mathematics

MDPI

Article

Oppositional Pigeon-Inspired Optimizer for Solving the Non-Convex Economic Load Dispatch Problem in Power Systems

Rajakumar Ramalingam [1], Dinesh Karunanidy [1], Sultan S. Alshamrani [2], Mamoon Rashid [3,*], Swamidoss Mathumohan [4] and Ankur Dumka [5,6]

[1] Department of Computer Science and Technology, Madanapalle Institute of Technology & Science, Madanapalle 517325, Andhra Pradesh, India
[2] Department of Information Technology, College of Computers and Information Technology, Taif University, P.O. Box 11099, Taif 21944, Saudi Arabia
[3] Department of Computer Engineering, Faculty of Science and Technology, Vishwakarma University, Pune 411048, Maharashtra, India
[4] Department of CSE, Unnamalai Institute of Technology, Kovilpatti 628502, Tamil Nadu, India
[5] Department of Computer Science and Engineering, Women Institute of Technology, Dehradun 248007, Uttarakhand, India
[6] Department of Computer Science and Engineering, Graphic Era Deemed to be University, Dehradun 248007, Uttarakhand, India
* Correspondence: mamoon.rashid@vupune.ac.in; Tel.: +91-7814346505

Abstract: Economic Load Dispatch (ELD) belongs to a non-convex optimization problem that aims to reduce total power generation cost by satisfying demand constraints. However, solving the ELD problem is a challenging task, because of its parity and disparity constraints. The Pigeon-Inspired Optimizer (PIO) is a recently proposed optimization algorithm, which belongs to the family of swarm intelligence algorithms. The PIO algorithm has the benefit of conceptual simplicity, and provides better outcomes for various real-world problems. However, this algorithm has the drawback of premature convergence and local stagnation. Therefore, we propose an Oppositional Pigeon-Inspired Optimizer (OPIO) algorithm—to overcome these deficiencies. The proposed algorithm employs Oppositional-Based Learning (OBL) to enhance the quality of the individual, by exploring the global search space. The proposed algorithm would be used to determine the load demand of a power system, by sustaining the various equality and inequality constraints, to diminish the overall generation cost. In this work, the OPIO algorithm was applied to solve the ELD problem of small- (13-unit, 40-unit), medium- (140-unit, 160-unit) and large-scale (320-unit, 640-unit) test systems. The experimental results of the proposed OPIO algorithm demonstrate its efficiency over the conventional PIO algorithm, and other state-of-the-art approaches in the literature. The comparative results demonstrate that the proposed algorithm provides better results—in terms of improved accuracy, higher convergence rate, less computation time, and reduced fuel cost—than the other approaches.

Keywords: economic load dispatch; pigeon-inspired optimizer; oppositional-based learning; swarm intelligence algorithm; oppositional-based pigeon-inspired optimizer

MSC: 68W50; 60G05; 60G51; 90C27

Citation: Ramalingam, R.; Karunanidy, D.; Alshamrani, S.S.; Rashid, M.; Mathumohan, S.; Dumka, A. Oppositional Pigeon-Inspired Optimizer for Solving the Non-Convex Economic Load Dispatch Problem in Power Systems. *Mathematics* **2022**, *10*, 3315. https://doi.org/10.3390/math10183315

Academic Editor: Jian Dong

Received: 29 July 2022
Accepted: 9 September 2022
Published: 13 September 2022

1. Introduction

With the rapid growth in technologies, ELD is considered one of the foremost challenging optimization problems in power systems. The main motive for addressing the ELD problem is to reduce the cost of power generation, by sustaining the different constraints involved in the generation units [1]. Several researchers have applied mathematical models, knowledge discovery and optimization techniques to resolve the ELD problem. The standard techniques, like lambda-generation techniques, and base-point techniques

from [2], provide optimal solutions, by incorporating the incremental cost curves of linear functions. However, these methods have failed to solve highly non-linear functions, and provide unsatisfactory solutions which result in huge losses in power generation costs. The non-smooth functionalities of generating units contain various features, like prohibited zones, different fuel options, value-point effects, ramp-rate limits and a start-up cost function which converts linear into non-linear characteristics [3]. Owing to the large-scale generating units, conventional methods have provided unreliable solutions, and have taken a lot of computational time to solve ELD problems. In later studies, dynamic programming techniques [4] have been used for ELD problems, but these have required high computational efforts to solve large-scale generating units.

In recent studies, many researchers have utilized various optimization algorithms to solve non-convex ELD problems with only value-point effects, viz., Particle Swarm Optimization with Sequential Quadratic Programming (PSO-SQP) [5], Genetic Algorithm (GA) [6], Evolutionary Programming (EP) [7], Improved Group Search Optimization (IGSO) [8], Incremental Artificial Bee Colony with Local Search (IABC-LS) [9], Hybrid Grey Wolf Optimizer (HGWO) [10], Self-Organizing Hierarchical Particle Swarm Optimization (SOH-PSO) [11], Genetic Algorithm with Pattern Search and SQP (GA-PS-SQP) [12], Modified Shuffled Frog-Leaping Algorithm (MSFLA) [13], Firefly Optimization (FA) [14], Chaotic Self-Adaptive Particle Swarm Optimization Algorithm (CSAPSO) [15], Combined Social Engineering Particle Swarm Optimization (SEPSO) [16], Starling Murmuration Optimizer (SMO) [17], Improved Moth-Flame optimization (IMFO) [18] and Diversity-Maintained Differential Evolution (DMDE) [19]. Among these search techniques, GA is considered to be the least efficient technique, because its optimal individuals are generally trapped in intensification rather than diversification, and it also suffers from the determination of control parameters, which results in excessive simulation time. Several new techniques, like IGSO, MSFLA, FA, HGWO, SOH-PSO, GA-PS-SQP and CSAPSO, have virtuoso competence in finding optimal solutions for non-convex generating units; however, the simulation time of the system is quite long; specifically, for CSAPSO, several iterations are carried out to specify the control parameter values; this limitation results in the technique having excessive execution time, and a large number of runs.

In addition, some sets of optimization algorithms are considered to solve non-convex ELD problems with only multi-fuel possibilities. These algorithms include Integer Coded Differential Evolution-Dynamic Programming (ICDEDP) [20], Chaotic Ant Swarm Optimization (CASO) [21], Bacteria Foraging Optimization (BFO) [22], Ant Colony Optimization (ACO) [23], Biogeography-Based Optimization (BBO) [24] and Krill Herd (KH) [25]. Among these techniques, ACO is the technique initially utilized for solving optimization problems in the engineering domain, specifically in path-identifying and parameter-tuning in electrical engineering. Although ACO and CASO have the cap potential of leading complicated constraints and non-convex goal features, in addition to their simplicity of simulation for optimization problems, they nevertheless suffer from numerous negative aspects, together with low-quality optimization individual and lengthy simulation time. The modified DE method, namely the ICDEDP technique, can be considered a more efficient technique than the other techniques, because it can obtain a good-quality solution within a short span of simulation; this DE technique has been globally utilized in power system optimization problems. In addition, other techniques—such as BBO, KH and BFO—have good capability in determining the optimal solutions for non-convex problems; however, the simulation times of these techniques are longer, due to the vast number of control parameters.

In contrast to the aforementioned sets, the techniques in the set of neural networks including the Adaptive Hopfield Neural Network (AHNN) [26], the Enhanced Augmented Lagrange Hopfield Network (EALHN) [27] and the Augmented Lagrange Hopfield Network (ALHN) [28] can impact on large-scale problems, but fail to deal with the ELD problem with a non-convex objective function. In EALHN and ALHN, the Lagrange function is merged with the Hopfield network to enhance efficacy. This process will help the techniques to converge towards the optimal more smoothly, and to obtain a good-quality

solution. However, in real-time power systems, both value points and fuel points need to be considered, for accurate and practical ELD solutions.

In some studies, both the constraints of value points and different fuel possibilities are considered for realistic ELD solutions comprising the Improved Particle Swarm Optimization (IPSO) [29], the Crisscross Optimization Algorithm (COA) [30], Differential Evolution and Particle Swarm Optimization (DEPSO) [31], the Oppositional Grey Wolf Optimization algorithm (OGWO) [32], Estimation of Distribution and Differential Evolution Cooperation (ED-DE) [33], the Real-Coded Chemical Reaction Algorithm (RCCRO) [34], Synergic Predator–Prey Optimization (SPPO) [35], the One Rank Cuckoo Search Algorithm (ORCSA) [36], the Real-Coded Genetic Algorithm (RCGA) [37] and the Improved Genetic Algorithm [38]. By utilizing the pros of each search technique, these improved novel techniques have adequate capability in finding good-quality solutions with better simulation time. However, the improved technique can lead to more complications with vast control parameters, and it can suffer from inappropriate selection of these parameters; in addition, its performance is degraded when applied to large-scale power systems entailing n number of generating units with various fuel possibilities and value-point effects.

A large portion of the above studies have focused on the adjustments of stochastic search techniques. Nonetheless, they have, once in a while, given consideration to the method of handling constraints. In reality, dealing with the constraints of ELD problems is significant when working with stochastic search techniques, for enhancing the optimization results. Our study aimed to fill the research gap, by contributing more towards addressing the constraints of ELD problems. Our contributions were twofold: initially, an enhanced PIO algorithm was introduced, to enrich the performance of the standard PIO algorithm; subsequently, a constraint-handling technique was utilized, to appropriately handle the equality constraints.

The Pigeon-Inspired Optimizer algorithm was inspired by the homing bias of pigeons, and was proposed by Duan and Qiao in 2014. This optimization algorithm was used because of its optimum performance at high merging speeds [39]. However, the PIO algorithm suffers in regard to global exploration and premature convergence. In addition, its performance is degraded when applied to high-dimensional problems. This problem can be overcome by using the Opposition-Based Learning technique. The OBL technique is widely used by researchers to boost convergence speed, by exploring the search space. In this work, a new metaheuristic algorithm—namely, the Oppositional Pigeon-Inspired Optimizer technique (OPIO)—was utilized, to solve non-convex ELD problems with various fuel possibilities and value-point effects.

The major contribution of this work is illustrated as follows:

(1) The proposed OPIO algorithm solves the non-convex ELD problem with multi-fuel possibilities and value-point effects, through two operators: namely, map and compass operator, and landmark operator. These operators enhance the local search ability by adopting the search boundary limits. Later, the Opposition-Based Learning strategy helps to explore the search space, as well as to enhance the exploration ability for target search agents. This process improves the search capability, and eradicates premature convergence, though the large-scale test system holds both multiple fuel possibilities and value-point effects.

(2) The proposed OPIO algorithm has a unique adjustable parameter: jump rate J_r. Parameter J_r helps to determine the global optimal solution, by influencing the adjustable value, within the range of 0 to 0.4. This parameter promotes the OPIO algorithm, to be robust and adaptable in solving ELD problems with different constraints.

(3) To validate the efficiency of the proposed OPIO algorithm, we used several test cases, which varied according to three scales: small-scale (i.e., 13, 40); medium-scale (i.e., 140, 160); and large-scale (i.e., 320, 640) generation units. The results of the various test cases confirmed that the proposed technique is a better potential solution than the state-of-the-art metaheuristic algorithms in the literature. The OPIO algorithm provided better performance in the 320- and 640-unit generation systems. This shows that the

formulated technique is a superior and reliable solution for large-scale ELD problems over multiple trials.

