Quasi-oppositional symbiotic organisms search algorithm for different economic load dispatch problems

D. Das*, A. Bhattacharya, and R. Narayan Ray

Department of Electrical Engineering, National Institute of Technology, Agartala, Pin-799046, India.

Received 11 April 2018; received in revised form 15 August 2018; accepted 15 October 2018

KEYWORDS
Economic load dispatch;
Opposition-based learning;
Prohibited operating zone;
Symbiotic organisms search;
Valve point loading.

Abstract. In this paper, an effective meta-heuristic technique called quasi-oppositional symbiotic organisms search is applied for solving non-convex economic dispatch problems. Symbiotic organisms search is a soft computing technique, inspired by organisms in the ecosystem. This technique is implemented for improving the solution quality in minimum time. In order to improve the convergence rate, quasi-reflected numbers are used here instead of pseudo-random numbers. Different equality and inequality constraints such as transmission loss, load demand, prohibited operating zone, generator operating limits, and boundary of ramp rate are considered here. The presence of multiple fuels and valve point are also considered in some cases. This algorithm is applied to four different test systems. Simulation results are compared with many recently developed optimization techniques to show the superiority and consistency of this method. Simulation results also show that the computational efficiency of this algorithm is much better than the other meta-heuristic methods available in the literature.

© 2020 Sharif University of Technology. All rights reserved.

1. Introduction

In recent years, the minimization of fuel cost during electrical power production has become a major challenge for power engineers. The objective of economic dispatch is to reduce the price of power generation while satisfying various constraints.

Classical optimization techniques have been developed to find a solution to ELD problems. However, they are subject to various limitations. In linear programming [1] method, generator fuel cost characteristics are approximated as piecewise linear. This is one of the major disadvantages of this method, even though it is fast and reliable. Dynamic programming [2] technique was introduced by Wood and Wollenberg [3] applied this technique for solving the ELD problems. However, with an increase in system size, its execution time also increases. These classical methods are mainly calculus based, and they make use of derivatives. In some cases, these techniques converge to local optimal point. Practical ELD considers ramp rate limit, prohibited operating zone, multiple fuel options, etc. Therefore, the resultant ELD becomes a non-convex optimization problem, which is very difficult to solve by classical methods. Therefore, in recent years, numerous meta-heuristic and heuristic processes have been applied to solve various economic dispatch problems.

Simulated Annealing (SA) [4] was developed in 1993. This algorithm was inspired by annealing in metallurgy, a technique that involves heating and controlled cooling of a material in order to increase the crystals size and reduce their defects. Panigrahi et
al. [5] applied this technique to dynamic economic load dispatch problem for determining the nearly global optimal solution. However, it is not easy to set a suitable value for control parameter, and the convergence speed of this technique is also very low.

In 1993, Walters and Sheble [6] proposed Genetic Algorithm (GA) for solving ELD problems. It was found that the potential of GA for global optimization was very high. However, the performance of this method was not good at identifying local optima, and this algorithm was not effective for smooth unimodal function. Therefore, in order to improve the performance of GA, Improved Genetic Algorithm with Multiplier Updating (IGA-MU) was developed by Chiang [7] in 2005 for solving power economic dispatch.

Kennedy and Eberhart [8] proposed Particle Swarm Optimization (PSO) technique, motivated by flocking of birds and fish schooling. Gaing [9] applied this technique to different constrained ELD problems. However, this method is not always guaranteed to obtain global best solution. There is a chance to get stuck in a local optimal point. Therefore, various modifications and hybridization of PSO such as New PSO with Local Random Search (NPSO LRS) [10], adaptive PSO [11], improved coordinated-aggregation-based PSO [12], improved PSO [13], and species-based quantum particle swarm [14] have been presented to improve the efficiency of this algorithm.

Storn and Price [15] proposed Differential Evolution (DE) based on a population-based stochastic search technique. This technique gives nearly global optimal solution by iterative refining of the population through reproduction and selection. Iba et al. [16] proposed this DE technique in order to solve ELD problems. However, it was found that the performance of this technique was not satisfactory in terms of convergence speed for a large system. Therefore, various modified and hybridized versions of DE with a generator of chaos sequences and sequential quadratic programming [17], DE-PSO-DE [18], and improved DE [19] have been introduced to get a better quality solution.

Evolutionary programming [20] is a soft computing method. This methodology is capable of finding a near-optimal solution. However, the major demerit of this method is that the execution period of this technique is too long and, also, the application of this technique is restricted in practical areas. Therefore, for improving the computational efficiency of this technique, an improved fast evolutionary programming was proposed by Sinha et al. [21] for economic dispatch problems.

In 2008, bacterial foraging with Nelder-Mead algorithm [22] was developed in order to solve the economic dispatch problems. Biogeography-Based Optimization (BBO) was proposed by Simon [23]. Bhattacharya and Chattopadhyay [24] applied this technique to various non-convex ELD problems. Bhattacharya et al. (2010) proposed a hybrid method that combined DE with BBO (DE/BBO) [25] to solve both non-convex and convex ELD problems.

Lam and Li [26] developed Chemical Reaction Optimization (CRO) method. This technique was applied to continuous benchmark function, and the performance of this technique was found satisfactory. Bhattacharjee et al. [27] applied this methodology to the constrained ELD problem for finding nearly global optimal results. Oppositional Real-Coded Chemical Reaction Optimization (ORCCRO) was introduced by Bhattacharjee et al. [28] to solve non-convex optimization problems.

In 2011, Rao et al. [29] developed Teaching Learning-Based Optimization (TLBO) method. TLBO is based on two phases of education and they are ‘Teacher phase’ and ‘Learner phase’. It is observed that the performance of TLBO method is satisfactory when tested on various benchmark functions. In 2013, TLBO method was applied by Bhattacharjee et al. [30] to solve ELD problems. Banerjee et al. [31] proposed TLBO to find the solution of ELD. Chaotic Teaching-Learning Based Optimization with Levy flight (CTLBO) was proposed by He et al. [32] in order to solve ELD problems.

Mirjalili et al. [33] discovered grey wolf optimizer algorithm in order to solve various mathematical problems. Kamboj et al. [34] used this technique to solve the problems of constrained ELD. Self-Adaptive Differential Harmony Search (SADHS) algorithm and Chaotic Self-Adaptive Differential Harmony Search algorithm (CSADHS) techniques were proposed by Rajagopalan et al. [35] in 2014. An efficient Krill herd method was proposed by Mandal et al. [36] for solving both non-convex and convex ELD problems. In 2015, Barisal and Prusty [37] proposed Oppositional Invasive-Weed Optimization (OIWO) method to solve large-scale ELD problems. Mirjalili proposed ant lion optimization [38], which was implemented by Kamboj et al. [39] for solving various problems of ELD. In 2015, Selvathra et al. [40] developed a hybrid cross-entropy method and quadratic programming for solving ELD problems. Al-Betar et al. [41] proposed Tournament-based harmony search algorithm for solving non-convex ELD problems. Ghorbani and Babaee proposed Exchange Market Algorithm (EMA) [42], motivated by the stock exchange trading method, for solving ELD problems. In 2018, Mohammadi et al. introduced Modified Crow Search Algorithm (MCSCA) [43] for solving the non-convex economic load dispatch problem.

In 2014, Cheng and Prayogo [44] proposed a new optimization technique called Symbiotic Organisms Search (SOS) in order to solve various mathematical and engineering design problems. The proposed...
methodology recreates the intelligent conduct seen among creatures in nature. The main advantage of this algorithm is that specific algorithm parameters are not required. In 2016, SOS technique was implemented by Duman [45] to find the solution of optimal power flow. In 2017, Guven et al. [46] proposed SOS method to solve economic dispatch problems.

Opposition-Based Learning (OBL) was proposed by Tizhoosh [47] in order to improve back propagation in neural networks. In order to approach the solution, OBL exploits the opposite numbers. By contrasting a number to the opposite number, a compact search space is required to obtain the correct solution. It was demonstrated that a quasi-opposite number [48] was likely to be closer to the solution as compared to a random number. Further to this, a quasi-opposite number was typically nearer to the solution compared to an opposite number.

Since the quasi-opposite number is used here instead of the pseudo-random number, the solution obtained through the initialization process is nearer to the optimal solution. Hence, the number of iterations required to reach the optimal solution is less than that in other initialization processes. It is observed that the computational efficiency of this technique is better, which is the reason why the present authors have adopted this methodology in Quasi Oppositional Symbiotic Organisms Search (QOSOS) for accelerating the speed of convergence of SOS to a greater extent. It is seen that QOSOS is performed better than SOS and other well-known optimization techniques. In this paper, the QOSOS algorithm is used for solving various ELD problems, and the results obtained by QOSOS method are compared to other optimization techniques. The details of this proposed technique are discussed in Section 4.

The main contribution of this paper is to implement the OBL method in SOS algorithm in order to accelerate the convergence rate of SOS algorithm. It has already been discussed above that if quasi-random number is used instead of pseudo-random number, then the convergence rate improves because the solution obtained through the initialization process is nearer to the best solution. In 2009, Egezez et al. [48] proved that quasi-opposite number was always nearer to the best solution than random number and opposite number. It was also observed that, for an original estimated solution, quasi-reflected set (expected probability is 11/16) had a greater tendency to reach an optimal point as compared to quasi-opposite set (expected probability is 9/16). SOS algorithm itself is a strong optimization technique. This algorithm is validated and tested in various benchmark functions, and it is found that this algorithm gives better performance than the other well-known meta-heuristic based methods. Therefore, the present authors have adopted OBL in SOS techniques in order to improve the convergence rate of SOS and obtain a solution with better quality.

2. Problem formulation

2.1. Objective function

The objective of economic dispatch is to minimize the total cost of power generation while satisfying various constraints. The fitness function of the economic dispatch case can be written as follows:

\[
C_T = \min \sum_{i=1}^{N} C_i(W_i) = \min \sum_{i=1}^{N} a_i + b_i W_i + c_i W_i^2,
\]

where \(C_i(W_i)\) shows the fuel price function of the \(i\)th generator; \(a_i\), \(b_i\), and \(c_i\) are the fuel price coefficients of the \(i\)th unit; \(N\) represents the total number of generating units; \(W_i\) is the output power of the \(i\)th unit. From this equation, it is observed that the fuel cost characteristic is quadratic in nature. However, in practice, the valve-point loading effect is introduced in the objective function. A sudden increase in losses has been observed when steam admission valve is opened. Therefore, a ripple is introduced in the cost function, which is known as the valve-point loading effect.

The overall objective function \(C_T\) of the economic dispatch problem with valve-point effect [28] can be expressed as follows:

\[
C_T = \left( \sum_{i=1}^{N} C_i(W_i) \right) = \left( \sum_{i=1}^{N} a_i + b_i W_i + c_i W_i^2 \right) + \left| E_i \times \sin \left( F_i \times (W_i^{\text{min}} - W_i) \right) \right|
\]

where \(E_i\) and \(F_i\) represent the coefficients of unit \(i\) with the valve-point effects.

