Wiley
International Journal of Energy Research
Volume 2024, Article ID 8420107, 34 pages
https://doi.org/10.1155/2024/8420107

WILEY

*Research Article*

# A Novel Hybrid Algorithm for Solving Economic Load Dispatch in Power Systems

**Khairul Eahsun Fahim**,[1] **Liyanage C. De Silva**,[1] **Viknesh Andiappan**,[2] **Sk. A. Shezan**,[3] **and Hayati Yassin**[1]

[1]*Faculty of Integrated Technologies, Universiti Brunei Darussalam, Gadong, Bandar Seri Begawan, BE1410, Brunei Darussalam*
[2]*Faculty of Engineering, Computing and Science, Swinburne University of Technology, Jalan Simpang Tiga, Kuching 93350, Sarawak, Malaysia*
[3]*Discipline of Engineering and Energy, Murdoch University, Perth, Australia*

Correspondence should be addressed to Viknesh Andiappan; vmurugappan@swinburne.edu.my and Hayati Yassin; hayati.yassin@ubd.edu.bn

Received 11 March 2024; Revised 9 September 2024; Accepted 10 October 2024

Academic Editor: Jing Shi



Various algorithms have been created in the past to take economic load dispatch (ELD) into account. These algorithms, however, concentrate on multiple tuning parameters, necessitating hyperparameter adjustment. A unique parameterless hybrid is presented to explicitly evaluate ELD for test systems and real-world power plant systems matching the operational limitations. In addition, earlier algorithms could only offer estimates of the final cost of fuel based on the hyperparameter choices. This may prevent the global minimum values from being met. To find comprehensive solutions to the ELD problem in power systems, this paper suggests a new method called the hybrid Jaya optimization algorithm, which uses the merits of the Jaya and teaching–learning-based optimization (TLBO) algorithms. This enhancement is proposed to improve the population variety, the balance between local and global search, and the early convergence of the original Jaya optimization method. A metaheuristic optimization technique called TLBO simulates the teaching–learning process in a classroom to optimize problems. The TLBO algorithm uses an exploration phase in which possible solutions are generated at random to discover the best solution. The algorithm then uses the exploitation phase to refine the search space-based parameter adjustments to enhance the quality of the best solution identified. On the other hand, the Jaya algorithm is a metaheuristic optimization algorithm motivated by the idea of social behavior in nature. Candidate solutions are improved repeatedly through cooperation and competition using a population-based approach, and each solution adjusts its position based on the best and worst answers in the population. By combining the advantages of both algorithms, hybrid Jaya (Jaya–TLBO) outperforms each method alone and minimizes the cost of power generation, improving convergence solution quality. To test its efficacy, the hybrid Jaya–TLBO algorithm is tested on four different test cases, such as an Institute of Electrical and Electronics Engineers (IEEE) 6-unit, 13-unit, 20-unit, 40-unit ELD system and an Indonesian 10-unit one. Simulation results show that the proposed algorithm is superior in cost minimization to other well-known algorithms that have been used recently. As a result, power system planners can utilize this technique to find the most economical load dispatch.

**Keywords:** artificial intelligence algorithm; economic load dispatch (ELD); evolutionary algorithm; hybrid optimization algorithm; Jaya–TLBO algorithm; power system optimization

## 1. Introduction

The optimal operation of power units in power systems using various fuel sources such as natural gas, oil, and coal necessitates a reduction in overall fuel cost while considering network constraints. These units' total fuel cost function is separated into various piecewise quadratic functions. The economic load dispatch (ELD) method, which takes into account unit equality and inequality constraints, is a critical optimization problem in the power system [1]. However, characteristics such as valve point loading effect (VPLE) from multivalve steam turbines, power losses, ramp rate limits, and prohibited operation zones

(POZs) make the ELD of electrical systems complicated to optimize. This is because these characteristics create a nonconvex, nonlinear, and nonsmooth ELD optimization problem, thus increasing the problem's complexity [2].

Examples of classical mathematical programing algorithms such as contriction-factor-particle swarm optimization (CF-PSO), quantum-behaved PSO (QPSO), multiobjective PSO (MOPSO), dynamic PSO (DPSO), hybrid PSO, cooperative PSO (CPSO), theta-PSO ($\theta$-PSO), and modified time varying PSO (MTVPSO) used to address the numerous ELD issues include the evolutionary programing (EP) approach [3], the Hopfield artificial neural network method [4], and the adaptive Hopfield neural network. With the help of these methods, solutions that are still far from being dependable, quick, and ideal were obtained. However, these methods are not always effective as they cannot guarantee the best outcome for several nonconvex and nonsmooth objective functions in power systems [5].