The rest of this work is categorized as follows: Section 2 delivers the mathematical formulation of the ELD problem, with objective functions and multiple constraints. The proposed Oppositional Pigeon-Inspired Optimizer algorithm is presented in detail in Section 3. In Section 4, the implementation of the OPIO algorithm, in solving the ELD problem, is presented. Section 5 provides proposed OPIO algorithm experimentation details, from six different test cases that varied from small-scale to large-scale systems, and the outcomes are compared with state-of-the-art metaheuristics algorithms. The conclusion of this work is presented in Section 6.

2. ELD Problem Formulation

The main motive of ELD is to reduce the overall power generation cost, by solving different disparity and parity constraints, to provide optimal generation among power producing units [32]. The objective function and the different constraints of the ELD problem are presented in this section.

2.1. Fitness Function

The fitness function of the ELD problem is to reduce the total power production cost by solving various constraints, and to gratify the load demand over some reasonable stage. A quadratic function is formulated, to approximate the fuel cost of the power-producing unit. The mathematical formulation of the power-generating unit is formulated as below:

$$\min \sum_{j=1}^{n} F_c(\Psi_j) \tag{1}$$

Here F_c denotes the fuel cost of the generator (in \$/h); Ψ_j denotes the output power of generator j (in MW); n stands for the overall power-generating unit in the power system.

In view of the value-point effects, ELD cost functions will have non-smooth points which provide inefficient results in practical generators. To process the practical generators, sinusoidal functions are included in the quadratic functions. The cost function, with value points of unit j, is represented as follows:

$$F_c = k_j\Psi_j^2 + l_j\Psi_j + m_j + \left|a_j \times \sin\left(b_j \times \left(\Psi_j^{low} - \Psi_j\right)\right)\right| \tag{2}$$

Here, k_j, l_j and m_j stand for the fuel cost coefficients of generator j; a_j and b_j stand for the value-point loading coefficients of generator j; Ψ_j^{low} is the low-level range power production of generator j.

The overall fuel cost function of n generator in real-time ELD is mathematically formulated as follows:

$$\min \sum_{j=1}^{n} \hat{F}_c(\Psi_j) = \sum_{j=1}^{n} \left[k_j\Psi_j^2 + l_j\Psi_j + m_j + \left|a_j \times \sin\left(b_j \times \left(\Psi_j^{low} - \Psi_j\right)\right)\right|\right] \tag{3}$$

where $\hat{F}_c$ stands for the real-time fuel cost of the generator.

To attain an accurate and more appropriate solution for the ELD problem, both various fuel possibilities and value-point effects are added with the cost functions. Most thermal generating units utilize multiple fuel possibilities, using the load and suitability of the power generation units. The cost function of generating unit j, with various fuel possibilities (q) and value-point effects, is mathematically formulated and presented as follows:

$$F_c(\Psi_j) = \begin{cases} k_{j1}(\Psi_j)^2 + l_{j1}(\Psi_j) + m_{j1} + \left|a_{j1} \times \sin\left(b_{j1} \times \left(\Psi_j^{low} - \Psi_j\right)\right)\right| if \Psi_j^{low} \le \Psi_j \le \Psi_{j1} \\ k_{j2}(\Psi_j)^2 + l_{j2}(\Psi_j) + m_{j2} + \left|a_{j2} \times \sin\left(b_{j2} \times (\Psi_{j2} - \Psi_j)\right)\right| if \Psi_{j1} \le \Psi_j \le \Psi_{j2} \\ k_{jq}(\Psi_j)^2 + l_{jq}(\Psi_j) + m_{jq} + \left|a_{jq} \times \sin\left(b_{jq} \times \left(\Psi_{jq} - \Psi_j^{low}\right)\right)\right| if \Psi_{jq} \le \Psi_j \le \Psi_j^{upper} \end{cases} \tag{4}$$

2.2. Constraints of the ELD Problem

The fitness function in Section 2.1 is formulated with a set of constraints, which are given below.

2.2.1. Operating Unit Limit

The power-generating unit must relay within the lower and upper boundary limits:

$$\Psi_j^{low} \leq \Psi_j \leq \Psi_j^{upper} j = 1, 2, \ldots, n \tag{5}$$

where Ψ_j^{upper} and Ψ_j^{low} denote the upper and lower boundary, respectively, of the output power of the generator j.

2.2.2. Power-Stabilizing Constraints

The overall generated power should be the same as the overall losses and overall load request of the units. This constraint is mathematically formulated as follows:

$$\sum_{j=1}^{n} \Psi_j - \Psi_{Demand} - \Psi_{Loss} = 0 \tag{6}$$

where Ψ_{Loss} and Ψ_{Demand} represent the overall power loss and power demand of the units. Based on Kron's loss technique, the transmission loss is given as follows:

$$\Psi_{Loss} = \sum_{j=1}^{n} \sum_{i=1}^{n} \Psi_j \beta_{ji} \Psi_i + \sum_{j=1}^{n} \beta_{0j} \Psi_j + \beta_{00} \tag{7}$$

where β_{ji} represents the loss coefficient element j and i of the symmetric matrix β; β_{0j} denotes the loss coefficient vector of j symmetric matrix β; and β_{00} represents a fixed loss coefficient concerning standard operating situations.

2.2.3. Restricted Operating Regions (RORs)

Due to oscillation or steam value process in the shaft bearing, the restricted operating region is considered. To avoid these issues, choosing the best operating region will drastically increase the optimum economy of the generating units. The boundary constraints of the standard operating section of generator j are formulated as follows:

$$\Psi_j \in \begin{cases} \Psi_j^{low} \leq \Psi_j \leq \Psi_{j,1}^{l} \\ \Psi_{j,i-1}^{l} \leq \Psi_j \leq \Psi_{j,i}^{l} \\ \Psi_{j,n_i}^{l} \leq \Psi_j \leq \Psi_j^{upper} \end{cases} \quad i = 2, 3, \ldots, n_j, \text{ j} = 1, 2, \ldots, \text{n} \tag{8}$$

where l, u denotes the lower and upper limits of specific power generating units, and n_j determines the number of restricted regions of generating unit j.

2.2.4. Ramp-Rate (RR) Constraint

In view of the lower and upper power production of the generator, the ramp-rate limit is considered. Each generating unit is controlled by the ramp-rate limit, which instructs the generator to function continually for the two nearest operating regions. This ramp-rate constraint is represented as follows:

$$max\left(\Psi_j^{low}, \Psi_j^0 - LSL_j\right) \leq \Psi_j \leq min\left(\Psi_j^{upper}, \Psi_j^0 + USL_j\right) \tag{9}$$

where LSL_j and USL_j represent the lower and upper slope (or ramp) limit of the generating unit j, and Ψ_j^0 denotes the current power generating unit j.

3. Preliminaries

In this section, we present three major mechanisms; firstly, the generic working process of the Pigeon-Inspired Optimizer is presented, secondly, the core concept of the Opposition-Based Learning technique is discussed; and, finally, the proposed methodology, with its working process, is presented.

3.1. Overview of Pigeon-Inspired Optimizer

The Pigeon-Inspired Optimizer (PIO) belongs to the family of swarm intelligence algorithms that were proposed by Haibin Duan and Peixin Qiao (2014) [39]. The PIO algorithm mimics the homing behaviors of pigeons. Most researchers apply SI algorithms to solve their domain-related NP-hard problems, in which search space is vast. SI algorithms are inspired by the social behavior of the swarm, with intellectual learning to determine high-quality solutions using mathematical formulations. The mathematical formulation of the swarm includes the position and velocity of the swarm iteration by iterations.

Pigeons have the ability to explore for food over the course of long intervals. In addition, pigeons exhibit intellectual homing behavior: for example, they carried messages during the First and Second World Wars. The PIO algorithm works on the basis of two unique operators, viz., map and landmark operators. This algorithm provides good optimum performance and higher merge speed than the other state-of-the-art metaheuristic algorithms like Ant Colony Optimization, Particle Swarm Optimization, Artificial Bee Colony Optimization and Differential Evolution algorithms.

3.1.1. Map and Compass Operator

Pigeons have a natural ability to perceive the orbital meadow, with the aid of a magnetic function that enables them to map. They utilize the altitude of the sun as a compass to fine-tune their current directions. Generally, pigeons depend less on the sun and on magnetic particles as they near their destinations. The map and compass operator can be mathematically formulated as follows:

$$V_j^{t+1} = V_j^t \times e^{-\rho t} + rand \times \left(X_g - X_j^t\right) \tag{10}$$

$$X_j^{t+1} = X_j^t + V_j^{t+1} \tag{11}$$

where V_j^t and X_j^t represent the velocity and position of the j individuals in the t iterations; ρ denotes the map and compass factor; *rand* determines the uniform random variable within [0, 1]; X_g denotes the global best individual; and X_j^{t+1} and V_j^{t+1} represent the new position and velocity of the j individual in the next t iteration.

3.1.2. Landmark Operator

A pigeon relies on natural landmarks once it has reached its destination. However, if the pigeon is far away from its destination, then it relies on the adjacent pigeons to adjust its position. In this algorithm, half of the pigeon population is allowed to adjust position, with the aid of the centered pigeons, while the pigeons comprising the other half of the population adjust their position in accordance with the desirable destination position. Most pigeons will not be familiar with their landmark in this view, so they will follow the top-ranked pigeons to determine their desired destination. The half-number of pigeons adjust their position with the following mathematical formulations:

$$N_P^{t+1} = \frac{N_p^t}{2} \tag{12}$$

$$X_c^{t+1} = \frac{\sum X_j^{t+1} \times Fit\left(X_j^{t+1}\right)}{N_p \sum Fit\left(X_j^{t+1}\right)} \tag{13}$$

where N_p^t represents the number of pigeons or population size in the current iteration t; and $Fit\left(X_c^{t+1}\right)$ denotes the fitness of the centered pigeons in the $t+1$ iteration. The new pigeon position is represented as:

$$X_j^{t+1} = X_j^t + rand \times \left(X_c^{t+1} - X_j^t\right) \tag{14}$$

The generic flow of the PIO algorithm is represented in Algorithm 1. In this algorithm, the map and compass operator is given in the initial while loop, and another loop is used to access their route and its correction in position.

Algorithm 1: Standard Pigeon-Inspired Optimizer (PIO)

Input: Number of Population N_p problem space D, Map and compass factor ρ, Number of generations ng_1, ng_2 where $ng_1 > ng_2$.
Output: X_g–Global best solution
1: Randomly generate the solution X_j
2: Compute the fitness of solutions $(X_1, X_2, \ldots, X_{N_p})$
3: Determine the minimal fitness solution as X_g.
4: while $(ng \geq 1)$ do.
5: Determine the velocity and position for each solution by Equations (10) and (11).
6: Compute fitness values of solutions $(X_1, X_2, \ldots, X_{N_p})$
7: Update global best solution X_g.
8: end while
9: while $(N_p \geq 1)$ do
10: Sort solutions by their fitness.
11: $N_p = N_p/2$
12: Compute the desired destination by Equation (13).
13: Update the position of the solution by Equation (14).
14: Update global best solution X_g
15: end while

3.2. Opposition-Based Learning Technique

The Opposition-Based Learning technique (OBL) was introduced by Tizhoosh [40] to enhance the convergence speed of traditional metaheuristic algorithms. This method utilizes the valuation of a current population against its opposite population, to determine the better solution for a specific problem. The OBL method has been utilized in different metaheuristic algorithms, to boost convergence speed [41,42]. The mathematical formulation of the OBL is defined as follows:

Let $\mu(\mu \in [p, q])$ be an actual integer. The contradictory integer μ^0 is formulated as:

$$\mu^0 = p + q - \mu^0 \tag{15}$$

For d–dimensional search space, the contradictory integer μ^0 is defined as:

$$\mu_j^0 = p_j + q_j - \mu_j \tag{16}$$

where $\mu_1, \mu_2, \ldots, \mu_d$ is a point in d-dimensional search space, i.e., $\mu_i \in \left[p_j, q_j\right]$; $j = \{1, 2, 3, \ldots, d\}$, and d represents the number of decision variables.