Practically, multiple fuel sources are used for dispatching unit, and every unit must be represented with a piecewise quadratic function. In a network, the fuel price function of the generator with valve point and multiple fuel option [27] can be represented by:

\[
C_T = \sum_{i=1}^{N} C_{ip}(W_i),
\]

Eq. (4) is shown in Box I,

where \(N\) is the total number of generators; \(W_{ip-1}\) is the lower limit for fuel option \(p\) of the \(i\)th generating unit; \(a_{ip}, b_{ip}, c_{ip}, E_{ip}, \) and \(F_{ip}\) represent the coefficients of fuel price of the \(i\)th generator with fuel option \(p\).

2.2. Constraints of ELD

2.2.1. Constraint of real power or demand

The overall generation must be equal to transmission loss and system demands. This can be represented as follows:
\[ \sum_{i=1}^{N} W_i - (W_D + W_L) = 0, \quad (5) \]

where \( W_L, W_D \) represent the total transmission loss and total system demand, respectively. The loss due to power transmission \( W_L \) can be calculated as follows:

\[ W_L = \sum_{i=1}^{N} \sum_{j=1}^{N} W_{B_{ij}} W_j + \sum_{i=1}^{N} B_{ai} W_i + B_{ao}. \quad (6) \]

### 2.2.2. Operating limits constraint of generator

The generated power by individual generator must vary within their maximum and minimum limits. Therefore:

\[ W_{i_{\text{min}}} \leq W_i \leq W_{i_{\text{max}}} \quad i = 1, 2, 3 \ldots N, \quad (7) \]

where \( W_{i_{\text{min}}} \) and \( W_{i_{\text{max}}} \) represent the lower and upper limits of the real power output of unit \( i \).

### 2.2.3. Ramp rate limit

In practical circumstances, the operating range of all online units may be confined by the ramp rate limit \[28\]. Depending on up (\( UR_i \)) and down (\( DR_i \)) ramp rate limits, the generation can be increased or decreased.

If generation increases:

\[ W_i - W_{i_{\text{th}}} \leq UR_i. \quad (8) \]

If generation decreases:

\[ W_{i_{\text{th}}} - W_i \leq DR_i. \quad (9) \]

where \( W_{i_{\text{th}}} \) is the power generation of the ith unit at an earlier hour.

### 2.2.4. Prohibited zone constraint

The units of the generator might have some zones where operation is limited because of faults in the machines, steam valve operation, boilers, vibration in shafts, etc. \[28\]. Thus, a discontinuous cost curve is produced in relation to the prohibited operating zone. Prohibited operating zone may be formulated as follows:

\[ W_{i_{\text{min}}} \leq W_i \leq W_{i_{\text{th}}}^{(k)}, \quad 2 \leq k \leq n_i. \quad (10) \]

or:

\[ W_{i_{\text{th}}}^{(k)} \leq W_i \leq W_{i_{\text{th}}}^{(k)} + W_{i_{\text{th}}}^{(k+1)} \]

where \( k \) represents total prohibited zone numbers of the \( i \)th unit. \( W_{i_{\text{th}}}^{(k)} \) and \( W_{i_{\text{th}}}^{(k+1)} \) represent higher and lower limits of the \( k \)th prohibited zone; \( n_i \) represents the total number of prohibited zones of the \( i \)th unit. During optimization, if \( W_{i_{\text{th}}}^{(k)} \geq W_i \geq (W_{i_{\text{th}}}^{(k)} + W_{i_{\text{th}}}^{(k+1)})/2 \), then \( W_i \) will be fixed to \( W_{i_{\text{th}}}^{(k+1)} \). Mathematically, this can be expressed as follows:

\[ W_i = W_{i_{\text{th}}}^{(k)} \quad \text{if} \quad W_{i_{\text{th}}}^{(k)} \geq W_i \geq (W_{i_{\text{th}}}^{(k)} + W_{i_{\text{th}}}^{(k+1)})/2, \quad (11) \]

\[ k = 2, 3, \ldots, n_i. \]

If \( W_{i_{\text{th}}}^{(k)} \leq W_i < (W_{i_{\text{th}}}^{(k)} + W_{i_{\text{th}}}^{(k+1)})/2 \), then \( W_i \) will be fixed to \( W_{i_{\text{th}}}^{(k+1)} \). Mathematically, this can be expressed as follows:

\[ W_i = W_{i_{\text{th}}}^{(k+1)} \quad \text{if} \quad W_{i_{\text{th}}}^{(k)} < W_i < (W_{i_{\text{th}}}^{(k)} + W_{i_{\text{th}}}^{(k+1)})/2, \quad (12) \]

\[ k = 2, 3, \ldots, n_i. \]

### 2.3. Calculation of slack generator power output

Without transmission loss:

\[ W_N = W_D - \sum_{i=1}^{(N-1)} W_i. \quad (13) \]

With transmission loss:

\[ W_N = W_D + W_L - \sum_{i=1}^{N-1} W_i. \quad (14) \]

By using Eqs. (6) and (14), the modified equation may be written as follows:
\[ B_{NN}W_N^2 + W_N \left( 2 \sum_{i=1}^{N-1} B_{Ni}W_i + B_{ON} - 1 \right) + \left( W_D + \sum_{i=1}^{N-1} \sum_{j=1}^{N-1} W_iB_{ij}W_j \right) + \sum_{i=1}^{N-1} B_{Ni}W_i - \sum_{i=1}^{N-1} W_i + B_{00} = 0. \tag{15} \]

\( W_N \) is the same as that mentioned in [25].

3. Symbiotic Organisms Search (SOS)

SOS is developed based on the relationship between two separate organisms. The word symbiosis represents the relationship between two specific species. Among the various advantageous relations found in nature, mutualism, commensalism, and parasitism [44] are the most widely recognized. Mutualism is a symbiotic relationship between two separate organisms, wherein both organisms get the advantage [44]. In commensalism between two organisms, one is benefited by the other without affecting it [44]. Parasitism is the relationship between two types of organisms in which one gets benefitted while harming the other [44]. SOS imitates the mutualism, commensalism, and parasitism phases of nature in order to generate a new solution.

3.1. Mutualism phase

The relationship between flowers and bees is an example of mutualism. Honeybees accumulate nectar flying from flower to flower and transform it into honey. In this process, honeybees circulate pollen, which inspires fertilization. Thus, both the honeybee and the flowers get benefitted.

In SOS, \( Z_i \) stands for an organism related to the \( i \)th individual from the eco-system. From the eco-system, another organism \( Z_j \) is then chosen randomly to interact with \( Z_i \). These two organisms are then made and involved in a mutualistic association, such that both \( Z_i \) and \( Z_j \) get benefitted. The new candidate solutions \( Z_{i\text{new}} \) and \( Z_{j\text{new}} \) are computed in light of the mutually beneficial interaction between \( Z_i \) and \( Z_j \). This is presented in Eqs. (16) and (17):

\[ Z_{i\text{new}} = Z_i + \text{rand} (0, 1) \times (Z_{\text{best}} - \text{MutualVector} \times b_{f_1}). \tag{16} \]

\[ Z_{j\text{new}} = Z_j + \text{rand} (0, 1) \times (Z_{\text{best}} - \text{MutualVector} \times b_{f_2}). \tag{17} \]

\[ \text{MutualVector} = \frac{Z_i + Z_j}{2}. \tag{18} \]

where \( b_{f_1} \) and \( b_{f_2} \) are the benefit factors. \( Z_{i\text{new}} \) and \( Z_{j\text{new}} \) are the new candidate solutions obtained after modifying the values of \( Z_i \) and \( Z_j \) in the mutualism phase. \( Z_{\text{best}} \) represents the best organism obtained so far among all sets in the population matrix (Ecoystem) based on the fitness value. Sometimes, in some mutualism connections, one organism may get benefitted to a large extent, while the other may have just satisfactory benefit. Thus, organism \( Z_i \) may get more significant advantage than \( Z_j \) when interaction occurs between the two. Here, benefit elements \( (b_{f_1} \) and \( b_{f_2} \) are resolved arbitrarily as either 1 or 2.

3.2. Commensalism phase

An example of the commensalism relationship is the relation between sharks and remora fish. The remora eats sustenance remains by attaching itself to the shark and, thus, gets benefitted. However, this does not affect the shark. Thus, the shark gets negligible, if any, benefit from the relations.

Similar to the mutualism stage, an organism, \( Z_j \), is chosen arbitrarily from the eco-system to associate with \( Z_i \). Here, organism \( Z_j \) gets benefit from the association, while organism \( Z_j \) neither gets benefitted nor hurt from the relation. Candidate solution \( Z_j \) is obtained by a commensal advantageous interaction between organisms \( Z_i \) and \( Z_j \), as given in Eq. (18). Organism \( Z_j \) is updated if a new fitness function value is superior to the pre-interaction fitness function value.

\[ Z_{i\text{new}} = Z_i + \text{rand} (-1, 1) \times (Z_{\text{best}} - Z_j). \tag{19} \]

3.3. Parasitism phase

Parasitism is a relationship between two organisms in which one gets benefit at the cost of harming the other. In Parasitism, the parasite gets benefit while the host gets harmful affect by the relationship. The tapeworms are divided flatworms, which are found in the inner parts of the entrails of creatures such as bovines, pigs; people may be taken as an example of parasitism. Here, the flatworms are the parasites, which eat the host’s partly digested sustenance and, thus, get benefitted, whereas the host is affected by being deprived of the supplements. In SOS algorithm, organism \( Z_i \) acts as the tapeworm by creating an artificial parasite known as “parasite vector”. In the search space, parasite vector can be obtained by copying organism \( Z_i \) and, then, modifying the randomly chosen dimensions with a random number. From the eco-system organic entity, \( Z_i \) is chosen arbitrarily and serves as a parasite host. If the fitness function value of the parasite vector is superior, organism \( Z_j \) will be murdered by it. Thus, the parasite vector will acquire the place in the eco-system. If \( Z_j \) is superior, then \( Z_j \) will resist the parasite and, hence, the parasite will never again be able to sustain in that eco-system.
4. Oppositional Based Learning (OBL)

Tizhooosh proposed the application of the OBL [47] technique to enhance computational speed and accelerate the rate of the convergence of various optimization algorithms.

If \( y \) is the real number between \([pq, pr]\), then the opposite number of \( y \) may be expressed as follows:

\[
y_0 = pq + pr - y.
\]

(20)

where \( y_0 \) is the opposite number of \( y \).

If \( y \) is a real number between \([pq, pr]\), then the quasi-point, \( y_{q0} \), can be expressed as follows:

\[
y_{q0} = m \cdot d (pc, y_0).
\]

(21)

where \( pc \) represents the midpoint of the interval \([pq, pr]\); rand \((pc, x_0)\) is a random number distributed uniformly between \( pc \) and \( y_0 \). A similar logic may be applied for reflecting the quasi-opposite point \( y_{qr} \). If \( y \) is a real number between \([pq, pr]\), then the value of \( y_{qr} \) can be defined as follows:

\[
y_{qr} = m \cdot d (pc, y).
\]

(22)

where \( y_{qr} \) is the quasi-reflected point.