Numerous optimization algorithms have been developed recently, including PSO [6] and its related algorithms such as QPSO, MTVPSO, and $\theta$-PSO [7], teaching–learning-based optimization (TLBO) [8], EP [9], the hybrid gray wolf optimizer (HGWO) [10], the ant lion optimizer (ALO) [11], and the cuckoo search algorithm (CSA) [12–14].

In several articles, the hybridization of two or more algorithms was discussed to improve the performance of metaheuristic techniques and find the optimum solution to the ELD problem. Examples include the hybrid ant colony-genetic algorithm merged with genetic algorithm (GA-API) technique [15], which combined the genetic algorithm and the ant colony algorithm; the hybrid artificial algae algorithm (HAAA) [16], which combined the simplex search method and the artificial algae method; and the improved augmented gray wolf optimizer (AGWO) algorithm [17] which combined the gray wolf optimization method and the random exploratory method.

The ELD problem can be solved using the fitness-distance balance (FDB) approach, which measures each candidate solution's distance from the optimal solution and assesses its fitness based on the overall generating cost [18]. The FDB score assists in choosing a range of candidates who balance exploitation and exploration by combining fitness and distance. The chosen candidates are subjected to symbiotic organism search (SOS) operators to produce novel solutions that guarantee optimality and diversity. In order to optimize power generation costs while satisfying demand and operational restrictions, this evaluation, selection, and generation procedure is repeated until convergence requirements are satisfied. To solve the ELD problem, the FDB-SOS algorithm increases search efficiency and robustness.

The survival of metaheuristic search algorithms (metaheuristic searches [MHSs]) is typically determined by fitness values, which do not accurately reflect natural adaptation and lead to premature convergence. This is addressed by the natural survivor method (NSM) [19], which integrates interindividual connections and environmental adaptability into the survivor selection process. NSM dynamically weights scores to adjust during the search process depending on an individual's fitness value, mating pool contribution, and population contribution. Utilizing three distinct MHS algorithms (stochastic fractal search [SFS], teaching learning based artificial bee colony algorithm (TLABC), and Lévy-flight success-history-based adaptive differential evolution with semi-parameter adaptive control of mutation and archive (LSHADE-SPACMA), NSM showed noteworthy enhancements in performance throughout 39 optimization tasks and 10 practical engineering quandaries. While NSM adds to the computational complexity, it offers a solid substitute to improve the efficiency of the MHS algorithm. It is believed to be suitable for solving some ELD problems.

To apply the fitness-distance-constraint (FDC) algorithm for solving ELD, computing the fitness and constraint violation values of an initial population of candidate solutions is required. Transformation of these violation values into fitness, distance to the feasible region, and diversity contribution (FDC) scores. The next step involves utilizing these FDC scores to help choose guide solutions that promote viable and varied candidates while striking a balance between constraint satisfaction and fitness. Finally, this FDC-based guide selection in an evolutionary algorithm (e.g., Adaptive Guided Differential Evolution [AGDE]) is such that it can dynamically switch between traditional fitness-based guiding and FDC guiding in response to constraint violations. Finally, this procedure will be repeated iteratively until the optimization conditions are satisfied, improving the population's performance on confined problems through crossover and mutation [20]. This process can be used to solve ELD problems and check its suitability.

Using the dynamic fitness-distance balance (dFDB) selection approach, a powerful optimization tool can be made by including it in a MHS algorithm. Analyze performance utilizing nonparametric statistical techniques while conducting extensive experimental research to test the dFDB method and optimization algorithm on benchmark test suites [21]. However, the efficacy of this algorithm for solving ELD is yet to be confirmed as not much research has been done using this algorithm.

The optimal power flow (OPF) and ELD problem can be solved using the adaptive fitness-distance balance (AFDB)-based SFS algorithm in contemporary power networks incorporating renewable energy sources and load demand uncertainties. First, the guiding mechanism in MHS algorithms is improved by developing the AFDB guide selection method. The SFS algorithm is combined with this AFDB technique to produce the AFDB-SFS algorithm, which has enhanced exploration, exploitation, and balanced search capabilities. In-depth experimental research evaluates how well the AFDB-SFS method performs against alternative optimization algorithms across five test systems and several benchmark test suites [22]. Finding the best solution and accelerating convergence speed, the AFDB-SFS algorithm successfully optimizes the cost and coordination of power grids, according to the results.