The Oppositional-Based Learning technique is generally used in two stages: firstly, in the initialization procedure; and secondly, in generating an opposite solution, using the jumping rate J_r. The proposed OBL algorithm is given in Algorithm 2.

Algorithm 2: Oppositional-Based Learning Algorithm

1: Initially the solutions are randomly initialized within the upper and lower boundary regions.
2: Determine the opposite solutions:
    2.1: *for i = 1:N_p*
    2.2: *for j = 1:d*
    2.3: *μ_(i,j)^0 = p_j + q_j − μ_(i,j)*
    2.4: *end for*
    2.5*: end for*
3: Sort the current solutions and opposite solutions in ascending order.
4: Choose the N_p the number of best candidate solutions.
5: Update the control parameters.
6: Generate the opposite solutions from current solutions using jumping rate J_r:
    6.1: *for j = 1:N_p*
    6.2: *for I = 1:d*
    6.3: *if J_r > rand*
    6.4: *opp(j,i) = min(i) + max(i) − P(j,i);*
    6.5: *else*
    6.6: *opp(j,i) = P(j,i);*
    6.7: *end*
    6.8: *end for*
    6.9: *end for*
7: Repeat steps 3 to 6 until the termination criterion is met.

4. Oppositional Pigeon-Inspired Optimizer Algorithm (Proposed)

The proposed Oppositional Pigeon-Inspired Optimizer algorithm is discussed in this section. The common search strategy of the proposed OPIO algorithm is like the PIO. However, the proposed OPIO algorithm utilizes a unique methodology to explore the search space of the pigeon, to discover the position of its hiding location. Moreover, the modified method provides better convergence in the pigeon population, which helps to achieve the optimal solution. As part of enhancement by the proposed method, in every iteration, the best pigeon is selected as the target. The selected pigeon position will be updated with the Oppositional-Based Learning, to enhance the convergence rate. However, selecting an arbitrary pigeon, from among the population, may result in a bad-quality landmark solution, with a large value for the fitness function (in the minimization problem), which leads to an unsuitable end point to move. In addition, selecting a random pigeon for the exploration phase will tend towards a bad destination, which minimizes the convergence rate. To select the best solution among the population at each iteration is a challenging task.

In this work, a priority-based election mechanism was introduced. This mechanism could be utilized for the minimization problem at each iteration for the pigeon i, so that ψ of the best pigeons in the solution set were elected. The benefit of this election mechanism was to elect the target pigeon among the list of the best pigeons in the stack. By this process, the pigeons could perform better in improvising their positions, by following the better target pigeons, and this resulted in a better convergence rate for the algorithm. Nevertheless, electing the value of ψ was significant: electing a very trivial value of ψ among the pigeons i could lead to being stuck in the local optima. In addition, selecting a large value for ψ could cause the bad target pigeon to be tricked. To eradicate these issues, in the initial iterations ψ started from a large value, for better diversification, and its number was reduced according to Equation (17); over the course of the iterations, its tendency towards the local optimum resulted in the ψ having a small value:

$$\psi^t = round\left(\psi_{max} - \frac{\psi_{max} - \psi_{min}}{N_g} \times t\right) \tag{17}$$

where, ψ^t stood for the value for selecting the best pigeon in iteration t, and ψ_{max} and ψ_{min} stood for the maximum and minimum values of ψ.

4.1. Constraint-Handling Technique

The ELD problem is complicated to solve, when considering the constraints. In past decades, various techniques have been adopted, to handle the constraints. The penalty function is considered to one of the most common constraint-handling techniques: it deals with the constraint problem by including some additional value to the objective function in (4). This function has been broadly utilized by various researchers, because of its simplicity and efficiency. The objective function is the minimization of the following representation:

$$F_{CN} = F_c + \varphi \left| \sum_{j=1}^{n} \Psi_j - \Psi_{Demand} - \Psi_{Loss} \right| \tag{18}$$

where F_{CN} stands for constraint-based objective function, and φ stands for the penalty coefficient of a real integer. If constraint (6) is other than zero, then the value of the second part in Equation (17) will be other than zero too, multiplied by the penalty value φ, and, finally, will be added to the fuel cost F_c. In other words, if Equation (6) does not meet the constraint, then this implies that the solution has a large objective function, and is likely to be rejected. On the other hand, if the solution meets the constraint (6), this implies that the solution holds a small objective function value, and is likely to be accepted. If the φ value is fixed with a large value, then the performance of the algorithm will be reduced, and this will lead to premature convergence. In addition, fixing the small value for φ fails to meet the inequality constraints.

4.2. Implementation of the OPIO Algorithm for the ELD Problem

In this section, the strategies for applying the OPIO algorithm, to solve the ELD problem, are examined. The main objective of the ELD optimization problem is to reduce the overall power generation cost. In the ELD problem, the total power generating unit (n) is proportional to the total decision variable of the optimization problem (d). Each position of the pigeon is represented as each anticipated power output of the generating units. In general, the ELD problem consists of some impartiality and disparity constraints, as discussed in Section 2.2. Each solution in the population should satisfy the constraints. For the smooth process of constraint handling, the value of φ is fixed as 100 in Equation (17) for the entire simulation, which attains an adequate performance with the power equality constraint.

The overall computational procedures of the proposed OPIO algorithm are described in detail as follows. In addition, the flowchart of the proposed OPIO algorithm is represented in Figure 1, and the proposed OPIO algorithm for solving the ELD problem is represented in Algorithm 3.

Step 1: Define the initial parameters with the characteristics of the generation units: φ; number of pigeons; maximum generations (n_g); other data, such as J_r, ρ.

Step 2: Initially, the arbitrary values for all generating units within the lower and upper operating boundary are generated using (5), except for the last generating unit. The computation of the last unit of power generation is calculated using (6), and it is validated, to ensure whether it satisfies the inequality constraints (5) or not. If the solution satisfies the constraints, then the solution is sustained; otherwise, it is abandoned. The pigeon position X, concerning the generating units, is initialized as follows:

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_i \\ \vdots \\ X_G \end{bmatrix} = \begin{bmatrix} X_{1,1} & \cdots & X_{1,b} & \cdots & X_{1,d} \\ \vdots & \ddots & \vdots & & \ddots \quad \vdots \\ X_{i,1} & \cdots & X_{i,b} & \cdots & X_{i,d} \\ \vdots & \ddots & \vdots & & \ddots \quad \vdots \\ X_{N_p,1} & \cdots & X_{N_p,b} & \cdots & X_{N_p,d} \end{bmatrix} i = 2, \ldots N_p; b = 1, 2, \ldots, d \tag{19}$$

where the component $X_{i,d}$ is the power outcome of the bth unit in individual X_i. For the OPIO algorithm, there is only one adjustable parameter: the jump-

ing rate J_r, which is fixed within the range of 0 to 0.4 for all test cases used in the experimentation.

Step 3: For each pigeon in the population, the power generating unit must satisfy the ramp-rate boundary, and not relay in the restricted operating zones. If the solution does not meet the constraints, then power outputs should be altered near to the boundary of the feasible solution. After processing the initialization, the main procedure of the OPIO algorithm process is as follows:

Step 4: Determine the velocity of the pigeon, using Equation (10), and update the position of the pigeon, using Equation (14). If the updated position of the pigeon does not satisfy the constraints, then alter the pigeon's position, as shown in Step 3.

Step 5: Compute the ψ factor, as in Equation (17).

Step 6: Choose the ψ of the best solutions from the population, and update the position for the selected pigeon, using the OBL technique (Algorithm 2).

Step 7: Check this step for the pigeon i:

a. The output power of the generating units must not reside in the RORs (see (8)) or contravene the operating unit limit (see (5)).
b. The lower and upper boundary rates of each of the generating units, from the preliminary state, should be in the satisfactory ranges, as given in (9). If the preliminary output power of the generating units is not specified, then the preliminary power of all power generating units should be within the satisfactory ranges.
c. If the RORs and ramp-rate limits are contravened, adjust the power outputs near to the feasible solution.

Step 8: Compute the overall power loss of the transmission lines for the pigeon i, as in (6).

Step 9: Compute the quality of the pigeon i, by interleaving its power outputs in the fitness function, as in (17).

Step 10: Repeat steps 4–9, until the stop criterion is met.

Step 11: The ELD solution is the best solution in the last iteration.

Algorithm 3: Proposed OPIO algorithm for solving ELD

1: Generate the initial population.
2: Determine the preliminary parameters.
3: Arbitrarily initialize the position of the pigeon in the search boundary space.
4: Check the RR and RORs constraints.
5: **While** (ng $\geq$ 1) **do**
6: Determine the velocity and position of the pigeon.
7: Determine the φ factor.
8: Select φ of the best pigeons from the population.
9: Apply OBL technique, using Algorithm 2.
10: **If** (*fit (Opp) < fit (X_(i,t))*) then
11: Replace the Opp solution
12: **Else**
13: do nothing
14: **End if**
15: Check the feasibility of the new position of the pigeons.
16: Calculate the transmission loss.
17: Evaluate the fitness of the new position of the pigeons.
18: Update the global best solution.
19: **End while**
20: **Output**: Visualize the global best solution.