The above-mentioned definitions can reach out to larger dimensions without much of a stretch.

4.1. Implementation of OBL in SOS algorithm

The OBL technique is implemented in SOS algorithm in order to accelerate the convergence speed of SOS.

4.1.1. Algorithm for the quasi-reflection-based initialization

The steps are given below:

1. Randomly generate initial population \( U \) between maximum and minimum limits of decision variables and generate a reflection weight \( \mu \) between \([0, 1]\);
2. Generate Quasi-Reflected Sets (QRSs) for each initially generated population set, \( U \), using the following procedure;
3. For \( e = 1 : A \) (\( A = \text{pop set} \))
4. For \( f = 1 : B \) (\( B = \text{decision variable} \))
5. If \( U_{e,f} < \text{Median} \)
6. \( \text{QRS}_{e,f} = U_{e,f} + (\text{Median} - U_{e,f}) \times \mu \)\%
(\( \text{Median} = U_{e,f} = (pq + pr)/2 \))
7. Else
8. \( \text{QRS}_{e,f} = \text{Median} + (U_{e,f} - \text{Median}) \times \mu \)
9. End
10. End
11. End
12. Evaluate fitness value for QRS and total population
13. Sort out best \( A \) individuals on the basis of their fitness
14. Store the best population sets

4.1.2. Effect of jumping rate

QRSs have been used in SOS algorithm for accelerating the convergence speed. However, it has been found that if QRSs are generated in every step, the simulation time may be prolonged. Therefore, in order to optimize the computational time, a control parameter called jumping rate has been used. It is a control parameter whose value is set by the user in order to skip the creation of the quasi-reflected set in certain generation. The effect of this parameter in QOSOS algorithm is explained in the flow chart of Section 4.2.

1. For \( f = 1 : B \) (\( B = \text{decision variable} \));
2. If \( U_{e,f} < \text{Median} \)
3. \( \text{QRS}_{e,f} = U_{e,f} + (\text{Median} - U_{e,f}) \times \mu \)\%
(\( \text{Median} = U_{e,f} = (pq + pr)/2 \));
4. Else
5. \( \text{QRS}_{e,f} = \text{Median} + (U_{e,f} - \text{Median}) \times \mu \)
6. End
7. End
8. End
9. Evaluate fitness value for QRS if selected by \( J_r \)
10. End

4.2. Application of oppositional symbiotic organism search algorithm in ELD

The flow chart of QOSOS algorithm is described in Figure 1, which shows the application of QOSOS algorithm in ELD problems. The steps of the QOSOS method applied to economic load dispatch problems are given below:

Step 1: Initialize the number of generators, cosize \( n \), maximum fitness evaluation \( \text{max_fit} \), initial population counter \( \text{num_iter} = 0 \), initial number of fitness evaluations \( \text{num_fit_eval} = 0 \) and maximum number of iterations \( \text{max_iter} \), etc.;

Step 2: Generate output of \((N - 1)\) number of thermal generators randomly based on cosize and dimension of the problem. These are called decision variables:

\[
Z_j^i = (Z_j^\text{max} - Z_j^\text{min}) \times \text{rand} + Z_j^\text{min}
\]

for \( j = 1, 2, 3 \cdots N - 1 \).
Figure 1. Flow chart of Quasi Oppositional Symbiotic Organisms Search (QOSOS) applied to ELD.
**Figure 1.** Flow chart of Quasi Oppositional Symbiotic Organisms Search (QOSOS) applied to ELD (continued).
The initial population matrix (ecosystem) is presented below:

\[
Ecosystem = \begin{bmatrix}
Z_{11} & Z_{12} & Z_{13} & \cdots & Z_{1n} \\
Z_{21} & Z_{22} & Z_{23} & \cdots & Z_{2n} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
Z_{n1} & Z_{n2} & Z_{n3} & \cdots & Z_{nn}
\end{bmatrix}
\]

**Step 3:** Calculate the real power generation for slack generator. Verify real power balance constraint given in Eq. (5) for slack generator. If the output of slack generator does not meet generation operating limit constraint and other constraints mentioned in Eqs. (6) and (12), then go to Step 2; otherwise, go to Step 4;

**Step 4:** Calculate the objective function;

**Step 5:** Generate the quasi-reflected set of the population matrix using Eqs. (20)–(22);

**Step 6:** Check the limit of all constraints using Eqs. (5) and (12). If the constraints are satisfied, then go to Step 7; otherwise, go to Step 5;

**Step 7:** Calculate the fitness function (fitness 1) using the quasi-reflected set;

**Step 8:** Based on the fitness value, sort out the best set of the initial population matrix (ecosystem);

**Step 9:** Update the set in various phases of SOS (mutualism, commensalism, and parasitism).

- **Mutualism phase**
  1. Choose one organism \( Z_j \) randomly from ecosystem \( (Z_j \neq Z_i) \);
  2. Determine mutual vector using Eq. (18) and benefit_factor;
  3. Modify the organisms \( Z_j \) and \( Z_i \) (\( Z_i \) stands for an organism related to the \( i \)th individual from the eco-system) on the basis of their mutual relationship in Eqs. (16) and (17);
  4. Check the limits of modified sets \( Z_{\text{new}} \) and \( Z_{j\text{new}} \). If any organism violates either upper or lower limit, then fix it to the respective limit and calculate the real power generation for slack generator. If the output of slack generator does not meet the generator’s operating limit constraint, then go to step (i); otherwise, go to step (iv);
  5. Calculate the fitness values \( f(Z_{\text{new}}) \) for \( Z_{\text{new}} \) and \( f(Z_{j\text{new}}) \) for \( Z_{j\text{new}} \) and check whether the modified sets are fitter than the previous set or not. If the modified set is fitter than the previous set, then accept it. Otherwise, reject this set and keep the previous set. The steps are given below:
    1. If \( f(Z_{\text{new}}) < \text{fitness}\, 1 \):
    2. If \( f(Z_{j\text{new}}) < \text{fitness}\, 1 \):
    3. \( Z_{\text{ecosystem}} = Z_{\text{new}} \);
    4. End;
  6. If \( f(Z_{j\text{new}}) < \text{fitness}\, 1 \):
  7. \( Z_{\text{ecosystem}} = Z_{j\text{new}} \);
  8. End.

- **Commensalism phase**
  1. Choose one organism \( Z_j \) randomly along with \( Z_i \);
  2. Modify \( Z_i \) using Eq. (19);
  3. Check the limit of all constraints. If any organism violates any constraint limit, then fix it to the respective limit and calculate the real power generation for slack generator. If the output of slack generator does not meet the generator’s operating limit constraint, then go to step (i); otherwise, proceed to step (iv);
  4. Calculate the fitness function value \( f(Z_{\text{new}}) \);
  5. If the modified set \( Z_{\text{new}} \) is fitter than the previous set, then accept the modified set to replace the previous one. Otherwise, reject the set and keep the previous one. The steps are given below:
    1. If \( f(Z_{\text{new}}) < \text{fitness}\, 1 \):
    2. \( \text{fitness}\, 1 = f(Z_{\text{new}}) \):
    3. \( Z_{\text{ecosystem}} = Z_{\text{new}} \);
    4. End.

- **Parasitism phase**
  1. Select one organism randomly and generate parasite_vector by mutating \( Z_i \) by means of a random number in between upper and lower bounds;
  2. Check the constraints of parasite_vector. If any organism violates either upper or lower limit, then set it to the respective limit and calculate the real power generation for slack generator. If output of slack generator does not satisfy the generator’s operating limit constraint, then go back to step (i); otherwise, go to step (iii);
  3. Calculate the fitness function \( f(\text{parasite}_\text{vector}) \);
  4. If the modified set is fitter than the previous set, then accept the modified set. Otherwise, reject the modified set and keep the previous one. The steps are given below:
    1. If \( f(\text{parasite}_\text{vector}) < \text{fitness}\, 1 \):
    2. \( \text{fitness}\, 1 = f(\text{parasite}_\text{vector}) \):
    3. \( Z_{\text{ecosystem}} = \text{parasite}_\text{vector} \);
    4. End.
| Units   | DE/BBO [25] | BBO [24] | GAAPI [42] | SOH-PSO [42] | EMA [42] | SOS | QOSOS |
|---------|-------------|----------|------------|--------------|----------|-----|-------|
| $P_1$ (MW) | 425.815607  | 455.000000 | 451.70     | 455.00       | 455.00   | 455.00   | 455.00 |
| $P_2$ (MW) | 419.88952   | 420.000000 | 380.00     | 380.00       | 380.00   | 380.00   | 380.00 |
| $P_3$ (MW) | 130.000000  | 130.000000 | 130.00     | 130.00       | 130.00   | 130.00   | 130.00 |
| $P_4$ (MW) | 127.109310  | 130.000000 | 129.53     | 130.00       | 130.00   | 130.00   | 130.00 |
| $P_5$ (MW) | 269.869956  | 270.000000 | 170.00     | 170.00       | 170.00   | 170.00   | 170.00 |
| $P_6$ (MW) | 459.155633  | 460.000000 | 469.96     | 460.00       | 460.00   | 460.00   | 460.00 |
| $P_7$ (MW) | 429.033732  | 430.000000 | 429.71     | 430.00       | 430.00   | 430.00   | 430.00 |
| $P_8$ (MW) | 69.906161   | 64.978264  | 75.35      | 117.53       | 72.0145  | 72.0979  | 71.6926 |
| $P_9$ (MW) | 58.753444   | 47.684539  | 34.96      | 77.90        | 58.6212  | 58.431100| 58.834260 |
| $P_{10}$ (MW) | 80.549854  | 48.809702  | 160.00     | 119.54       | 160.00   | 129.9989 | 160.00 |
| $P_{11}$ (MW) | 47.210660  | 59.049411  | 79.75      | 54.50        | 80.0000  | 80.0000  | 80.0000 |
| $P_{12}$ (MW) | 73.167992  | 55.000000  | 80.00      | 80.00        | 80.0000  | 80.0000  | 80.0000 |
| $P_{13}$ (MW) | 27.608892  | 26.853800  | 34.21      | 25.00        | 25.0000  | 25.0000  | 25.0000 |
| $P_{14}$ (MW) | 15.494490  | 22.765547  | 21.14      | 17.86        | 15.0000  | 15.0000  | 15.0000 |
| $P_{15}$ (MW) | 24.922388  | 36.935999  | 21.02      | 15.00        | 15.0000  | 15.0000  | 15.0000 |
| Total power (MW) | 2658.07018  | 2657.155242 | 2660.36 | 2662.29 | 2660.6626 | 2660.5288 | 2660.5270 |
| Power loss (MW) | 28.0702 | 27.15521443 | 30.36 | 32.28 | 30.6626 | 30.5288 | 30.5270 |
| Fuel cost ($/h) | 3270.70296 | 3271.23959 | 3273.295 | 3275.139 | 32704.4503 | 32702.9358 | 32702.9352 |

**Step 10:** Save the modified set;  
**Step 11:** Generate random numbers;  
**Step 12:** Check whether the jumping rate is greater than the random number or not. If it is greater than the random number, then go to Step 13; otherwise, go to Step 17;  
**Step 13:** Using the modified set, generate the quasi-reflected set;  
**Step 14:** Check the limit of all constraints. If all constraints are satisfied, then go to Step 15; otherwise, go to Step 13;  
**Step 15:** Calculate the fitness function using the quasi-reflected set;  
**Step 16:** Sort out the best set between the modified and QRoS;  
**Step 17:** If the quasi-reflected set is fitter than the modified set, then accept this set. Otherwise, reject the quasi-reflected set and keep the modified one;  
**Step 18:** Determine the best fitness and best organism;  
**Step 19:** Go to Step 9 and repeat until the predefined max FE is reached.