Kahraman et al. [23] proposed a dynamic switched crowding (DSC) technique, which dynamically computes crowding distances utilizing combinations of decision, objective, and unified spaces, hence improving multiobjective evolutionary algorithms (MOEAs). The DSC technique addresses premature !convergence problems, which uses a switching mechanism to preserve diversity and balance between exploration and exploitation. Compared to competitors, the dynamic switched crowding-multiobjective adaptive guided differential evolution (DSC-MOAGDE) method performs more efficiently,

achieving up to 120.21% better cost optimization on multimodal multiobjective optimization problems (MMOPs) and real-world AC-OPF and alternating current/direct current-optimal power flow (AC/DC-OPF) problems. Among the study's main contributions are the novel DSC approach and its practical application to MOEAs, which yields significant gains in complex optimization problems and dynamic reference space strategies. The outcomes validate DSC's efficacy in offering optimal solutions and well-balanced search capabilities in both decision and objective domains. Though not much literature has been found on this specific literature that tries to solve ELD problem, researchers can give it a go and try to find out the suitability of this algorithm for solving ELD problems.

Ozkaya et al. [24] proposed a DSC-based multiobjective symbiotic organism search (MOSOS) algorithm in the year 2024 is an efficient way to handle the combined heat and power economic emission dispatch (CHPEED) problem, which is characterized by conflicting objectives and many restrictions. This approach avoids local solution traps by promoting exploration and preserving a healthy balance between exploitation and exploration, which can be beneficial in finding the optimal minimal solution for the ELD problem. Through experimental investigations on Institute of Electrical and Electronics Engineers (IEEE) Congress on evolutionary computation (CEC) 2020 MMOPs and CHPEED challenges, DSC-MOSOS achieved cost improvements of 0.2% to 16.55% and emission reductions of 0.2–42.97 kg, outperforming the base MOSOS algorithm and 14 competitors. Furthermore, DSC-MOSOS proved its high efficacy and performance with a 91.16% success rate in stability analysis.

*1.1. Single Objective Self-Organizing Tree Algorithm (SOTA).* There are different variants of the ELD problem. Due to its noncontinuous and nonconvex constraints, the combined heat and power economic dispatch (CHPED) problem necessitates a strong optimization strategy. To improve exploration and balance exploitation while addressing the drawbacks of the original artificial rabbits optimization (ARO) algorithm, the AFDB-based ARO algorithm was created by Ozkaya et al. [25] using benchmark and CHPED problems, experimental research comparing AFDB-ARO with ARO and other algorithms demonstrated that AFDB-ARO performed better, achieving optimal solutions in the majority of situations with a mean success rate of 87.62%. For the first time, extensive test cases were used to evaluate the algorithm on systems with up to 192 units and restrictions such as valve point loading impact, transmission losses, and forbidden operation zones. Stability analysis validated that AFDB-ARO consistently and effectively finds workable solutions.

In Kahraman et al.'s [22] study, the dFDB-SFS algorithm, with its robust optimization capabilities inspired by natural growth phenomena, can be effectively applied to solve the ELD problem by optimizing power output to minimize generation costs while meeting demand and operational constraints [22]. Utilizing the dFDB method for balanced exploration and exploitation, the algorithm ensures accurate and diverse solutions, making it well-suited for ELD's complex and multimodal landscape. Its demonstrated accuracy and robustness in global optimization tasks, such as photovoltaic modeling, further underscore its potential effectiveness in achieving cost-effective and reliable power system operation [22].

Through adaptive parameter tuning and a fitness-distance balancing mechanism, the improved adaptive gaining-sharing knowledge algorithm developed by Kaharaman et al. with FDB-based guiding mechanism (FDBAGSK) offers a robust solution for the ELD problem by balancing exploration and exploitation. FDBAGSK efficiently develops optimal generation schedules by formulating the ELD issue with an objective function to minimize operational expenses and include restrictions like power balance and generating limits. Benchmark comparisons indicate its effectiveness and show that it can manage complex constraints and nonconvex objectives, which makes it a valuable tool for modern power systems optimization [26].

By scoring each candidate based on both fitness value and distance from the optimum solution, the FDB selection method improves MHS methods. Due to its dual consideration, it is beneficial in handling complicated problems such as ELD by preventing early convergence and maintaining diversity. FDB guarantees balanced exploration and exploitation in ELD applications, enhancing search efficiency and preventing local optima. Integrating FDB with algorithms such as SOS leads to more dependable and efficient optimal load distribution schemes. Overall, FDB is a strong option for optimizing ELD problems because of its capacity to strike a compromise between fitness and diversity.