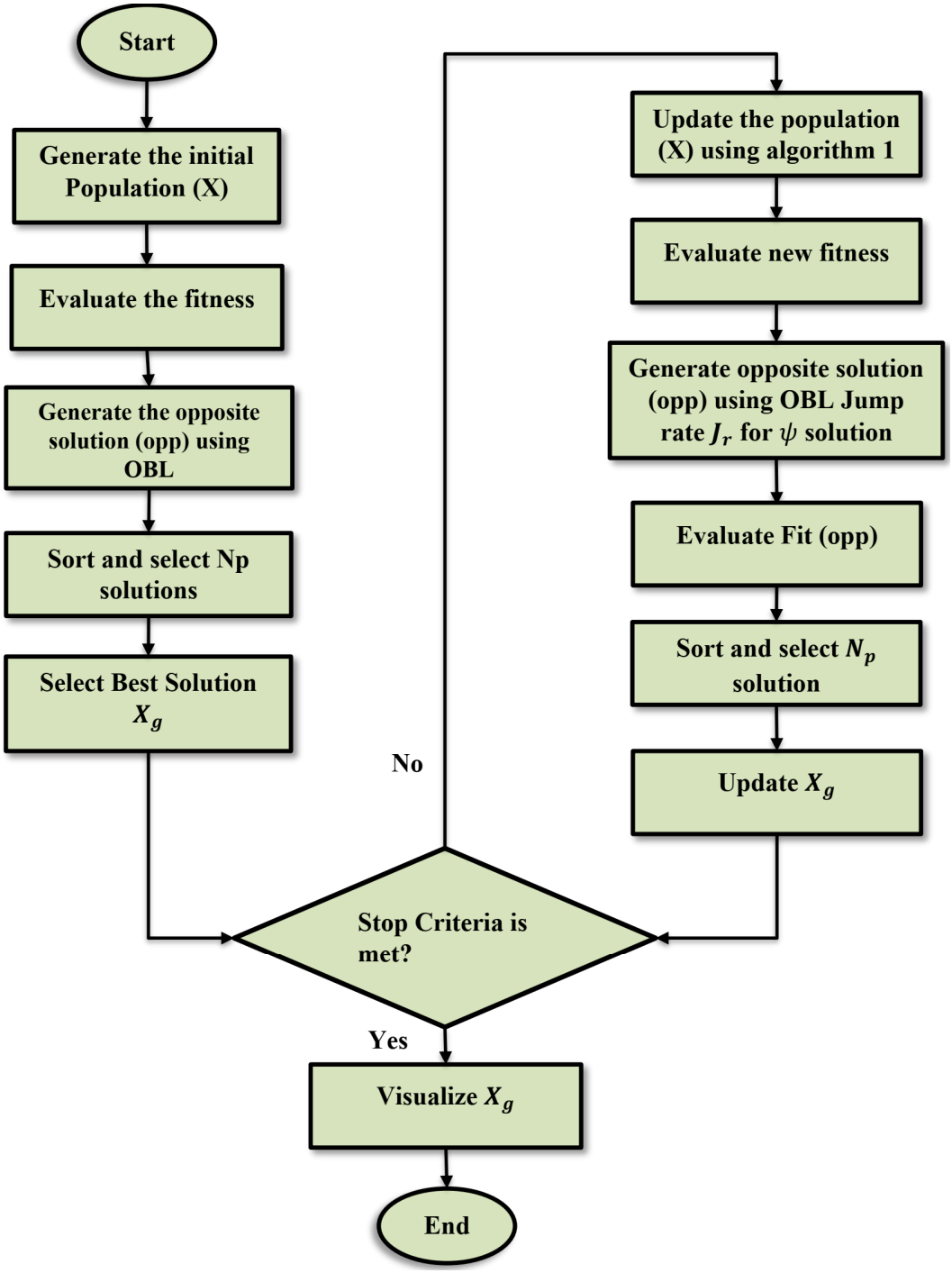

Figure 1. Flowchart of Proposed OPIO algorithm.

5. Results and Discussion

The proposed OPIO algorithm was applied to solve ELD issues. Three various test systems with three different fuel possibilities and non-linearities, such as ramp-rate ranges, value-point consequences and interdicted working region, were studied, to assess the execution of the formulated OPIO method. The formulated OPIO technique was written in MATLAB R2016a, implemented on a 2.6 GHz Intel I5 PC. The execution of the formulated OPIO algorithm was justified by utilizing three different test systems: small- (13-unit, 40-unit), medium- (140-unit, 160-unit) and large-scale (320-unit and 640-unit). The acquired outcomes from the formulated OPIO technique were differentiated to various state-of-the-

art metaheuristic techniques reported in the literature. The different test systems, with the number of generating units and their constraints, are outlined below:

(i) *Test Case 1: Small-Scale Test Systems (13-Unit and 40-unit)*
   a. 13-unit test case: in this test case, a 13-unit generator system, with constraints such as different fuel costs and value-point effects, was considered. The power load demand (P_D) of the system was fixed at P_D = 2520 MW [7,43];
   b. 40-unit test case: this test case held a 40-unit generator system, with value-point effects considered, and the power load demand of the system was fixed at P_D = 10,500 MW [7,43].

(ii) *Test Case 2: Medium-Scale Test Systems (140-unit and 160-unit)*
   a. 140-unit test case: in this test case, a 140-unit generator system, with constraints such as value-point effects, ramp-rate limits, and prohibited accomplishment unit, was considered. The power load demand (P_D) of the system was fixed at P_D = 49,342 MW [44];
   b. 160-unit test case: this test case held a 160-unit generator system, with value-point effects considered. The power load demand of the system was fixed at P_D = 43,200 MW [45].

(iii) *Test Case 3: Large-Scale Test Systems (320-unit and 640-unit)*
   a. 320-unit test case: a large-scale system with a 320-unit generator system, with different fuel options and value-point loading effects, was considered here. The power load demand of the system was fixed at P_D = 86,400 MW. The input data of the 10-unit system were duplicated 32 times in this system [46].
   b. 640-unit test case: a test case with a 640-unit generator system, with multiple fuel options and value-point load effects, was considered here. The load demand of the system was increased by up to P_D = 1,72,800 MW. The input data of the 10-unit system were replicated 64 times in this system [30].

The convergence of metaheuristic algorithms mainly relies on the possibility of a proper value. The proposed technique may deliver a different solution when the choice of insert value is not appropriate. To select the proper input parameters, repeated simulation is required. For the OPIO algorithm, after a repeated number of runs, the lower and upper jumping rates were fixed within the range of 0 to 0.4. For effective simulation, we considered a population size of 50, and 100 was selected as the maximum number of iterations for the test systems.

5.1. Test Case 1a: 13-Unit

In this instance, the formulated OPIO technique was tested on a small-scale 13-unit system, which held uneven fuel cost and value-point effects. The dataset of the fuel cost and the limit utility of numerous vigorous energy providers were taken from [43], and the load order was fixed as 2520 MW. To examine the execution of the proposed OPIO technique and the conventional PIO algorithm, the assumed outcomes were differentiated from the various metaheuristic algorithms, viz., Oppositional Grey Wolf Optimization (OGWO) [32], Improved Particle Swarm Optimization (IPSO) [29], One Rank Cuckoo Search Algorithm (ORCSA) [36], Crisscross Optimization Algorithm (COA) [30], Real-Coded Genetic Algorithm (RCGA) [37], Improved Genetic Algorithm (IGA) [38] and Pigeon-Inspired Optimization (PIO) [39].

Table 1 provides the comparative results of the OPIO and PIO algorithms for active power generators along with other techniques. As shown in Table 1, the solution provided by the OPIO algorithm reached a fuel cost of 24512.45$/hr, which was less than all the compared algorithms; the outcomes of the formulated techniques conveyed that it was superior in finding the best or near-best solution. To ensure the efficacy and effectiveness of the technique, the simulation was carried out over 100 runs, on both the proposed OPIO algorithm and the conventional PIO algorithm, and its result is given in Table 2. As shown

in Table 2, the OPIO produced a better solution for 97 runs, which was far better than all compared algorithms. The statistical outcomes conveyed that the formulated OPIO algorithm delivered better results compared with various algorithms. The convergence of the minimization fuel-cost function over the iteration cycles of the proposed OPIO algorithm and the standard PIO algorithm were noted, and are displayed in Figure 2. Figure 2 shows that the proposed algorithm converged faster towards the optimal solution that did not have further changes, which validated the active constancy of the formulated technique.

Table 1. Test outcomes of various algorithms for a 13-unit system with PD = 2520 MW.

Unit	OGWO	IPSO	COA	ORCSA	PIO	OPIO (Proposed)
1	628.2948	628.1678	628.3451	628.4524	628.3124	628.5647
2	299.0451	298.8798	298.5478	298.3575	298.3567	298.9856
3	296.4501	297.6984	297.6874	297.3457	297.4254	297.6245
4	159.6421	159.2387	159.3564	159.2349	159.6548	159.7542
5	159.7154	159.1254	159.8957	159.5796	159.5347	159.6589
6	159.5484	159.3567	159.3567	159.2134	159.8975	159.3521
7	159.6879	159.8954	159.6542	159.6875	159.7543	159.1256
8	159.6877	159.6872	159.6513	159.3579	159.5421	159.2564
9	159.6542	159.9877	159.3542	159.7765	158.8578	159.3658
10	76.4854	77.6513	77.6854	77.5587	77.3567	77.6574
11	114.8742	113.3685	114.2314	114.2254	113.3687	114.8975
12	91.5874	92.6975	92.8674	92.6478	92.6898	92.8785
13	92.5412	92.3515	92.4578	92.3542	92.3277	92.8547
Fuel Cost ($/h)	24,513.4847	24,514.6875	24,512.8754	24,513.5464	24,514.5467	**24,512.4578**
Power loss (MW)	40.2975	40.3051	40.3645	40.3897	40.5781	**40.1584**

Table 2. Comparison outcomes of different algorithms for a 13-unit system.

Algorithms	Best ($/H)	Mean ($/H)	Worst ($/H)	Standard Deviation	Successful Runs (%)	Test time (S)
OGWO	24,512.72	24,512.86	24,514.65	0.1031	92	5.89
IPSO	24,517.68	24,517.96	24,518.21	0.3154	84	5.98
IGA	24,516.42	24,517.76	24,519.78	NA	82	6.21
RCGA	24,514.54	24,515.87	24,517.89	0.1578	88	6.89
COA	24,512.87	24,513.65	24,515.68	0.1047	93	5.47
ORCSA	24,513.54	24,513.54	24,516.67	NA	87	8.65
PIO	24520.54	24521.75	24532.95	0.2645	79	11.00
OPIO (Proposed)	24512.45	**24512.67**	24513.54	0.0875	97	5.14

5.2. Test Case 1b: 40-Unit

To access the feasibility of the proposed OPIO algorithm, another small-scale test case, of a 40-unit power generation system along with value-point belongings, was used. The benchmark value of the 40-unit power system was approached from [43], and its load demand was fixed as 10,500 MW. The outputs of the power generation and fuel cost of various algorithms like OGWO, IPSO, IGA, RCGA, COA, ORCSA, PIO and OPIO are shown in Table 3: the best cost of the PIO and OPIO algorithms reached 136,588.57 $/h and 136,447.87$/h, respectively; it is also notable that the OPIO algorithm provided the best solution among the compared techniques, by achieving the load demand and other constraints.

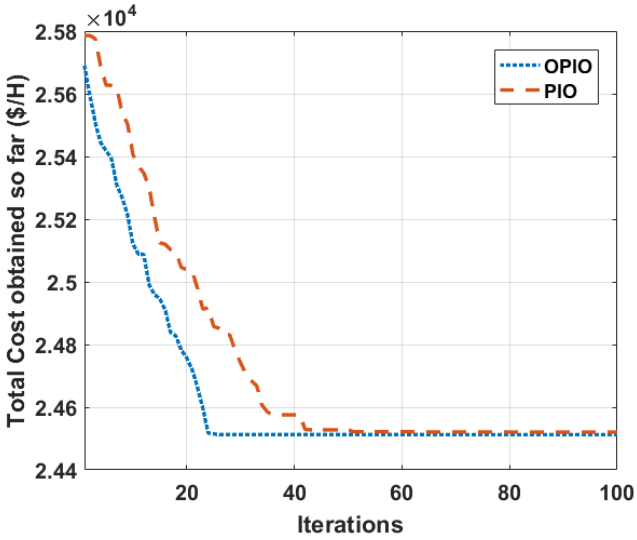

Figure 2. Convergence results of the OPIO and PIO algorithms for a 13-unit system.

Table 3. Simulation outcomes of different algorithms for a 40-unit system with PD = 10,500 MW.