## 5. Results and discussions

The QOSOS algorithm is applied to different test cases of ELD problem, and its performance is compared to various algorithms available in the literature. This algorithm has been coded in Matlab 9 with a computer equipped with 2.40 GHz core i3 to execute the program.

### 5.1. Description of test systems

#### 5.1.1. Test system I

A 15-generator system with a system demand of 2630 MW is considered here. The transmission loss, prohibited operating zone, and ramp rate limit constraints are included in this case. The input data are available in [42]. The results obtained by QOSOS, SOS, DE/BBO [25], BBO [24], GAAPI [42], SOH-PSO [42], and EMA [42] are displayed in Table 1. The fuel price convergence curve is presented in Figure 2. Best, worst, and average fuel costs achieved

![Figure 2](image.png)
Table 2. Performance analysis of different methods taken after 50 try-outs.

| Methods   | Generation cost ($/h) | Time/iteration, (s) | No of hits to minimum solution |
|-----------|-----------------------|---------------------|---------------------------------|
|           | Max.                  | Min.                | Average                         |                               |
| QOSOS     | 32705.5912            | 32702.9352          | 32703.0414                      | 2.1                           | 48                             |
| SOS       | 32705.5915            | 32702.9358          | 32703.0420                      | 3                             | 48                             |
| BBO [24]  | 32713.4991            | 32712.3959          | 32712.5282-384                  | 17.5                          | 44                             |
| DE/BBO [25]| 32710.2306            | 32707.0296          | 32707.2864                      | 12.4                          | 46                             |
| MDE [12]  | 33245.54              | 32917.87            | 33066.76                        | NA                            | NA                             |
| PSO [42]  | 33031                 | 32858               | 32989                           | NA                            | NA                             |
| ABC [12]  | 32708.27              | 32707.85            | 32707.95                        | NA                            | NA                             |
| PSO-SIF [12]| 32709.92             | 32706.8800          | 32707.7900                      | NA                            | NA                             |
| θ-PSO [42] | 32744.0306            | 32706.6856          | 32711.4955                      | NA                            | NA                             |
| EMA [12]  | 32704.4506            | 32704.4503          | 32704.4504                      | NA                            | NA                             |
| RCCRO* [27]| 32698.9950329897      | 32698.995032997     | 32698.99503298                   | 4                             | 50                             |
| IA_EDP* [49]| 32823.7590           | 32698.2038          | 32750.2176                      | NA                            | NA                             |

*: Infeasible Solution.

by QOSOS, SOS, DE/BBO [25], MDE [42], PSO [42], BBO [24], ABC [42], PSO [42], PSO-SIF [42], θ-PSO [42], EMA [42], RCCRO [27], and IA_EDP [49] over 50 trails are presented in Table 2.

5.1.2. Test system II

A 40-unit system with a load demand of 10500 MW is considered here. The valve-point effect and transmission loss are also considered in this case. The input data are available in [28]. The β-loss coefficients of the transmission losses are obtained from the 6-generator system [50] by multiplying rows and columns up to 40 numbers of units. Simulation results of QOSOS method are compared to SOS, BBO, DE/BBO, SDE [51], and GAAPI [52] algorithms. Fuel price convergence curve for QOSOS and SOS is shown in Figure 3. The best results of these methods are displayed in Table 3. Mean, best, and worst fuel costs are obtained by QOSOS, SOS, and ORCCRO [28]. BBO, DE/BBO over 50 try-outs are displayed in Table 4.

5.1.3. Test system III

A-110 unit with a system demand of 15000 MW is considered here. The fuel price curve is quadratic in nature. The input data are available in [37]. The best results obtained by the proposed QOSOS and SOS methods are presented in Tables 5 and 6, respectively. The fuel price convergence curve is presented in Figure 4. Mean, average, and worst fuel prices achieved by QOSOS, SOS, OIWO [37], ORCCRO [28], SAB [37], SAF [37], SA [37], BBO [28], and DE/BBO [28] over 50 try-outs are shown in Table 7.

5.1.4. Test system IV

Here, a 160-unit system with multiple fuel options is considered. The input data are available in [37]. The transmission loss has not been considered here. The system demand is 43200 MW. The best results achieved by QOSOS and SOS method are shown in Tables 8 and 9, respectively. The best, average, and maximum fuel costs obtained by various methods

![Figure 3](image1.png)

**Figure 3.** Convergence characteristics of the 40-generator system with loss obtained by Quasi Oppositional Symbiotic Organisms Search (QOSOS) and Symbiotic Organisms Search (SOS).

![Figure 4](image2.png)

**Figure 4.** Convergence characteristics of the 110-generator system obtained by Quasi Oppositional Symbiotic Organisms Search (QOSOS) and Symbiotic Organisms Search (SOS).
| Unit     | Power outputs (MW) |
|----------|--------------------|
|          | QOSOS | SDE [51] | GAAP [52] | BBO | DE/BBO |
| $P_1$ (MW) | 113.998208 | 110.06 | 114 | 112.541727 | 111.010874 |
| $P_2$ (MW) | 113.990687 | 112.41 | 114 | 113.215833 | 113.707080 |
| $P_3$ (MW) | 119.999761 | 120.00 | 120.00 | 119.506795 | 118.639216 |
| $P_4$ (MW) | 190.000000 | 188.72 | 190 | 188.371629 | 189.492251 |
| $P_5$ (MW) | 96.996697 | 85.91 | 97 | 90.412350 | 86.322613 |
| $P_6$ (MW) | 140.000000 | 140.00 | 140.00 | 139.048610 | 139.874742 |
| $P_7$ (MW) | 299.999670 | 250.19 | 300 | 294.971421 | 299.863328 |
| $P_8$ (MW) | 300.000000 | 290.68 | 300 | 299.181239 | 285.420119 |
| $P_9$ (MW) | 299.999941 | 300 | 300 | 296.464436 | 296.289081 |
| $P_{10}$ (MW) | 279.598777 | 282.01 | 285.25 | 279.885675 | 285.071561 |
| $P_{11}$ (MW) | 168.802970 | 180.82 | 226.3 | 160.149083 | 164.609461 |
| $P_{12}$ (MW) | 94.001371 | 168.74 | 204.72 | 96.736599 | 94.000727 |
| $P_{13}$ (MW) | 484.031540 | 469.96 | 346.48 | 484.043477 | 486.304449 |
| $P_{14}$ (MW) | 484.040027 | 484.17 | 434.32 | 483.316174 | 480.659779 |
| $P_{15}$ (MW) | 484.056224 | 487.73 | 431.34 | 483.766660 | 480.651462 |
| $P_{16}$ (MW) | 484.041270 | 482.30 | 440.22 | 483.299544 | 485.019668 |
| $P_{17}$ (MW) | 489.278603 | 499.64 | 500 | 490.831472 | 487.942092 |
| $P_{18}$ (MW) | 489.280840 | 411.32 | 500 | 492.185686 | 491.080800 |
| $P_{19}$ (MW) | 511.288305 | 510.47 | 550 | 511.281207 | 511.789157 |
| $P_{20}$ (MW) | 511.293735 | 542.04 | 550 | 521.551143 | 544.886056 |
| $P_{21}$ (MW) | 526.286092 | 544.81 | 550 | 526.424242 | 528.929799 |
| $P_{22}$ (MW) | 550.000000 | 550.00 | 550 | 538.296144 | 540.587228 |
| $P_{23}$ (MW) | 523.298180 | 550.00 | 550 | 534.744028 | 524.983999 |
| $P_{24}$ (MW) | 523.266462 | 528.16 | 550 | 521.195398 | 524.119592 |
| $P_{25}$ (MW) | 524.774829 | 524.16 | 550 | 526.144729 | 534.491096 |
| $P_{26}$ (MW) | 523.376004 | 539.10 | 550 | 544.438777 | 529.147514 |
| $P_{27}$ (MW) | 10.030764 | 10.00 | 11.44 | 11.305008 | 10.709604 |
| $P_{28}$ (MW) | 10.081928 | 10.37 | 11.56 | 10.209941 | 10.000000 |
| $P_{29}$ (MW) | 10.002668 | 10.00 | 11.42 | 10.713711 | 10.000314 |
| $P_{30}$ (MW) | 87.851470 | 96.10 | 97 | 88.27085 | 90.062133 |
| $P_{31}$ (MW) | 190.000000 | 185.33 | 190 | 189.843396 | 189.814900 |
| $P_{32}$ (MW) | 189.999971 | 189.54 | 190 | 189.935000 | 187.686113 |
| $P_{33}$ (MW) | 190.000000 | 189.96 | 190 | 189.128321 | 189.909857 |
| $P_{34}$ (MW) | 199.999915 | 199.90 | 200 | 198.066854 | 199.833444 |
| $P_{35}$ (MW) | 199.999994 | 196.25 | 200 | 199.916450 | 199.926719 |
| $P_{36}$ (MW) | 164.818600 | 185.85 | 200 | 194.351445 | 163.038804 |
| $P_{37}$ (MW) | 109.999979 | 109.72 | 110 | 109.429766 | 109.847519 |
| $P_{38}$ (MW) | 109.999982 | 110.00 | 110 | 109.558632 | 109.203936 |
| $P_{39}$ (MW) | 109.999917 | 95.71 | 110 | 109.621734 | 109.596388 |
| $P_{40}$ (MW) | 549.999929 | 532.47 | 550 | 527.819702 | 543.227545 |

**Total power (MW)** 11458.484603

**Power loss (MW)** 995.484602

**Fuel cost ($/hr)** 136452.357142
such as ORCCRO [28], BBO [28], DE/BBO [37], EDE-DE [37], IGA-MU [37], CGA-MU [37], OIWO [37], SOS, and proposed QOSOS methods are presented in Table 10. The fuel price convergence curve for the 160-generator system is shown in Figure 5.

### 5.2. Parameter tuning

To check the impact of jumping rate on QOSOS algorithm, four test systems are taken and the program is executed for 50 individual trials for each system. The value of jumping rate varied from 0.1-0.9. The results obtained by QOSOS algorithm are shown in Table 11. From this table, it is found that when the value of jumping rate is 0.4, then the cost obtained by QOSOS algorithm is minimum for all systems. No changes are found when the value of jumping rate is above or below 0.4.

