiriwibhakorn, "DSPSO–TSA for economic dispatch problem with nonsmooth and noncontinuous cost functions," *Energy Convers. Manage.*, vol. 51, no. 2, pp. 365–375, Feb. 2010.
[36] K. T. Chaturvedi, M. Pandit, and L. Srivastava, "Self-organizing hierarchical particle swarm optimization for nonconvex economic dispatch," *IEEE Trans. Power Syst.*, vol. 23, no. 3, pp. 1079–1087, Aug. 2008.
[37] A. I. Selvakumar and K. Thanushkodi, "Anti-predatory particle swarm optimization: Solution to nonconvex economic dispatch problems," *Electr. Power Syst. Res.*, vol. 78, no. 1, pp. 2–10, Jan. 2008.
[38] H. Lu, P. Sriyanyong, Y. H. Song, and T. Dillon, "Experimental study of a new hybrid PSO with mutation for economic dispatch with non-smooth cost function," *Int. J. Elect. Power Energy Syst.*, vol. 32, no. 9, pp. 921–935, Nov. 2010.
[39] S. K. Wang, J. P. Chiou, and C. W. Liu, "Non-smooth/non-convex economic dispatch by a novel hybrid differential evolution algorithm," *IET Gener., Transmiss. Distrib.*, vol. 1, no. 5, pp. 793–803, Sep. 2007.
[40] Z.-L. Gaing, "Particle swarm optimization to solving the economic dispatch considering the generator constraints," *IEEE Trans. Power Syst.*, vol. 18, no. 3, pp. 1187–1195, Aug. 2003.
[41] S. Pothiya, I. Ngamroo, and W. Kongprawechnon, "Application of multiple tabu search algorithm to solve dynamic economic dispatch considering generator constraints," *Energy Convers. Manage.*, vol. 49, no. 4, pp. 506–516, Apr. 2008.
[42] N. Noman and H. Iba, "Differential evolution for economic load dispatch problems," *Electr. Power Syst. Res.*, vol. 78, no. 8, pp. 1322–1331, Aug. 2008.

JIANG-TAO YU received the B.S. and M.S. degrees from Northeast Forestry University, Harbin, China, in 2009 and 2012, respectively. She is currently pursuing the Ph.D. degree with the Department of Electrical Engineering, Yeungnam University, South Korea. She was with the Department of Electronic Information and Electrical Engineering, Anyang Institute of Technology, China. Her current research interests include power system optimization and evolutionary algorithms.

CHANG-HWAN KIM received the B.S. and M.S. degrees from Yeungnam University, Gyeongbuk, South Korea, in 2013 and 2015, respectively, where he is currently pursuing the Ph.D. degree with the Department of Electrical Engineering. His research interests include power system control and operation, optimal power flow and evolutionary computation, and EMTP and ATP application.

ABDUL WADOOD received the B.S. degree in electronic engineering from the University of Engineering and Technology Peshawar, Pakistan, in 2012, and the M.S. degree in electrical engineering from the Sarhad University of Science and Information Technology, Peshawar, Pakistan, in 2015. He is currently pursuing the Ph.D. degree with the Department of Electrical Engineering, Yeungnam University, South Korea. His research interests include power system relays and power system protection and evolutionary algorithms.

TAHIR KHURSHAID received the B.S. degree in electrical engineering from Jammu University, India, in 2011, and the M.S. degree in power system engineering from Galgotias University, Greater Noida, India, in 2014. He is currently pursuing the Ph.D. degree with the Department of Electrical Engineering, Yeungnam University, South Korea. His research interests include power system protection, power system analysis, and design and power system deregulation.

SANG-BONG RHEE received the B.S., M.S., and Ph.D. degrees from Hanyang University, South Korea, in 1994, 1999, and 2004, respectively. He was a Research Professor with the School of Electrical and Computer Engineering, Sung-Kyunkwan University, South Korea. He is currently an Assistant Professor with the Department of Electrical Engineering, Yeungnam University, Gyeongbuk, South Korea. His research interests include distribution system control and operation, and artificial intelligence applications to power system protection.