Unit	OGWO	IPSO	COA	ORCSA	PIO	OPIO (Proposed)
1	114.2743	114.2876	113.8502	110.12	111.52	114.24
2	114.4501	114.4621	114.1203	112.28	112.34	114.12
3	120.3567	120.4212	119.7458	120.23	119.28	120.31
4	183.3685	181.5412	182.4127	188.54	182.45	190.54
5	87.1256	87.3542	88.5864	85.37	87.34	97.56
6	140.3645	140.3747	140.3289	140.24	139.52	140.25
7	300.1254	300.2346	299.6517	250.28	198.24	300.54
8	300.2349	300.3277	292.3428	290.74	186.38	300.26
9	300.4501	300.4578	299.6433	300.52	193.12	300.49
10	279.0451	279.3874	279.5423	282.31	179.41	205.49
11	243.3277	243.6752	168.2597	180.25	162.27	226.47
12	94.5874	94.3259	94.2355	168.52	94.39	204.56
13	484.3051	484.4578	484.2511	469.78	486.22	346.52
14	484.1584	484.3277	484.6425	484.26	487.33	434.58
15	484.2314	484.3542	484.3266	487.39	483.26	431.29
16	484.5412	484.1564	484.6501	482.62	484.25	440.21
17	489.5781	489.6475	489.4523	499.16	494.61	500.34
18	489.4578	489.5423	489.6244	411.19	489.76	500.33
19	511.8785	511.6432	511.1289	510.27	512.34	550.27
20	511.8754	511.5428	511.6451	542.37	513.21	550.96
21	523.3228	523.5746	549.3347	544.29	543.18	550.14
22	546.3738	547.7433	549.6455	550.29	548.38	550.54
23	523.1035	523.4728	523.4589	550.37	521.56	550.32
24	523.0678	523.1532	523.4313	528.18	525.23	550.46
25	523.5181	523.4421	523.1204	524.67	529.67	550.28
26	523.4767	523.4775	523.4217	539.28	540.31	550.39
27	10.3344	10.1067	10.1265	10.34	12.46	11.27
28	10.8011	10.7836	10.1024	10.24	10.96	11.34
29	10.6445	10.4521	10.2301	10.22	10.34	11.16
30	87.302	87.9827	87.6658	96.42	89.45	97.16
31	190.5847	190.2498	190.1322	185.24	189.04	190.34
32	190.8664	190.3277	189.4582	189.26	189.47	190.28
33	190.9983	190.4562	190.4251	189.37	187.43	190.64
34	200.5471	200.2841	199.2217	199.16	198.27	200.34
35	200.5847	200.6128	200.7541	196.54	199.69	200.41
36	164.9983	164.9833	164.3242	185.28	165.34	200.67
37	110.2147	110.2348	110.2323	109.58	109.54	110.24
38	110.3341	110.4355	109.2344	110.76	109.31	110.11
39	110.4616	110.5436	110.3333	95.17	109.44	110.37
40	511.9904	511.2239	550.4219	532.59	548.23	550.16
Fuel Cost ($/h)	136,441.8527	136,446.7842	136,442.689	136,549.8756	136,588.5746	136,441.876
Power loss (MW)	964.75	963.2045	945.2143	958.39	979.85	940.12

The comparative outcomes of the overall fuel cost, success rate, standard deviation and execution time acquired by the OPIO algorithm, along with the various techniques, are given in Table 4. Based on Table 4, the OPIO algorithm achieved the best solution 96 times out of 100 trials. In addition, the mean costs of the OPIO and IPSO algorithms were equal to 136,441.87$/h and 136,542.87$/h, respectively. This clearly shows that the statistical outcomes of the OPIO algorithm were more stable than those of the OGWO, IPSO, COA, RCGA, ORCSA and PIO algorithms. In addition, the time required to achieve the minimal fuel cost for the proposed algorithm was 10.14/sec, which was minimal in relation to other algorithms. The convergence graph of the total fuel cost of the proposed OPIO algorithm and the conventional PIO algorithm is given in Figure 3. Based on Figure 3, it can be seen that the formulated OPIO procedure provides the best active rate compared to the PIO algorithm.

Table 4. Comparison results of various algorithms for a 40-unit system.

Algorithms	Best ($/H)	Mean ($/H)	Worst ($/H)	Standard Deviation	Successful Runs (%)	Test Time (S)
OGWO	136,441.85	136,445.87	136,447.54	0.1365	94	11.52
IPSO	136,446.78	136,542.87	136,588.55	0.2345	89	12.65
IGA	136,454.56	NA	NA	NA	NA	NA
RCGA	136,587.21	136,687.52	136,742.65	0.3874	84	13.41
COA	136,442.68	136,448.54	136,468.54	0.1865	92	12.87
ORCSA	136,549.87	NA	NA	NA	NA	NA
PIO	136,588.57	136,698.32	136,721.54	NA	NA	NA
OPIO (Proposed)	136,441.87	**136,441.95**	136,443.81	0.1021	96	10.14

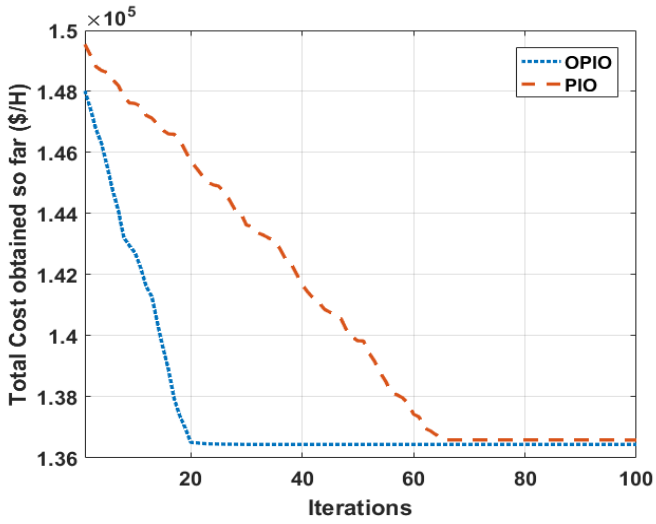

Figure 3. Convergence results of the OPIO and PIO algorithms for a 40-unit system.

5.3. Test Case 2a: 140-Unit

In this instance, the formulated PIO algorithm was tested on the medium-scale of a 140-unit power generation system, and the load order was taken as 49,342 MW [46]. In this test case, non-smooth constraints, such as value-point consequence, interdicted executing section and ramp-limits were included. The execution was repeated for 100 trials, to confirm the dominance of the proposed methods with the obtained results of the OGWO, IPSO, COA, RCGA, ORCSA and PIO algorithms, which are presented in Table 5. As shown in Table 5, the OPIO reached 1,559,498.78$/h, which was the minimum, compared to the other algorithms. In other words, the obtained outcomes clearly showed that the OPIO algorithm achieved a low fuel-cost value, compared to other methods.

Table 5. Test outcomes of various algorithms for a 140-unit system with PD = 49,342 MW.

Unit	PIO	OPIO (Proposed)	Unit	PIO	OPIO (Proposed)	Unit	PIO	OPIO (Proposed)
1	114.3542	119.1244	48	250.2564	250.4159	95	978.1244	978.2456
2	189.2341	189.2344	49	250.3577	250.3648	96	682.3277	682.1247
3	190.2134	190.2333	50	250.4525	250.3014	97	720.2441	720.1689
4	190.2625	190.2455	51	165.2134	165.4258	98	718.2355	718.2245
5	168.7569	168.6479	52	165.3426	165.2486	99	720.2466	720.3144
6	190.2345	190.3247	53	165.5412	165.4857	100	964.2344	964.2188
7	490.2625	490.2655	54	165.5122	165.5574	101	958.3477	958.2177
8	490.3438	490.3677	55	180.3211	180.2675	102	1007.1255	1007.5248
9	496.5255	496.5829	56	180.2144	180.5974	103	1006.3425	1006.7413
10	496.5648	496.5574	57	103.2644	103.4428	104	1013.6487	1013.6944
11	496.6522	496.6548	58	198.4522	198.5674	105	1020.6666	1020.1024
12	496.7566	496.7742	59	312.3248	312.4295	106	954.2377	954.2188
13	506.2256	506.2389	60	280.9517	282.5479	107	952.1244	952.1277
14	509.3364	509.4861	61	163.3211	163.4287	108	106.3244	1006.2384
15	506.2355	506.2479	62	95.2347	95.2261	109	1013.2311	1013.5244
16	505.2144	505.2498	63	160.2358	160.4521	110	1021.6322	1021.5644
17	506.3422	506.3451	64	160.3459	160.2349	111	1015.2344	1015.2988
18	506.9548	506.4692	65	490.2378	490.2587	112	94.2155	94.2577
19	505.2344	505.6498	66	196.4356	196.5477	113	94.3157	94.2188
20	505.3214	505.4582	67	490.5612	490.6287	114	94.4233	94.1024
21	505.4255	505.2398	68	490.6458	489.3017	115	244.5612	244.1657
22	505.5412	505.2179	69	130.2648	130.2688	116	244.6124	244.1287
23	505.5378	505.2489	70	234.8465	234.5144	117	244.8477	244.5218
24	505.6522	505.9548	71	137.2659	137.2955	118	95.3244	95.2476
25	537.4612	537.2144	72	325.5641	325.4872	119	95.6245	95.2348
26	537.2344	537.3499	73	195.2347	195.3488	120	116.2377	116.2478
27	549.6254	549.2674	74	175.4529	175.6247	121	175.4298	175.3489
28	549.3244	549.3014	75	175.2349	175.4277	122	2.2144	2.1758
29	501.2333	501.2648	76	175.8945	175.6648	123	4.3322	4.1689
30	501.4622	501.3478	77	175.5217	175.2486	124	15.4625	15.4873
31	506.2477	506.2018	78	330.2175	330.2487	125	9.2344	9.2481
32	506.2344	506.3014	79	531.2648	531.1248	126	12.9588	12.6475
33	506.2237	506.4251	80	531.2647	531.2476	127	10.2647	10.6598
34	506.8546	506.6517	81	398.4275	436.2186	128	112.6599	112.4581
35	500.2649	500.2689	82	56.1975	56.2874	129	4.2689	43275
36	500.2014	500.2478	83	115.2348	115.3495	130	5.2644	5.1694
37	241.3645	241.2349	84	115.4756	115.6248	131	5.6544	5.4572
38	241.6477	241.2847	85	115.2679	115.7749	132	50.2688	50.6489
39	774.2655	774.5842	86	207.6548	207.4568	133	5.2177	5.1483
40	769.8542	769.4158	87	207.1673	207.1168	134	42.6588	42.5976
41	3.9655	3.2685	88	175.9485	175.4906	135	42.7588	42.6577
42	3.8522	3.1749	89	175.2648	175.2681	136	41.2355	41.6572
43	250.3466	250.3489	90	175.3348	175.6489	137	17.6588	17.2954
44	246.5147	249.5681	91	175.2247	175.2213	138	17.4955	17.3597
45	250.3416	250.3189	92	580.2234	580.6477	139	7.2655	7.2265
46	250.3429	250.6475	93	645.1958	645.2188	140	26.5884	26.4621
47	240.3168	249.6257	94	984.2655	984.2377	Fuel Cost ($/H)	1,560,412.55	1,559,498.78

Moreover, the statistical outcomes of the formulated OPIO algorithm, and various conventional procedures, are given in Table 6. Based on Table 6, the formulated OPIO algorithm provided the best outcomes, in terms of best, worst and mean cost, and less execution time compared to the various procedures. However, the best and mean costs of the OPIO and COA algorithms were equal to 1,559,498.78 $/h, 1,559,499.21 $/h, 1,559,521.36 $/h and 1,559,521.88 $/h, respectively. Even though the COA algorithm competed with the OPIO algorithm, the OPIO algorithm was quite efficient in achieving the best outcome in minimal

iterations, compared to the various procedures. The convergence of the formulated OPIO algorithm and the conventional PIO methods with iteration cycles is displayed in fig 4. From Figure 4, it can be seen that the OPIO technique attained the best solution within 20 iterations; this confirms that the OPIO algorithm had better convergence, because of its magnificent diversification and intensification abilities.