### 5.3. Comparative study

#### 5.3.1. Quality of solution

In this paper, four test systems have been considered in order to investigate the solution quality of this proposed method. The best result achieved by QOSOS method is presented in Tables 1, 3, 5, and 8. The best

### Table 4. Performance analysis of different methods taken after 50 try-outs.

| Methods | Generation cost ($/hr) | Time/iteration (sec) | No. of hits to min. solution |
|---------|-------------------------|----------------------|-----------------------------|
| QOSOS   | Max. 136452.357142      | Min. 136452.357142   | Average 136452.357142       | 0.05 50 |
| SOS     | Max. 136464.264253      | Min. 136464.264253   | Average 136464.264253       | 0.064 50 |
| ORCCRO [28] | Max. 136855.190000   | Min. 136855.190000   | Average 136855.190000       | 0.07 50 |
| BBO     | Max. 137587.823244      | Min. 137026.823244   | Average 137116.583244       | 0.30 41 |
| DE/BBO  | Max. 137150.766993      | Min. 136950.766993   | Average 136966.766993       | 0.16 45 |

### Table 5. Power output for Test system III against minimum fuel price using Quasi Oppositional Symbiotic Organisms Search (QOSOS) algorithm (PD = 15000 MW).

| Units      | Power outputs (MW) | Units      | Power outputs (MW) | Units      | Power outputs (MW) | Units      | Power outputs (MW) | Units      | Power outputs (MW) |
|------------|--------------------|------------|--------------------|------------|--------------------|------------|--------------------|------------|--------------------|
| 1           | 2.400260           | 21         | 68.900000          | 41         | 157.397800         | 61         | 45.013730          | 81         | 10.000000          |
| 2           | 2.400033           | 22         | 68.901740          | 42         | 220.000000         | 62         | 45.000000          | 82         | 12.443004          |
| 3           | 2.401600           | 23         | 68.900000          | 43         | 439.991800         | 63         | 184.977100         | 83         | 20.001900          |
| 4           | 2.400001           | 24         | 349.998700         | 44         | 559.997200         | 64         | 184.980500         | 84         | 199.895100         |
| 5           | 2.400000           | 25         | 400.000000         | 45         | 660.000000         | 65         | 184.968200         | 85         | 324.936800         |
| 6           | 4.000099           | 26         | 399.999400         | 46         | 615.208600         | 66         | 184.892900         | 86         | 439.984800         |
| 7           | 4.006499           | 27         | 499.999900         | 47         | 5.4001605          | 67         | 70.403840          | 87         | 21.870970          |
| 8           | 4.001311           | 28         | 499.996400         | 48         | 5.4014250          | 68         | 70.003120          | 88         | 20.892530          |
| 9           | 4.000657           | 29         | 199.989700         | 49         | 8.400334           | 69         | 70.000000          | 89         | 77.273580          |
| 10          | 6.251800           | 30         | 99.975330          | 50         | 8.400000           | 70         | 399.999800         | 90         | 83.322540          |
| 11          | 60.19975           | 31         | 10.001320          | 51         | 8.401234           | 71         | 399.999210         | 91         | 38.117670          |
| 12          | 35.768520          | 32         | 19.999700          | 52         | 12.000035          | 72         | 399.999830         | 92         | 96.862830          |
| 13          | 54.903420          | 33         | 79.994530          | 53         | 12.000130          | 73         | 103.998400         | 93         | 439.998800         |
| 14          | 25.008950          | 34         | 249.978000         | 54         | 12.000000          | 74         | 190.713000         | 94         | 500.000000         |
| 15          | 25.001280          | 35         | 359.973400         | 55         | 12.000000          | 75         | 89.816370          | 95         | 399.997800         |
| 16          | 25.000960          | 36         | 399.991400         | 56         | 25.446850          | 76         | 49.933960          | 96         | 471.430300         |
| 17          | 154.996900         | 37         | 40.000000          | 57         | 25.450350          | 77         | 160.005700         | 97         | 3.600000           |
| 18          | 154.997600         | 38         | 69.920000          | 58         | 35.008500          | 78         | 289.383700         | 98         | 3.600013           |
| 19          | 154.996400         | 39         | 99.992710          | 59         | 39.172200          | 79         | 172.673200         | 99         | 4.400455           |
| 20          | 154.968600         | 40         | 119.994100         | 60         | 45.000000          | 80         | 118.916100         | 100        | 4.402490           |

**Fuel cost ($/h)** 197939.7436
Table 6. Power output for Test system III against minimum fuel price using Symbiotic Organisms Search (SOS) algorithm (PD = 15000 MW).

| Units | Power outputs (MW) | Units | Power outputs (MW) | Units | Power outputs (MW) | Units | Power outputs (MW) | Units | Power outputs (MW) |
|-------|------------------|-------|------------------|-------|------------------|-------|------------------|-------|------------------|
| 1     | 2.404854         | 21    | 68.900240        | 41    | 150.077500       | 61    | 45.001110        | 81    | 10.000000        |
| 2     | 2.409338         | 22    | 68.938320        | 42    | 219.979000       | 62    | 45.004710        | 82    | 12.000000        |
| 3     | 2.400179         | 23    | 68.999420        | 43    | 439.999000       | 63    | 184.999400       | 83    | 20.000000        |
| 4     | 2.400021         | 24    | 349.999300       | 44    | 559.998700       | 64    | 184.999800       | 84    | 199.984000       |
| 5     | 2.400092         | 25    | 399.999900       | 45    | 660.000000       | 65    | 184.999380       | 85    | 319.204100       |
| 6     | 4.000000         | 26    | 399.999300       | 46    | 618.452400       | 66    | 184.999800       | 86    | 439.996600       |
| 7     | 4.000000         | 27    | 499.998400       | 47    | 5.400033         | 67    | 70.000020        | 87    | 10.026810        |
| 6     | 4.000000         | 28    | 500.000000       | 48    | 5.400037         | 68    | 70.000000        | 88    | 54.874640        |
| 9     | 4.000000         | 29    | 199.996800       | 49    | 8.400010         | 69    | 70.000020        | 89    | 87.245410        |
| 10    | 66.101900        | 30    | 99.997130        | 50    | 8.400016         | 70    | 359.196700       | 90    | 79.516720        |
| 11    | 58.838280        | 31    | 10.000010        | 51    | 8.486224         | 71    | 399.993500       | 91    | 51.425710        |
| 12    | 31.808300        | 32    | 19.980380        | 52    | 12.000000        | 72    | 399.999800       | 92    | 89.335600        |
| 13    | 50.054110        | 33    | 78.049800        | 53    | 12.000070        | 73    | 107.233000       | 93    | 439.981600       |
| 14    | 25.001380        | 34    | 219.997800       | 54    | 12.000000        | 74    | 184.549600       | 94    | 499.997100       |
| 15    | 25.000290        | 35    | 359.982400       | 55    | 12.000060        | 75    | 89.961320        | 95    | 599.987800       |
| 16    | 25.000380        | 36    | 399.966000       | 56    | 25.692140        | 76    | 50.000000        | 96    | 465.698500       |
| 17    | 154.983900       | 37    | 399.993200       | 57    | 25.209730        | 77    | 160.017300       | 97    | 3.600011         |
| 18    | 155.000000       | 38    | 69.999360        | 58    | 35.003600        | 78    | 302.931700       | 98    | 3.600000         |
| 19    | 154.994000       | 39    | 99.998330        | 59    | 35.000060        | 79    | 180.131500       | 99    | 4.404125         |
| 20    | 154.937100       | 40    | 117.832300       | 60    | 45.003020        | 80    | 119.966400       | 100   | 4.400107         |

| Methods | Generation cost ($/h) | Time (s) | No of hits to minimum solution |
|---------|-----------------------|----------|-------------------------------|
| QOSOS   | 197939.7436           | 24       | 50                            |
| SOS     | 197944.835            | 27       | 48                            |
| OIWO [37]| 197989.93            | 45       | 48                            |
| ORCCRO [28]| 198016.89         | 46       | NA                            |
| SAB [37]| NA                   | NA       | NA                            |
| SAF [37]| NA                   | NA       | NA                            |
| SA [37]| NA                   | NA       | NA                            |
| BBO [28]| 199102.59            | 115      | 41                            |
| DE/BBO [28]| 198828.57        | 132      | 43                            |

fuel costs obtained by QOSOS and other optimization techniques are shown in Figure 6. Minimum, worst, and mean values of different algorithms are presented in Tables 2, 4, 7, and 10. From these tables, it is found that the cost obtained by QOSOS technique is better than that by other well-known optimization methods such as RCCRO, BBO, DE/BBO, SOS, OIWO, and so on. For example, in a 100-unit system, the fuel cost obtained by QOSOS method is less (9964.7743 $/h) than that by other well-known algorithms like SOS (9964.1983 $/h), DE/BBO (10007.05 $/h), BBO (10008.71 $/h), CGA-MU (10143.73 $/h), ED-DE
Table 8. Power output for Test system IV against minimum fuel price using Quasi Oppositional Symbiotic Organisms Search (QOSOS) algorithm \((FD = 43200 \text{ MW})\).