Table 6. Statistical comparison results of test case 2a (140-unit system with PD = 49,342 MW).

Algorithms	Best ($/H)	Mean ($/H)	Worst ($/H)	Standard Deviation	Successful Runs (%)	Test Time (S)
OGWO	1,559,710.65	1,559,715.64	1,559,751.21	0.1512	95	43.98
IPSO	1,560,453.89	NA	NA	NA	NA	NA
IGA	1,561,254.85	NA	NA	NA	NA	NA
RCGA	1,559,957.62	1,560,521.35	1,561,542.96	0.5641	84	49.54
COA	1,559,499.21	1,559,521.88	1,559,645.21	0.1234	92	44.66
ORCSA	1,559,987.42	1,560,387.36	1,561,662.54	NA	NA	NA
PIO	1,560,412.55	1,561,542.13	1,562,874.62	NA	NA	NA
OPIO (Proposed)	1,559,498.78	**1,559,521.36**	1,559,587.62	0.1123	96	40.21

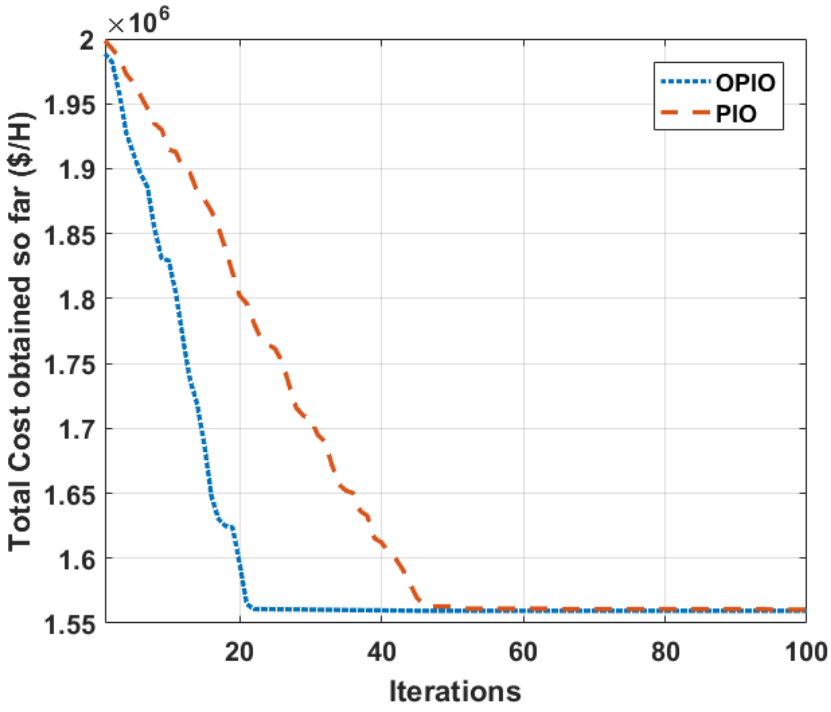

Figure 4. Convergence results of the OPIO and PIO algorithms for a 140-unit system.

5.4. Test Case 2b: 160-Unit

To access the feasibility of the formulated OPIO technique, another medium-scale test case of a 160-unit test system, along with non-convex value-point properties, was used. As to validation, the viability and efficacy of the formulated technique transmission loss was unnoticeable. For this medium-scale unit, a replicated 10 different fuel-option values were taken from [41], the power load was increased by 16, and the power load was fixed as 43,200 MW. Table 7 provides the attained better cost of the proposed OPIO algorithm, with other algorithms, by satisfying the constraints. Based on Table 7, the OPIO achieved 9625.15 $/h, which was the best result, compared to the other algorithms. This confirms that the least total fuel cost was for the 160-unit generation system.

Table 7. Test outcomes of various algorithms for 160-unit system with PD = 43,200 MW.

Unit	PIO	OPIO (Proposed)	Unit	PIO	OPIO (Proposed)	Unit	PIO	OPIO (Proposed)
1	211.1057	224.3218	55	207.3485	265.4215	109	402.5278	430.2109
2	208.4572	200.6548	56	254.6289	257.1026	110	272.9458	260.5614
3	335.6534	355.2671	57	294.3275	277.3041	111	199.2658	217.4259
4	243.9547	228.4601	58	245.2964	235.4216	112	204.7594	199.6504
5	266.1958	305.4286	59	420.3581	394.5027	113	251.4298	357.2169
6	237.4295	249.5013	60	270.1485	278.1659	114	249.5048	251.4209
7	282.1473	309.4681	61	198.3475	240.3415	115	266.4175	267.2044
8	239.6475	218.4627	62	212.3458	213.1452	116	240.5219	252.3011
9	408.6427	335.2485	63	348.2571	241.6521	117	290.3584	290.4255
10	265.4581	270.4195	64	259.2543	248.6574	118	242.1507	233.4452
11	225.1049	187.2589	65	292.8594	232.2485	119	412.6259	329.5622
12	217.4685	195.4271	66	221.6579	245.6574	120	242.4957	300.4586
13	336.4216	353.2049	67	286.5419	310.2415	121	211.1048	237.5248
14	232.4582	241.6204	68	242.3859	232.5218	122	210.6247	207.6648
15	259.3248	273.4209	69	348.5796	353.2016	123	245.6284	237.5601
16	237.4581	228.1064	70	288.6473	281.4025	124	227.2094	210.6598
17	265.3482	277.4295	71	227.4961	219.4259	125	273.4587	241.2045
18	236.5219	224.1058	72	215.3333	220.4015	126	250.6418	246.5129
19	414.2188	404.6257	73	339.4857	343.2016	127	258.6458	293.3277
20	272.3158	314.2045	74	244.6589	250.4012	128	243.4109	245.2011
21	222.2507	205.4952	75	280.4276	273.5248	129	382.4692	431.2655
22	204.3258	200.4672	76	231.5942	217.5486	130	296.4108	248.5555
23	333.2429	374.1526	77	286.4957	305.4295	131	200.1472	199.7468
24	237.3854	234.1059	78	225.3489	243.4582	132	214.2845	217.4511
25	304.5521	284.1057	79	282.6472	253.6241	133	334.2074	242.6589
26	245.2965	221.4695	80	265.3485	271.9648	134	234.6248	229.3544
27	265.5421	253.1284	81	223.4685	223.4258	135	286.3049	250.3014
28	237.4951	223.6244	82	200.1479	191.3045	136	247.1658	236.1045
29	381.2659	241.5019	83	334.6517	349.5261	137	290.6547	305.1588
30	267.3204	280.6642	84	246.5923	234.5201	138	229.3104	230.4266
31	223.1584	175.4269	85	274.2986	275.2496	139	391.6248	430.2984
32	214.6549	203.6248	86	249.1634	208.4257	140	270.6248	257.1659
33	334.5671	348.2015	87	297.5842	282.5967	141	212.4286	228.4358
34	247.9547	219.5202	88	240.6713	255.3048	142	226.1958	184.2384
35	250.4682	283.4029	89	340.2986	409.4672	143	335.2648	370.5691
36	229.4572	244.1527	90	291.4726	269.3541	144	245.1382	219.4581
37	265.4953	313.2658	91	212.4589	192.3547	145	261.2017	263.1594
38	243.1572	236.1259	92	210.4976	201.1064	146	244.6218	235.4275
39	413.2685	340.2015	93	337.4682	350.2496	147	294.3581	300.6598
40	277.5878	333.3541	94	236.4196	258.6412	148	235.2481	242.1574
41	202.1574	230.5298	95	261.77165	290.3485	149	390.2571	387.4598
42	228.6257	209.9654	96	235.6279	226.4153	150	253.4192	267.4125
43	352.1469	341.4027	97	278.9648	299.4035	151	193.2488	184.5298
44	227.4583	233.6248	98	237.5944	223.4257	152	222.5027	212.4158
45	272.4159	299.1047	99	420.6519	366.6591	153	355.9418	376.1548
46	241.6051	263.5218	100	268.1695	288.4251	154	242.3158	248.5476
47	266.5384	243.6201	101	224.3186	227.2045	155	277.6428	278.1549
48	230.4572	212.3048	102	209.6581	210.5499	156	246.7128	223.2435
49	423.6514	362.4295	103	337.9547	363.2011	157	318.5472	264.6248
50	254.9571	303.2048	104	230.9528	222.3495	158	241.6014	234.5278
51	280.9654	181.4269	105	269.4681	234.1058	159	341.6547	390.4855
52	211.4582	211.5274	106	235.6428	232.4259	160	261.2485	286.4597
53	337.4692	364.6542	107	274.9648	282.3045	Fuel Cost ($/H)	9738.4526	9625.1573
54	238.4729	234.9512	108	244.3018	236.4159			

The statistical results from over 100 trials of the proposed algorithm, compared to the OGWO, IPSO, IGA, RCGA, COA, ORCSA and PIO algorithms, are shown in Table 8: as can be seen, the OPIO algorithm performance—for example, best (9625.44 $/h), mean (9647.62 $/h) and worst (9649.62 $/h)—was relatively acceptable, whereas the other algorithms deteriorated, due to an increase in the number of generators and traps in the locally optimal solutions. As per the acquired outcomes, we observed that the formulated OPIO technique was more vigorous and systematically structured, compared to the conventional and various algorithms. The active loop of formulated technique and conventional algorithm with iteration cycle is displayed in Figure 5. Figure 5 shows that the formulated procedure provided feasible convergence within 25 iterations, though there was an increase in the number of generation units compared to the standard PIO algorithm.

Table 8. Statistical comparison results for test case 2b (160-unit with PD = 43,200 MW).

Algorithms	Best ($/H)	Mean ($/H)	Worst ($/H)	Standard Deviation	Successful Runs (%)	Test time (S)
OGWO	9768.62	9772.21	9774.62	0.1421	94	18.52
IPSO	10,008.65	10,009.65	10,010.56	0.3654	85	36.95
IGA	10,009.82	10,010.98	10,011.26	NA	NA	NA
RCGA	9954.23	10,002.62	10,004.63	0.4521	90	27.62
COA	9664.32	9702.36	9776.85	0.1262	94	16.85
ORCSA	9845.45	9898.12	9902.42	NA	NA	NA
PIO	9738.45	9812.64	9836.95	0.3641	88	44.34
OPIO (Proposed)	9625.15	**9647.62**	9649.62	0.1145	96	16.21

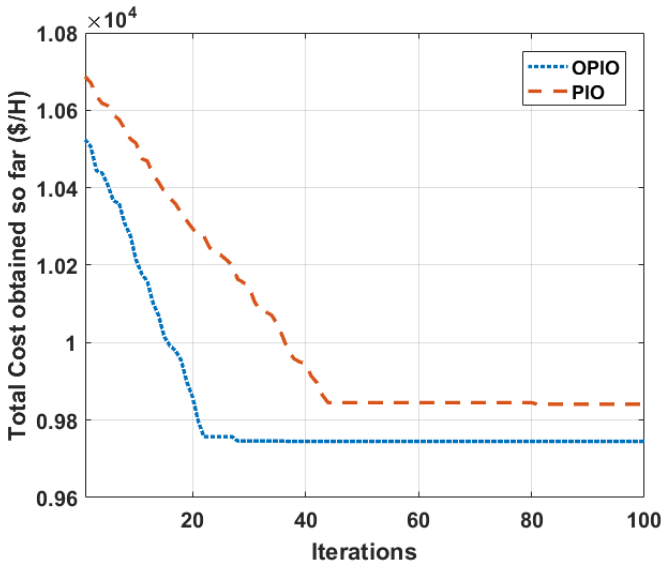

Figure 5. Convergence results of the OPIO and PIO algorithms for a 160-unit system.