| Units | Power outputs (MW) | Units | Power outputs (MW) | Units | Power outputs (MW) | Units | Power outputs (MW) | Units | Power outputs (MW) |
|-------|--------------------|-------|--------------------|-------|--------------------|-------|--------------------|-------|--------------------|
| 1     | 215.4707           | 31    | 214.5395           | 61    | 214.4882           | 91    | 216.3361           | 121   | 213.4613           |
| 2     | 208.2421           | 32    | 212.9581           | 62    | 213.1896           | 92    | 213.6980           | 122   | 212.9507           |
| 3     | 270.5703           | 33    | 270.5191           | 63    | 270.5373           | 93    | 273.5828           | 123   | 274.5690           |
| 4     | 212.8673           | 34    | 210.7176           | 64    | 210.9827           | 94    | 213.6695           | 124   | 242.8669           |
| 5     | 369.5572           | 35    | 369.8329           | 65    | 269.6497           | 95    | 272.6426           | 125   | 272.5625           |
| 6     | 241.7894           | 36    | 239.6364           | 66    | 239.2367           | 96    | 241.1156           | 126   | 242.9666           |
| 7     | 290.1276           | 37    | 290.0534           | 67    | 294.7046           | 97    | 289.9545           | 127   | 295.9490           |
| 8     | 243.2671           | 38    | 239.9100           | 68    | 242.1926           | 98    | 243.3335           | 128   | 240.8942           |
| 9     | 435.7920           | 39    | 438.4512           | 69    | 436.4554           | 99    | 437.9711           | 129   | 438.9170           |
| 10    | 279.0626           | 40    | 276.4841           | 70    | 275.8674           | 100   | 280.1669           | 130   | 279.0697           |
| 11    | 213.4607           | 41    | 214.8151           | 71    | 216.5635           | 101   | 214.5162           | 131   | 212.4382           |
| 12    | 210.2358           | 42    | 213.7012           | 72    | 209.7291           | 102   | 208.7377           | 132   | 210.7201           |
| 13    | 274.6187           | 43    | 270.5417           | 73    | 274.6109           | 103   | 270.5600           | 133   | 274.6145           |
| 14    | 240.3118           | 44    | 238.5647           | 74    | 239.6399           | 104   | 239.3716           | 134   | 242.0576           |
| 15    | 272.9514           | 45    | 269.5641           | 75    | 272.6561           | 105   | 269.1597           | 135   | 269.4461           |
| 16    | 239.6383           | 46    | 242.4611           | 76    | 243.8048           | 106   | 238.6986           | 136   | 238.5637           |
| 17    | 292.4608           | 47    | 294.8516           | 77    | 292.4516           | 107   | 292.4488           | 137   | 287.6749           |
| 18    | 240.4642           | 48    | 240.5824           | 78    | 241.9238           | 108   | 242.4625           | 138   | 241.7909           |
| 19    | 437.4157           | 49    | 435.6231           | 79    | 438.7262           | 109   | 436.2056           | 139   | 431.9449           |
| 20    | 275.8047           | 50    | 278.8602           | 80    | 275.4191           | 110   | 278.7097           | 140   | 280.4117           |
| 21    | 215.4967           | 51    | 213.4739           | 81    | 215.5277           | 111   | 216.6034           | 141   | 214.4785           |
| 22    | 209.4840           | 52    | 208.4938           | 82    | 212.2087           | 112   | 210.4831           | 142   | 209.7288           |
| 23    | 277.6109           | 53    | 270.5002           | 83    | 270.6099           | 113   | 271.5742           | 143   | 274.6101           |
| 24    | 241.5208           | 54    | 242.8655           | 84    | 238.0290           | 114   | 240.1773           | 144   | 243.3230           |
| 25    | 269.1498           | 55    | 272.6260           | 85    | 269.5430           | 115   | 269.8243           | 145   | 272.1808           |
| 26    | 239.7741           | 56    | 241.5204           | 86    | 239.2385           | 116   | 240.7154           | 146   | 240.8490           |
| 27    | 290.4996           | 57    | 292.4338           | 87    | 292.3779           | 117   | 289.6335           | 147   | 292.5119           |
| 28    | 239.7753           | 58    | 242.0573           | 88    | 241.6545           | 118   | 242.0599           | 148   | 241.1172           |
| 29    | 438.6057           | 59    | 438.2665           | 89    | 438.9649           | 119   | 436.8218           | 149   | 439.8503           |
| 30    | 276.0220           | 60    | 277.0527           | 90    | 279.0049           | 120   | 275.8644           |

Fuel cost ($/h): 9964.7743

(10012.68 $/h), IGA-MU (10042.47 $/h), ORCCRO (10004.20 $/h), and OJWGO (9981.9834 $/h). The same results are found in other test systems. Therefore, it may be concluded that the performance of this QOSOS algorithm is better in terms of solution quality.

5.3.2. Robustness

The performance of QOSOS algorithm is judged after running the program for 50 numbers of trials. Out of 50 trials, QOSOS hits the minimum solution 48 times for Test system I, 50 times for Test system II, 50 times for Test system III, and 48 times for Test system IV. Therefore, the success rate of QOSOS is 96%, 100%, 100%, and 96%, respectively. However, in the case of other techniques like BBO, the success rate is 88% for Test system I, 82% for Test system 2, 82% for Test system III, and 80% for Test system IV. In the case of DE/BBO, out of 50 trials, DE/BBO hits the
Table 9. Power output for Test system IV against minimum fuel price using Symbiotic Organisms Search (SOS) algorithm (PD = 43200 MW).

| Units | Power outputs (MW) | Units | Power outputs (MW) | Units | Power outputs (MW) | Units | Power outputs (MW) | Units | Power outputs (MW) |
|-------|--------------------|-------|--------------------|-------|--------------------|-------|--------------------|-------|--------------------|
| 1     | 213.4391           | 31    | 214.5556           | 61    | 213.4332           | 91    | 214.3808           | 121   | 215.1922           |
| 2     | 210.4716           | 32    | 210.7216           | 62    | 210.9736           | 92    | 212.2070           | 122   | 212.4339           |
| 3     | 277.3943           | 33    | 272.5008           | 63    | 274.6178           | 93    | 209.5789           | 123   | 275.6178           |
| 4     | 242.0593           | 34    | 241.9270           | 64    | 238.5639           | 94    | 238.9687           | 124   | 241.7895           |
| 5     | 272.4628           | 35    | 265.4339           | 65    | 272.5994           | 95    | 273.0647           | 125   | 272.1735           |
| 6     | 242.7277           | 36    | 230.7388           | 66    | 241.2541           | 96    | 242.1924           | 126   | 241.5203           |
| 7     | 289.8278           | 37    | 290.1600           | 67    | 285.3304           | 97    | 290.1447           | 127   | 295.1800           |
| 8     | 240.3105           | 38    | 240.9832           | 68    | 241.3873           | 98    | 241.1176           | 128   | 238.8334           |
| 9     | 439.9740           | 39    | 433.5012           | 69    | 433.0181           | 99    | 436.5588           | 129   | 439.7279           |
| 10    | 269.8171           | 40    | 277.6605           | 70    | 278.8895           | 100   | 276.4331           | 130   | 277.2311           |
| 11    | 214.5256           | 41    | 212.6012           | 71    | 214.1033           | 101   | 212.3253           | 131   | 216.1747           |
|       |                    |       |                    |       |                    |       | Fuel cost ($/h):    |       | 9965.1983          |
| 12    | 209.4848           | 42    | 215.4363           | 72    | 211.9881           | 102   | 213.6914           | 132   | 210.2369           |
| 13    | 277.6397           | 43    | 273.7193           | 73    | 275.6212           | 103   | 272.7214           | 133   | 272.5977           |
| 14    | 242.3276           | 44    | 241.7895           | 74    | 239.7388           | 104   | 242.1933           | 134   | 241.3896           |
| 15    | 269.3434           | 45    | 271.6897           | 75    | 276.4731           | 105   | 271.2958           | 135   | 276.6546           |
| 16    | 241.3871           | 46    | 240.5786           | 76    | 241.5218           | 106   | 239.9127           | 136   | 243.0005           |
| 17    | 290.0881           | 47    | 296.6654           | 77    | 292.6219           | 107   | 290.2505           | 137   | 295.0211           |
| 18    | 240.9811           | 48    | 239.9064           | 78    | 241.7884           | 108   | 240.5734           | 138   | 237.7396           |
| 19    | 437.0648           | 49    | 435.8193           | 79    | 436.6464           | 109   | 433.5166           | 139   | 436.1278           |
| 20    | 277.9264           | 50    | 273.0427           | 80    | 279.0826           | 110   | 282.1311           | 140   | 276.5405           |
| 21    | 213.4619           | 51    | 214.4037           | 81    | 215.5236           | 111   | 214.4906           | 141   | 213.4624           |
| 22    | 214.9596           | 52    | 210.2307           | 82    | 214.4346           | 112   | 210.4817           | 142   | 212.7071           |
| 23    | 271.6030           | 53    | 275.6084           | 83    | 273.5992           | 113   | 271.5884           | 143   | 273.5283           |
| 24    | 241.7849           | 54    | 238.8384           | 84    | 242.3259           | 114   | 241.6547           | 144   | 240.3115           |
| 25    | 272.4845           | 55    | 272.7626           | 85    | 272.6467           | 115   | 273.1699           | 145   | 272.5002           |
| 26    | 240.7138           | 56    | 241.2563           | 86    | 242.4612           | 116   | 240.8523           | 146   | 240.8490           |
| 27    | 290.5941           | 57    | 293.8904           | 87    | 292.5437           | 117   | 288.3130           | 147   | 287.6947           |
| 28    | 238.1615           | 58    | 240.4132           | 88    | 241.5877           | 118   | 239.3601           | 148   | 239.3738           |
| 29    | 437.7040           | 59    | 434.6733           | 89    | 436.6298           | 119   | 437.9882           | 149   | 439.3068           |
| 30    | 278.8737           | 60    | 279.1716           | 90    | 279.0592           | 120   | 277.8026           | 150   | 276.5218           |

Table 10. Performance analysis of different methods taken after 50 try-outs.

| Methods      | Generation cost ($/h) | Time (s) | No of hits to minimum solution |
|--------------|-----------------------|----------|--------------------------------|
| QOSOS        | 9965.4929             | 12.8     | 48                             |
| SOS          | 9967.2648             | 14.3     | 48                             |
| DE/BBO [28]  | 100.10.36             | 35       | 42                             |
| BBO [28]     | 100.10.39             | 44       | 40                             |
| CGA-MU [37]  | N/A                   | NA       | NA                             |
| ED-DE [37]   | N/A                   | NA       | NA                             |
| IGA-MU [37]  | N/A                   | NA       | NA                             |
| ORCRO [28]   | 1004.45               | 19       | 48                             |
| OIWO [37]    | 9083.988              | 17.3     | 46                             |
minimum solution 46 times for Test system I, 45 times for Test system II, 43 times for Test system III, and 42 times for Test system IV. Therefore, the success rates of DE/BBO are 92%, 90%, 86%, and 84%, respectively. Therefore, consistency of QOSOS method is found to be more than that of many other well-known soft computing methods.

5.3.3. Computational efficiency
The main objective of OBL is to accelerate the convergence rate. It has been observed that a quasi-opposite number [48] is likely to be nearer to the solution than a random number. Therefore, the authors have applied this in SOS in order to improve the convergence rate. In Test system I, it is found that the time/iteration of QOSOS technique is only 2.1 sec, which is less than other techniques like SOS (3 sec), BBO (17.5 sec), DE/BBO (12.4 sec), and RCCRO (4 sec). Therefore, by using QOSOS method, the computational efficiency is improved by 30%, 88%, 83.06%, and 47.5%, respectively. In Test system II, the time taken by QOSOS algorithm to complete one iteration is 0.05 sec. However, the time taken by SOS, ORCCRO, BBO, and DE/BBO algorithms for the same test system is 0.064 sec, 0.07 sec, 0.2 sec, and 0.16 sec, respectively. In Test system II, the improvement rates of computational efficiency by using QOSOS technique are 21.87%, 28.57%, 75%, and 68.75%, respectively. The similar performance is observed when it is applied to large-scale power systems such as 110-unit (the improvement rates of computational efficiency by using QOSOS methods as compared to SOS, OIWO, ORCCRO, BBO, and DE/BBO algorithms are 11.11%, 22.58%, 46.66%, 79.13%, and 81.81%, respectively) and 160-unit systems (the improvement of computational efficiency by using QOSOS methods as compared to SOS, DE/BBO, BBO, ORCCRO, and OIWO methods are 10.48%, 63.42%, 70.90%, 32.63%, and 26.01%, respectively). The simulation times for all test systems are described in Tables 2, 4, 7, and 10, respectively. From these tables, it is found that the computational efficiency of QOSOS is better than that of the recently developed optimization techniques.