5.5. Test Case 3a: 320-Unit

In this instance, a wide scale 320-unit generation system, that included a value-point effect and three various fuel possibilities, was used to evaluate the execution of the formulated OPIO technique. For this 320-unit system, 32 times replicated, 10 different fuel options were taken from [41], and the power load was considered as 86,400 MW. Simulation results were carried out for 1000 iterations, for the 320-unit generation system, and its comparative results are illustrated in Table 9. Table 9 shows that the OPIO algorithm provided 19,968.95$/h, which was the minimal production cost compared to different state-of-the-art algorithms. On the other hand, the test times of the COA and OPIO algorithms were nearly equal, at 412.95/sec and 410.65/sec, respectively. However, the OPIO algorithm showed a unique performance, in attaining the best fuel cost for 96 runs out of 100 trials. This proves

that the formulated OPIO algorithm was vigorous, and deliberately well-organized, compared to the PIO algorithm and the different approaches presented in this study. To ensure the efficacy of the formulated OPIO technique, the convergence over different iterations is shown in Figure 6. From Figure 6, it can be seen that the execution of the OPIO technique provided the best convergence over the standard PIO algorithm.

Table 9. Statistical comparison results of test case 3a (320-unit system with PD = 86,400 MW).

Algorithms	Best ($/H)	Mean ($/H)	Worst ($/H)	Standard Deviation	Successful Runs (%)	Test Time (S)
OGWO	19,985.62	19,989.41	19,992.34	0.5465	86	489.35
COA	19,971.43	19,986.37	19,990.75	0.7248	92	412.95
ORCSA	20,045.29	NA	NA	NA	NA	NA
PIO	20,254.42	20,267.95	20,312.61	NA	NA	527.24
OPIO (Proposed)	19,968.95	**19,972.16**	19,978.91	0.4087	96	410.65

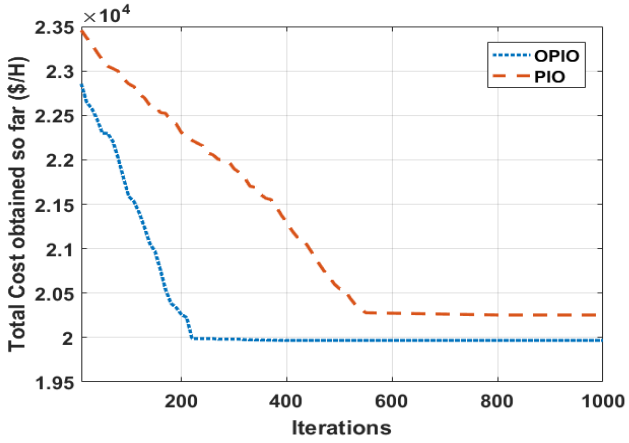

Figure 6. Convergence results of the OPIO and PIO algorithms for a 320-unit system.

5.6. Test Case 3b: 640-Unit

To ensure the efficacy of the formulated OPIO method, we tested it on another large-scale generation system, a 640-unit system with value-point properties and three various fuel possibilities. This 640-unit system included the data of 10 multiple-fuel systems from [41], which were duplicated 64 times, and the load demand was fixed at 172,800 MW. The simulation results of the 640-unit system were iterated over 1000 iterations, and its comparative results are shown in Table 10, where it can be seen that the OPIO algorithm achieved minimum fuel cost related to the various state-of-the-art techniques. The statistical results achieved, by performing 100 trials of the different algorithms and their comparative outcomes, are illustrated in Table 10, which shows that the OPIO algorithm reached 39,963.78 $/h by balancing the local search and global search, as well as converging faster towards the optimal solution. Moreover, the OPIO algorithm achieved the best solution for almost 96 runs out of 100 trials, which clearly demonstrates that the proposed algorithm can sustain the best position for various runs. The convergence results of the proposed OPIO algorithm and the PIO algorithm are displayed in Figure 7. In Figure 7, the formulated OPIO technique provided better convergence, which demonstrates its superiority over the standard PIO algorithm and other state-of-the-art techniques. The overall experimentation outcomes convey that the proposed OPIO algorithm achieved better efficiency, along with a trade-off between exploration and exploitation.

Table 10. Comparison results of various algorithms on a 640-unit system.

Algorithms	Best ($/H)	Mean ($/H)	Worst ($/H)	Standard Deviation	Successful Runs (%)	Test time (S)
OGWO	40,123.65	40,132.85	40,152.18	0.8542	90	704.85
COA	39,968.81	39,970.52	39,974.32	0.3419	94	682.54
ORCSA	40,189.62	NA	NA	NA	NA	NA
PIO	41,072.28	NA	NA	NA	NA	NA
OPIO (Proposed)	39,963.78	39,964.75	39,967.82	0.2451	96	677.27

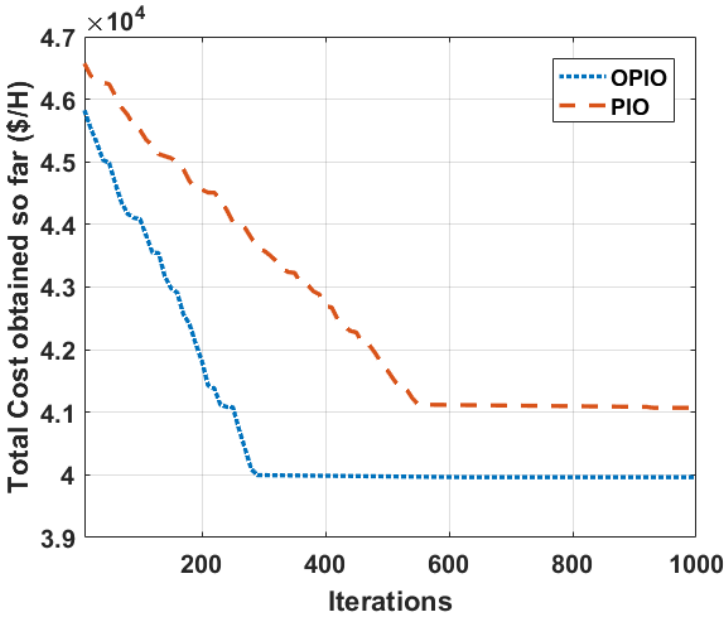

Figure 7. Convergence results of the OPIO and PIO algorithms for a 640-unit system.

5.7. The Result Analysis of Wilcoxon Signed-Rank Test

In this work, a non-parametric test—namely, the Wilcoxon signed-rank test—was utilized, to perform the statistical comparison of the proposed algorithm with the compared algorithms. The best solutions were attained by each technique for the corresponding test cases during 30 independent runs. In this study, the Wilcoxon signed-rank test was performed with a significance level $\alpha = 0.05$. The results, analyzed by the Wilcoxon signed-rank test, are presented in Table 11 for test cases of 13, 40, 140, 160, 320 and 640-generating units. In Table 11, the significance differences of the proposed algorithm and compared algorithms are marked with the value of H (i.e., H with a value of 1 specifies that there was a significance difference; otherwise, the H value is 0, if there was no significance difference). In addition, the symbol S with "+", "=" and "_" denotes that the proposed technique was superior, equal or inferior, respectively, to the compared algorithms. Furthermore, we used four compared algorithms generically, to determine the significance difference with the proposed algorithm. It is clear from Table 11 that the proposed OPIO algorithm provided results superior to those of the COA, ORCSA and PIO algorithms, and equal to the OGWO algorithm for the test case 13-unit system. For the test case 40-unit system, the OPIO algorithm provided results superior to those of the COA, ORCSA and PIO algorithms, but not to the OGWO algorithm. Finally, w/t/l specified the win/tie/loss count by Wilcoxon signed-rank test for the six test case generating unit systems. Thus, from the above discussion, it is clear that the proposed OPIO algorithm attained better solutions, and had better exploring capability, compared to the existing algorithms.

Table 11. Wilcoxon signed-rank test between OPIO and four compared algorithms for test case 13, 40, 140, 160, 320 and 640-unit systems.

Test Case	OPIO vs											
	OGWO			COA			ORCSA			PIO		
	p-Value	H	S	*p*-Value	H	S	*p*-Value	H	S	*p*-Value	H	S
13-unit	1.00×10^{0}	0	=	1.88×10^{-6}	1	+	1.51×10^{-6}	1	+	1.98×10^{-6}	1	+
40-unit	3.85×10^{-1}	0	–	1.00×10^{0}	0	=	1.88×10^{-6}	1	+	1.75×10^{-6}	1	+
140-unit	1.87×10^{-6}	1	+	1.70×10^{-6}	1	+	1.46×10^{-6}	1	+	1.65×10^{-6}	1	+
160-unit	1.55×10^{-6}	1	+	1.00×10^{0}	0	=	1.65×10^{-6}	1	+	1.75×10^{-6}	1	+
320-unit	1.73×10^{-6}	1	+	1.75×10^{-6}	1	+	1.75×10^{-6}	1	+	1.79×10^{-6}	1	+
640-unit	1.55×10^{-6}	1	+	1.11×10^{-6}	1	+	1.73×10^{-6}	1	+	1.75×10^{-6}	1	+
w/t/l	4/1/1			4/2/0			6/0/0			6/0/0		

6. Conclusions and Future Work

In this article, we have provided a novel metaheuristic algorithm named the Oppositional Pigeon-Inspired Optimizer (OPIO), which is formulated to deal with the ELD problem, with value-point consequences and numerous fuel possibilities. From the literature, it can be seen that the standard PIO algorithm is considered a promising optimization technique, which attracts the researcher by its superiority in addressing various optimization problems. However, it suffers in regard to global search ability and premature convergence when it is applied to large-scale optimization problems. Because of these issues, we merged Opposition-Based Learning into a standard PIO algorithm, which helped to eradicate early convergence, aided knowledge discovery and enhanced comprehensive searchability. The formulated OPIO algorithm was applied to non-convex ELD problems with different constraints, such as multiple fuel possibilities, value-point consequence, interdicted zones and ramp-rate. The experimentation was carried out on three different ELD test cases, viz., small-scale (13-unit and 40-unit), medium-scale (140-unit and 160-unit) and large-scale (320-unit and 640-unit) test cases. The exploratory outcomes showed the superiority of the formulated OPIO technique—in relation to higher potential solutions, better convergence rate, robustness and better computational efficiency—over the PIO algorithm and other state-of-the-art metaheuristic algorithms. In future, this work could be used in other fields of optimization, owing to the technique's high potential for dealing with the problematic optimization issues of many practical power systems. In addition, the outcome of the results can be compared with potential algorithms such as SEPSO [16], SA-QSFS [42] and QANA [47].

Author Contributions: Conceptualization, R.R.; methodology, R.R. and D.K.; validation, S.S.A. and M.R.; formal analysis, A.D.; writing—original draft preparation, R.R.; writing—review and editing, M.R. and S.M.; supervision, R.R. and D.K; funding acquisition, S.S.A. All authors have read and agreed to the published version of the manuscript.

Funding: This study was funded by the Deanship of Scientific Research, Taif University Researchers Supporting Project number (TURSP-2020/215), Taif University, Taif, Saudi Arabia.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data in this research paper will be shared upon request to the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Balamurugan, R. Application of Shuffled Frog Leaping Algorithm for Economic Dispatch with Multiple Fuel Options. In Proceedings of the 2012 International Conference on Emerging Trends in Electrical Engineering and Energy Management (ICETEEEM), Chennai, India, 13–15 December 2012; pp. 191–197. [CrossRef]
2. Aravindhababu, P.; Nayar, K.R. Economic dispatch based on optimal lambda using radial basis function network. *Int. J. Electr. Power Energy Syst.* **2002**, *24*, 551–556. [CrossRef]
3. Wood, A.J.; Wollenberg, B.F.; Sheblé, G.B. *Power Generation, Operation, and Control*; John Wiley & Sons: Hoboken, NJ, USA, 2013.
4. Liang, Z.X.; Glover, J.D. A zoom feature for a dynamic programming solution to economic dispatch including transmission losses. *IEEE Trans. Power Syst.* **1992**, *7*, 544–550. [CrossRef]
5. Victoire, T.A.A.; Jeyakumar, A.E. Hybrid PSO-SQP for economic dispatch with valve-point effect. *Electr. Power Syst. Res.* **2004**, *71*, 51–59. [CrossRef]
6. Chiang, C.L. Genetic-based algorithm for power economic load dispatch. *IEE Proc. Gener. Trans. Distrib.* **2007**, *1*, 261–269. [CrossRef]
7. Sinha, N.; Chakrabarti, R.; Chattopadhyay, P.K. Evolutionary programming techniques for economic load dispatch. *IEEE Evol. Comput.* **2003**, *7*, 83–94. [CrossRef]
8. Tan, Y.; Li, C.; Cao, Y.; Lee, K.Y.; Li, L.; Tang, S.; Zhou, L. Improved group search optimization method for optimal power flow problem considering valve-point loading effects. *Neurocomputing* **2015**, *148*, 229–239. [CrossRef]
9. Aydin, D.; Ozyon, S. Solution to non-convex economic dispatch problem with valve point effects by incremental artificial bee colony with local search. *Appl. Soft Comput.* **2013**, *13*, 2456–2466. [CrossRef]
10. Jayabarathi, T.; Raghunathan, T.; Adarsh, B.R. Ponnuthurai Nagaratnam Suganthan. Economic dispatch using hybrid grey wolf optimizer. *Energy* **2016**, *111*, 630–641. [CrossRef]
11. Chaturvedi, K.T.; Pandit, M.; Srivastava, L. Self-organizing hierarchical particle swarm optimization for nonconvex economic dispatch. *IEEE Trans. Power Syst.* **2008**, *23*, 1079–1087. [CrossRef]
12. Alsumait, J.S.; Sykulski, J.K.; Al-Othman, A.K. A hybrid GA-PS-SQP method to solve power system valve-point economic dispatch problems. *Appl. Energy* **2010**, *87*, 1773–1781. [CrossRef]
13. Roy, P.; Roy, P.; Chakrabarti, A. Modified shuffled frog leaping algorithm with genetic algorithm crossover for solving economic load dispatch problem with valve-point effect. *Appl. Soft Comput.* **2013**, *13*, 4244–4252. [CrossRef]
14. Yang, X.; Hosseini, S.S.S.; Gandomi, A.H. Firefly Algorithm for solving non-convex economic dispatch problems with valve loading effect. *Appl. Soft Comput.* **2012**, *12*, 1180–1186. [CrossRef]
15. Wang, Y.; Zhou, J.; Lu, Y.; Qin, H.; Wang, Y. Chaotic self-adaptive particle swarm optimization algorithm for dynamic economic dispatch problem with valvepoint effects. *Expert Syst. Appl.* **2011**, *38*, 14231–14237.
16. Faisal, A.N.; Cao, M.; Shen, L.; Fu, R.; Šumarac, D. The combined social engineering particle swarm optimization for real-world engineering problems: A case study of model-based structural health monitoring. *Appl. Soft Comput.* **2022**, *123*, 108919.
17. Zamani, H.; Nadimi-Shahraki, M.H.; Gandomi, A.H. Starling murmuration optimizer: A novel bio-inspired algorithm for global and engineering optimization. *Comput. Methods Appl. Mech. Eng.* **2022**, *392*, 114616. [CrossRef]
18. Nadimi-Shahraki, M.; Ali Fatahi, H.Z.; Abualigah, L. An improved moth-flame optimization algorithm with adaptation mechanism to solve numerical and mechanical engineering problems. *Entropy* **2021**, *23*, 1637. [CrossRef]
19. Nadimi-Shahraki, M.; Zamani, H. DMDE: Diversity-maintained multi-trial vector differential evolution algorithm for non-decomposition large-scale global optimization. *Expert Syst. Appl.* **2022**, *198*, 116895. [CrossRef]
20. Balamurugan, R.; Subramanian, S. Hybrid integer coded differential evolution dynamic programming approach for economic load dispatch with multiple fuel options. *Energy Convers. Manag.* **2008**, *49*, 608–614. [CrossRef]
21. Cai, J.; Mab, X.; Li, Q.; Li, L.; Peng, H. A multi-objective chaotic ant swarm optimization for environmental/economic dispatch. *Int. J. Electr. Power Energy Syst.* **2010**, *32*, 337–344. [CrossRef]
22. Ghoshal, S.P.; Chatterjee, A.; Mukherjee, V. Bio-inspired fuzzy logic based tuning of power system stabilizer. *Expert Syst. Appl.* **2009**, *36*, 9281–9292. [CrossRef]
23. Pothiya, S.; Ngamroo, I.; Kongprawechnon, W. Ant colony optimisation for economic dispatch problem with non-smooth cost functions. *Int. J. Electr. Power Energy Syst.* **2010**, *32*, 478–487. [CrossRef]
24. Roy, P.K.; Ghoshal, S.P.; Thakur, S.S. Biogeography-based optimization for economic load dispatch problems. *Elect. Power Compon. Syst.* **2010**, *38*, 166–181. [CrossRef]
25. Mandal, B.; Roy, P.K.; Mandal, S. Economic load dispatch using krill herd algorithm. *Int. J. Electr. Power Energy Syst.* **2014**, *57*, 1–10. [CrossRef]
26. Lee, K.Y.; Sode-Yome, A.; Park, J.H. Adaptive Hopfield neural networks for economic load dispatch. *IEEE Trans. Power Syst.* **1998**, *13*, 519–525. [CrossRef]
27. Vo, D.N.; Ongsakul, W. Economic dispatch with multiple fuel types by enhanced augmented Lagrange Hopfield network. *Appl. Energy* **2012**, *91*, 281–289. [CrossRef]
28. Vo, N.D.; Ongsakul, W.; Polprasert, J. The augmented Lagrange Hopfield network for economic dispatch with multiple fuel options. *Math. Comput. Model.* **2013**, *57*, 30–39.
29. Barisal, A.K. Dynamic search space squeezing strategy based intelligent algorithm solutions to economic dispatch with multiple fuels. *Int. J. Electr. Power Energy Syst.* **2013**, *45*, 50–59. [CrossRef]

30. Meng, A.; Li, J.; Yin, H. An efficient crisscross optimization solution to large-scale non-convex economic load dispatch with multiple fuel types and valve-point effects. *Energy* **2016**, *113*, 1147–1161. [CrossRef]
31. Sayah, S.; Hamouda, A. A hybrid differential evolution algorithm based on particle swarm optimization for nonconvex economic dispatch problems. *Appl. Soft Comput.* **2013**, *13*, 1608–1619. [CrossRef]
32. Pradhan, M.; Roy, P.K.; Pal, T. Oppositional based grey wolf optimization algorithm for economic dispatch problem of power system. *Ain Shams Eng. J.* **2017**, *9*, 2015–2025. [CrossRef]
33. Wang, Y.; Li, B.; Weise, T. Estimation of distribution and differential evolution cooperation for large scale economic load dispatch optimization of power systems. *Inf. Sci.* **2010**, *180*, 2405–2420. [CrossRef]
34. Bhattacharjee, K.; Bhattacharya, A.; Dey, S.H.N. Chemical reaction optimization for different economic dispatch problems. *IET Gener. Transm. Dis.* **2014**, *8*, 530–541. [CrossRef]
35. Singh, N.J.; Dhillon, J.S.; Kothari, D.P. Synergic predator-prey optimization for economic thermal power dispatch problem. *Appl. Soft Comput.* **2016**, *43*, 298–311. [CrossRef]
36. Thang, T.N.; Dieu, N.V. The application of one rank cuckoo search algorithm for solving economic load dispatch problems. *Appl. Soft Comput.* **2015**, *37*, 763–773.
37. Amjady, N.; Nasiri-Rad, H. Nonconvex economic dispatch with AC constraints by a new real coded genetic algorithm. *IEEE Trans. Power Syst.* **2009**, *24*, 1489–1502. [CrossRef]
38. Chiang, C.-L. Improved Genetic Algorithm for Power Economic Dispatch of Units with Valve-Point Effects and Multiple Fuels. *IEEE Trans. Power Syst.* **2005**, *20*, 4. [CrossRef]
39. Duan, H.; Qiao, P. Pigeon-inspired optimization: A new swarm intelligence optimizer for air robot path planning. *Int. J. Intell. Comput. Cybern.* **2014**, *7*, 24–37. [CrossRef]
40. Tizhoosh, H.R. Opposition-Based Learning: A New Scheme for Machine Intelligence. In Proceedings of the International Conference on Computational Intelligence for Modeling, Control and Automation, Vienna, Austria, 28–30 November 2005; pp. 695–701.
41. Roy, P.K.; Paul, C.; Sultana, S. Oppositional teaching learning-based optimization approach for combined heat and power dispatch. *Int. J. Elect. Power Energy Syst.* **2014**, *57*, 392–403. [CrossRef]
42. Alkayem, N.F.; Shen, L.; Asteris, P.G.; Sokol, M.; Xin, Z.; Cao, M. A new self-adaptive quasi-oppositional stochastic fractal search for the inverse problem of structural damage assessment. *Alex. Eng. J.* **2022**, *61*, 1922–1936. [CrossRef]
43. Coelho, L.D.S.; Mariani, V.C. Combining of chaotic differential evolution and quadratic programming for economic dispatch optimization with valve point effect. *IEEE Trans. Power Syst.* **2006**, *21*, 989–996.
44. Zou, D.; Li, S.; Wang, G.G.; Li, Z.; Ouyang, H. An improved differential evolution algorithm for the economic load dispatch problems with or without valve-point effects. *Appl. Energy* **2016**, *181*, 375–390. [CrossRef]
45. Srinivasa Reddy, A.; Vaisakh, K. Shuffled differential evolution for large scale economic dispatch. *Electr. Power Syst. Res.* **2013**, *96*, 237–245. [CrossRef]
46. Sahoo, S.; Mahesh Dash, K.; Prusty, R.C.; Barisal, A.K. Comparative analysis of optimal load dispatch through evolutionary algorithms. *Ain Shams Eng. J.* **2015**, *6*, 107–120. [CrossRef]
47. Mohammad, Z.H.; Nadimi-Shahraki, H.; Amir, H.G. QANA: Quantum-based avian navigation optimizer algorithm. *Eng. Appl. Artif. Intell.* **2021**, *104*, 104314.