It is also seen that when OBL is applied in SOS algorithm, then the convergence rate becomes faster than other techniques such as SOS, BBO, DE/BBO, RCCRO, EMA, and so on. For example, in Test system I, the number of iterations required to reach minimum value is lower in the case of QOSOS (shown in Figure 2) as compared to other methods such as RCCRO, DE/BBO, and BBO [27]. From Figure 2, it is found that by applying QOSOS algorithm, the number of iterations required to reach the best solution is only 6, whereas, from Ref. [27], it is observed that the number of iterations required to reach the minimum solution is approximately 158 for RCCRO, 174 for DE/BBO, and 190 for BBO. The same results are observed in Test system II. Here, it is observed that the total iteration required to reach minimum solution is 8 in the case of QOSOS method (shown in Figure 3), which is much less than other well-known techniques such as ORCCRO, BBO, and DE/BBO [28]. From [28], it is found that the number of iterations required to reach the best solution is approximately 191 for ORCCRO, 193 for DE/BBO, and 196 for BBO.

| Jumping rate | Test system I | Test system II | Test system III | Test system IV |
|--------------|---------------|---------------|----------------|---------------|
| 0.1          | 32711.3247   | 13649.114871  | 19794.9963     | 9971.2547     |
| 0.2          | 32715.3217   | 13650.331856  | 19793.459613   | 9975.3308     |
| 0.3          | 32705.6698   | 13641.20478   | 19794.0.1236   | 9968.6450     |
| 0.4          | 32702.9352   | 13645.357142  | 19793.7436     | 9964.7743     |
| 0.5          | 32708.7741   | 13647.524650  | 19794.0213     | 9966.2739     |
| 0.6          | 32710.4796   | 13649.12387   | 19794.1302     | 9969.8802     |
| 0.7          | 32708.2200   | 13647.123652  | 19794.0147     | 9971.5014     |
| 0.8          | 32712.3382   | 13646.517895  | 19794.0396     | 9965.9871     |
| 0.9          | 32705.2030   | 13650.214789  | 19794.23887    | 9968.1882     |
Therefore, it is clear that by using the OBL technique in SOS, the convergence rate becomes much faster than other well-known optimization methods.

5.4. Statistical analysis
In recent years, various statistical methods [53, 54] have been used for performing comparisons between different algorithms. In this paper, Friedman test and Quade test are chosen to assess the performance of QOSOS algorithm statistically as compared to SOS and other well-known optimization methods. A null hypothesis ($H_0$) and an alternative hypothesis ($H_1$) are required to be defined in order to perform Friedman test and Quade test. $H_0$ denotes that there is no difference in the performance of the methods under comparison, and $H_1$ denotes that there is a difference in performance of the methods. A significance level of 5% is chosen. Table 12 describes the statistical analysis of the results obtained by QOSOS, SOS, DE/BBO, and BBO algorithms, and Table 13 describes the statistical analysis of the results obtained using QOSOS, SOS, and ORCCRO algorithms. Table 12 shows that F-statistic (chi-square) value is 11.1000 and Q-statistic value is 6. From Table 13, the F-statistic and Q-statistic values are found to be 6 and 12, respectively. Thus, in both of the cases, F-statistic value is greater than its corresponding critical chi-square value (7.82 for case 1 and 3.99 for case 2) and Q-statistic value is also

![Figure 6. Best cost obtained by Quasi Oppositional Symbiotic Organisms Search (QOSOS) and other well-known optimization techniques for Test systems I, II, III, and IV.](image_url)

![Table 12. Ranks achieved by Friedman and Quade tests in Test systems I, II, III, and IV. The statistic computed and related p-values are also shown.](table_url)
Table 13. Ranks achieved by Friedman and Quade tests in Test systems II, III, and IV. The statistic computed and related p-values are also shown.

| Test systems | Friedman test | Quade test |
|--------------|---------------|------------|
|              | QOSOS | SOS | ORCCRO | Test systems | QOSOS | SOS | ORCCRO |
| Test system II | 1     | 2   | 3      | Test system II | -3    | 0   | 3      |
| Test system III | 1    | 2   | 3      | Test system III | -2    | 0   | 2      |
| Test system IV  | 1     | 2   | 3      | Test system IV  | -1    | 0   | 1      |
| Statistic         | 6     |      |         | Statistic         | 12    |      |         |
| p-value            | 0.0498 |      |         | p-value            | 0.0304 |      |         |

Table 14. Average errors obtained in test case in Test system I, Test system II, Test system III, and Test system IV.

| Test systems | QOSOS | SOS | DE/BBO | BBO |
|--------------|-------|-----|--------|-----|
| Test system I | 0.1062 | 0.1068 | 4.3512 | 9.593084 |
| Test system II | 0   | 11.907111 | 514.4098 | 664.226102 |
| Test system III | 0   | 0.2476 | 386.9164 | 473.7064 |
| Test system IV  | 0.0287 | 0.5066 | 42.7857 | 44.3857 |

Table 15. Average errors obtained in test case in Test systems II, III, and IV.

| Test systems | QOSOS | SOS | ORCCRO |
|--------------|-------|-----|--------|
| Test system II | 0   | 11.907111 | 402.832858 |
| Test system III | 0   | 0.2476 | 76.5764 |
| Test system IV  | 0.0287 | 0.5066 | 39.4357 |

greater than its critical value (3.86 for case 1 and 10.92 for case 2). It is also found that p-values obtained by Friedman test and Quade test are less than that at a 5% significance level. This proves that the null hypothesis can be rejected, which signifies a considerable difference in performance between the algorithms. The average errors of various techniques are shown in Tables 14 and 15. The average errors have been calculated as follows:

1. The minimum value among all algorithms for each test system has been chosen;
2. The minimum value has been subtracted from the mean value obtained by each algorithm;
3. All algorithms (rank-wise) have been arranged based on the value of average error.

Thus, based on the average errors evaluated for different cases, the algorithms are ranked, and the results are presented in Tables 12 and 13. From these tables, it is found that the rank acquired by QOSOS algorithm is the lowest, which indicates better performance of QOSOS. Therefore, it may be concluded that, in terms of quality solution, the QOSOS algorithm gives better results than other recently developed optimization techniques.

6. Conclusion

In this paper, Quasi Oppositional Symbiotic Organisms Search (QOSOS) was applied to find the solution to various complex economic dispatch problems. To investigate the computational efficiency, feasibility, and consistency of this method, four different tests were used here. The simulation results of QOSOS method were compared with results of other optimization techniques such as Symbiotic Organisms Search (SOS), Biogeography-Based Optimization (BBO), DE/BBO, Oppositional Real-Coded Chemical Reaction Optimization (ORCCRO), Teaching-Learning Based Optimization (TLBO), Exchange Market Algorithm (EMA), and so on. From these analyses, it was found that the consistency, convergence rate, and solution quality of QOSOS algorithm were better. Therefore, it may be used to find a solution to various complex optimization problems.

Acknowledgement

The authors would like to acknowledge the Department of Electrical Engineering, NIT Agartala for providing laboratory facilities.

References

1. Fanshel, S. and Lynes, E.S. “Economic power generation using linear programming”, IEEE Trans. Power Appar. Syst., 83(4), pp. 347-356 (1964), DOI: 10.1109/TPAS.1964.476611
2. Bellman, R., The Theory of Dynamic Programming, Rand Corp Santa Monica CA (1954).
3. Wood, J. and Wollenberg, B.F., *Power Generation, Operation, and Control*, John Wiley and Sons, 2nd Edn., Wiley New York (1984).

4. Ingber, L. “Simulated annealing : Practice versus Theory”, *Math. Comput. Modeling*, 18(11), pp. 29–57 (1993). DOI: 10.1016/0895-7177(93)90204-C

5. Panigrahi, C.K., Chattopadhyay, P.K., and Chakrabarti, R.N., et al. “Simulated annealing technique for dynamic economic dispatch”, *Electr. Power Compon. Syst.*, 34(5), pp. 575–586 (2006). DOI: 10.1080/1532500060083843

6. Walters, D.C. and Shleble, G.B. “Genetic algorithm solution of economic dispatch with valve point loadings”, *IEEE Trans. Power Syst.*, 8(3), pp. 1325-1331 (1993). DOI: 10.1109/59.260861

7. Chiang, C.L. “Improved genetic algorithm for power economic dispatch of units with valve-point effects and multiple fuels”, *IEEE Trans. Power Syst.*, 20(4), pp. 1690–1699 (2005). DOI: 10.1109/TPWRS.2005.857924

8. Kennedy, J. and Eberhart, R. “Particle swarm optimization”, In *Proc. IEEE Int. Conf. Neural Networks*, IV, pp. 1942–1948 (1995). DOI: 10.1109/ICNN.1995.488068

9. Gaiw, Z.-L. “Particle swarm optimization to solving the economic dispatch considering the generator constraints”, *IEEE Trans. Power Syst.*, 18(3), pp. 1187–1195 (2003). DOI: 10.1109/TPWRS.2003.814889

10. Selvakumar, I. and Thanushkodi, K. “A new particle swarm optimization algorithm for nonconvex economic dispatch problems”, *IEEE Trans. Power Syst.*, 22(1), pp. 42-51 (2007). DOI: 10.1109/TPWRS.2006.889132

11. Panigrahi, B.K., Pandi, V.R., and Das, S. “Adaptive particle swarm optimization approach for static and dynamic economic load dispatch”, *Energy Convers. Manage.*, 49(6), pp. 1407–1415 (2008). DOI: 10.1016/j.enconman.2007.12.023

12. Vlachogiannis, I.K. and Lee, K.Y. “Economic load dispatch - a comparative study on heuristic optimization techniques with an improved coordinated aggregation-based PSO”, *IEEE Trans. Power Syst.*, 24(2), pp. 991–1001 (2009). DOI: 10.1109/TPWRS.2009.2016524

13. Park, J.B., Jeong, Y.W., Shin, J.R., et al. “An improved particle swarm optimization for non-convex economic dispatch problems”, *IEEE Trans. Power Syst.*, 25 (1), pp. 156–166 (2010). DOI: 10.1109/TPWRS.2009.2030293

14. Hosseini ted, V., Rafiee, M., Almahdian, M., et al. “Species-based quantum particle swarm optimization for economic load dispatch”, *Int. J. Electr. Power & Energy Syst.*, 63, pp. 311–322 (2014). DOI: 10.1016/j.ijepes.2014.05.066

15. Storn, R. and Price, K.V. “Differential evolution a simple and efficient heuristic for global optimization over continuous spaces”, *J. Global Optim.*, 11(4), pp. 341–359 (1997). DOI: 10.1023/A:100820282

16. Noman, N. and Iba, H. “Differential evolution for economic load dispatch problems”, *Electr. Power Syst. Res.*, 78(3), pp. 1322–1331 (2008). DOI: 10.1016/j.epr.2007.11.007

17. Coelho, L.D.S. and Mariani, V.C. “Combining of chaotic differential evolution and quadratic programming for economic dispatch optimization with valve-point effect”, *IEEE Trans. Power Syst.*, 21 (2), pp. 989–996 (2006). DOI: 10.1109/TPWRS.2006.873410

18. Paroush, R.P. and Das, K.N. “A novel hybrid optimizer for solving economic load dispatch problem”, *Int. J. Electr. Power & Energy Syst.*, 78, pp. 108–126 (2016). DOI: 10.1016/j.ijepes.2015.11.058

19. Zhou, D., Li, S., Wang, G-G., et al. “An improved differential evolution algorithm for the economic load dispatch problems with/without valve-point effects”, *Appl. Energy*, 181, pp. 375–390 (2016). DOI: 10.1016/j.apenergy.2016.08.067

20. Jayabharathi, T., Jayapralash, K., Jayakumar, N., et al. “Evolutionary programming techniques for different kinds of economic dispatch problems”, *Electr. Power Syst. Res.*, 75(2), pp. 169–176 (2005). DOI: 10.1016/j.epsr.2004.08.001

21. Sinha, N., Chakrabarti, R., and Chattopadhyay, P.K. “Evolutionary programming techniques for economic load dispatch”, *IEEE Trans. Evol. Comput.*, 7(1), pp. 83–94 (2003). DOI: 10.1109/TEVC.2002.806888

22. Panigrahi, B.K. and Pandi, V.R. “Bacterial foraging optimization Nelder-Mead hybrid algorithm for economic load dispatch”, *IET Generation, Transm. Distrib.*, 2(4), pp. 556–65 (2008). DOI: 10.1049/iet-gtd:20070122

23. Simon, D. “Biogeography-based optimization”, *IEEE Trans. Evol. Comput.*, 12(6), pp. 702-713 (2008). DOI: 10.1109/TEVC.2008.919004

24. Bhattacharya, A. and Chattopadhyay, P.K. “Biogeography-based optimization for different economic load dispatch problems”, *IEEE Trans. Power Syst.*, 25(2), pp. 1064–1077 (2010). DOI: 10.1109/TPWRS.2009.2034525

25. Bhattacharya, A. and Chattopadhyay, P.K. “Hybrid differential evolution with biogeography-based optimization for solution of economic load dispatch”, *IEEE Trans. Power Syst.*, 25(4), pp. 1955–1964 (2010). DOI: 10.1109/TPWRS.2010.2043270

26. Lam, A.Y.S. and Li, V.O.K. “Chemical-reaction-inspired metaheuristic for optimization”, *IEEE Trans. Evol. Comput.*, 14(8) pp. 381–399 (2010). DOI: 10.1109/TEVC.2009.2033580

27. Bhattacharjee, K., Bhattacharya, A., and Dey, S.H.N. “Chemical reaction optimisation for different economic dispatch problems”, *IET Gener. Transm. Distrib.*, 8(3), pp. 530–541 (2014). DOI: 10.1049/iet-gtd:2013.0122

28. Bhattacharjee, K., Bhattacharya, A., and Dey, S.H. “Oppositional real coded chemical reaction optimization for different economic dispatch problems”, *Int. J.*
29. Rao, R.V., Savsani, V.J., and Vakharia, D.P., “Teaching-learning based optimization: a novel method for constrained mechanical design optimization problems”, Comp. Aided Design, 43(3), pp. 303–315 (2011). DOI: 10.1016/j.cad.2010.12.015

30. Bhattacharjee, K., Bhattacharya, A., and Dey, S.H.N., “Teaching learning based optimization for different economic dispatch problems”, Scientia Iranica, 21(3), pp. 870–884 (2013).

31. Banerjee, S., Maity, D., and Chanda, C.K., “Teaching learning based optimized valve point loading considering valve point loading effect”, Int. J. Electr. Power & Energy Syst., 73, pp. 456–464 (2015). DOI: 10.1016/j.ijepes.2015.05.036

32. He, X., Rao, Y., and Huang, J., “A novel algorithm for economic dispatch of power systems”, Neurocomputing, 171, pp. 1454–1461 (2016). DOI: 10.1016/j.neucom.2015.07.107

33. Mirjalili, S., Mirjalili, S.M., and Lewis, A., “Grey wolf optimizer”, Adv. Eng. Softw., 69, pp. 46–61 (2014). DOI: 10.1016/j.advengsoft.2013.12.007

34. Kamboj, V.K., Bath, S.K., and Dhillon, J.S., “Solution of non-convex economic load dispatch problem using grey wolf optimizer”, Neural Comput & Applic., 27(5), pp. 1301–1316 (2006). DOI: 10.1007/s00521-015-1934-8

35. Rajagopalan, A., Songod, V., and Govindasamy, R., “Solving economic load dispatch problems using chaotic self-adaptive differential harmony search algorithm”, Int. Trans. Electr. Energy Syst., 25(5), pp. 845–856 (2014). DOI: 10.1002/etep.1877

36. Mandal, B., Roy, P.K., and Mandal, S., “Economic load dispatch using krill herd algorithm”, Int. J. Electr. Power Energy Syst., 57, pp. 1–10 (2014). DOI: 10.1016/j.ijepes.2013.11.016

37. Barisal, A.K. and Prusty, R.C., “Large scale economic dispatch of power systems using oppositional invasive weed optimization”, Appl. Soft Comput., 29, pp. 122–137 (2015). DOI: 10.1016/j.asoc.2014.12.014

38. Mirjalili, S., “The ant lion optimizer”, Adv Eng Softw., 83, pp. 80–98 (2015). DOI: 10.1016/j.advengsoft.2015.01.010

39. Kamboj, V.K., Bladoria, A., and Bath, S.K., “Solution of non-convex economic load dispatch problem for small-scale power systems using ant lion optimizer”, Neural Comput & Applic., 28, pp. 2181–2192 (2016). DOI: 10.1007/s00521-015-2148-9

40. Sulvatra, M.S.P., Easter, S.E., Victoire, T.A., et al., “A hybrid with cross-entropy method and sequential quadratic programming to solve economic load dispatch problem”, IEEE Sys. Journal, 9(3), pp. 1031–1044 (2015). DOI: 10.1109/JSYST.2013.2297471

41. Al-Betar, M.A., Awadallah, M.A., Khader, A.T., et al., “Tournament based harmony search algorithm for non-convex economic load dispatch problem”, Appl. Soft comput., 47, pp. 449–459 (2016). DOI: 10.1016/j.asoc.2016.05.034

42. Ghobani, N. and Balaei, E., “Exchange market algorithm for economic load dispatch”, Int. J. Electr. Power Energy Syst., 75, pp. 19–27 (2016). DOI: 10.1016/j.ijepes.2015.08.013

43. Mohammadi, F. and Abd, H., “A modified crow search algorithm (MCsA) for solving economic load dispatch problem”, Appl. Soft Comput., 71, pp. 51–65 (2018). DOI: 10.1016/j.asoc.2018.06.040

44. Cheng, M.Y. and Prayogo, D., “Symbiotic organisms search: A new metaheuristic optimization algorithm”, Computers & Structures, 139, pp. 98–112 (2014). DOI: 10.1016/j.compstruc.2014.03.007

45. Duman, S., “Symbiotic organisms search algorithm for optimal power flow problem based on valve-point effect and prohibited zones”, Neural Comput & Applic., 28, pp. 3571–3585 (2016). DOI: 10.1007/s00521-016-2355-0

46. Guvenç, U., Duman, S., Sonmez, Y., et al., “Symbiotic organisms search algorithm for economic load dispatch problem with valve point effect”, Scientia Iranica, 25(6), pp. 3490–3506 (2017).

47. Tizhoosh, H., “Opposition-based learning: A new scheme for machine intelligence”, In Proceedings of the International Conference on Computational Intelligence for Modelling Control and Automation, Austria, pp. 695–701 (2005). DOI: 10.1109/CIMCA.2005.1631345

48. Egezer, M., Simon, D., and Du, D., “Optimization”, In Proceedings of the IEEE International Conference on Oppositional Biogeography-Based Systems, Man and Cybernetics, San Antonio, TX, USA, pp. 1009–1014 (2009). DOI: 10.1109/CEC.2011.5949792

49. Aragon, V.S., Esquivel, S.C., and Coello, C.A.C., “An immune algorithm with power redistribution for solving economic load dispatch problems”, Inf. Sciences, 295, pp. 609–632 (2014). DOI: 10.1016/j.ins.2014.10.026

50. Ciorenţii, I. and Kyriakides, E., “Efficient hybrid optimization solution for the economic dispatch with nonsmooth cost function”, In Proc. IEEE Power Tech, Bucharest, Romania, pp. 1–7 (2009). DOI: 10.1109/PHTC.2009.5282062

51. Reddy A.S. and Vaiashka, K., “Shuffled differential evolution for large scale economic dispatch”, Electr. Power Syst. Res., 96, pp. 237–245 (2013). DOI: 10.1016/j.ener.2012.11.010

52. Ciorenţii, I. and Kyriakides, E., “A GA-API solution for the economic dispatch of generation in power system operation”, IEEE Trans. Power Syst., 27(1), pp. 233–242 (2011). DOI: 10.1109/TPWRS.2011.2168883

53. Derac, J., Garcia, S., Molina, D., et al., “A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary
and swarm intelligence algorithms", *Swarm and Evolutionary Computation*, 1, pp. 3–18 (2011). DOI: 10.1016/j.swevo.2011.02.002

54. Shenkin, D.J., *Hand Book of Parametric and Non Parametric Statistical Procedures*, 4th Ed., Chapman & Hall/CRC (2006).

**Biographies**

**Diptanu Das** received his BE in Electrical Engg from NIT Agartala, India (formerly Tripura Engineering College) in 2009, MTech in Electrical Engg from NIT Agartala, India in 2011. He is currently pursuing the PhD degree at the Department of Electrical Engineering at NIT Agartala, India. He is also working as an Assistant Professor at the Department of Electrical Engineering of NIT Agartala, India. His areas of interest include power system optimization, power electronics, power, and energy system.

**Aniruddha Bhattacharya** received his BSc Engg in Electrical Engg from Regional Institute of Technology, Jamshedpur, India, 2000, M.E. E. and PhD in Electrical Power System from Jadavpur University, Kolkata, India, 2008 and 2011, respectively. His employment experience includes Siemens Metering Limited, India; Jindal Steel & Power Limited, Raigarh, India; Bankura Unnyani Institute of Engineering, Bankura, India; Dr. B. C. Roy Engineering College, Durgapur, India. He is currently an Assistant Professor at the Electrical Engineering Department, NIT Agartala, India. His areas of interest include power system load flow, optimal power flow, economic load dispatch, and soft computing applications to power system problems.

**Rup Narayan Ray** received his BE in Electrical Engg from University of Calcutta, India in 1985, his ME from Bengal Engineering College, Shibpur, India with specialization in Electrical Machines in 1995, and his PhD in Electrical Engineering from Jadavpur University, Kolkata, India in 2010. He joined teaching as a Lecturer at NIT Agartala (formerly Tripura Engineering College) in 1987. He is currently an Associate Professor at the Electrical Engineering Department, NIT Agartala, India. His areas of interest include electrical machines and drives, power qualities, and distributed generations.