The fitness-distance balance artificial ecosystem optimization (FDBAEO) algorithm can solve ELD problems by balancing exploration and exploitation through fitness and distance metrics. This involves evaluating candidates based on cost efficiency and diversity, then selecting and optimizing the best candidates iteratively to minimize fuel costs while meeting system constraints. FDBAEO's robust search performance and diversity maintenance ensure efficient convergence to the optimal load distribution [27].

The fitness distance balance-based chimp optimization algorithm (FDBChOA) enhances the original ChOA by addressing its premature convergence and poor diversity issues, thus improving its global optimization capabilities. FDBChOA has been applied to optimize power system stabilizer parameters, yielding superior results to the original ChOA in unconstrained benchmarks and constrained engineering design problems [28]. This demonstrates FDBChOA's effectiveness in solving real-world engineering and global optimization problems.

The original ChOA's premature convergence and poor diversity flaws are addressed by the FDBChOA, which strengthens the algorithm's capacity for worldwide optimization. Applying FDBChOA to power system stabilizer parameter optimization has produced better results than using the original ChOA in limited engineering design issues and unconstrained benchmarks. This shows that a variety of real-world engineering and global optimization problems, including the ELD problem, can be successfully solved by FDBChOA [28].

The OPF problem in hybrid AC/DC power grids is better searched for using the Improved phasor particle swarm optimization (PPSO) with FDB. The algorithm improves exploration and balanced search capabilities by revamping PPSO's search operators and using FDB to pick solution possibilities.

Empirical findings reveal that FDBPPSO proficiently addresses the intricate OPF problem and additional benchmark optimization assignments, offering resilient and effective solutions for hybrid AC/DC power systems [29]. When solving ELD problems, the FDBPPSO technique can be used to optimize load distribution across power-generating units to reduce costs while maintaining system restrictions.

Apart from the algorithms listed above, some other algorithms that can be used to solve ELD problems are FDB-CHOA [28],dFDB-GBO [30], and FDB-AOA [31]. Though these algorithms have the potential to solve ELD problems, customizing them is a time-consuming task. Hence, comparative studies with these algorithms are intended for future studies.

Rao [32] first presented the Jaya algorithm, a population-based optimization technique, in 2016. In several disciplines, including engineering, finance, and medicine, complicated issues are optimized using this straightforward yet effective technique. In each iteration, the Jaya algorithm modifies the population of solutions utilizing two main rules, attraction and repulsion. Repulsion keeps the population from local minima and guarantees that the best solutions are drawn toward the ideal solution, preventing stagnation [33]. The algorithm calculates the new solution using the attraction and repulsion principles in each iteration, updating the population while assessing the fitness of each solution. A termination criterion must be satisfied for the process to end.

However, Xue and Wu proposed another population-based algorithm called the TLBO [34] algorithm before proposing the Jaya algorithm in 2011. It is a population-based optimization technique. It is modeled after the teaching–learning process in the classroom, where students and teachers work together to increase comprehension of a subject. The teachers and students comprise the two groups the TLBO algorithm divides into. Students benefit from teachers' information sharing by learning from it and expanding their knowledge. The algorithm then updates the population by the top answers discovered by teachers and pupils, causing the process to converge to the global optimum. In the context of ELD, the TLBO algorithm is applied in the context of ELD by simulating a teaching and learning process among a population of solutions to iteratively optimize the allocation of power generation among different units, aiming to achieve the most cost-effective distribution while satisfying load demand and operational constraints.

The ELD problem, which is a critical task in power system operation to distribute the load among multiple generators optimally, is addressed by the suggested hybrid Jaya–TLBO algorithm in this paper, which is referred to as the hybrid Jaya algorithm in the rest of the paper. To increase the convergence rate and correctness of the solution, the algorithm combines the exploitation ability of the TLBO algorithm with the exploration ability of the Jaya algorithm. The Jaya algorithm is used to initialize the population in the hybrid algorithm, and the TLBO method is then used to enhance the quality of the result. In terms of lowering fuel costs and minimizing power loss, the suggested algorithm outperforms other existing algorithms regarding accuracy and dependability. A viable method for addressing the ELD problem in the electricity sector is the hybrid Jaya–TLBO algorithm.

The hybrid Jaya approach has been used to address the ELD problem. This study investigates a quadratic cost function with generator limitations, VPLE, power loss, ramp rate limits, and POZ to determine the proposed hybrid Jaya algorithm's effectiveness in solving the ELD problem. On test systems with six generators, 13 generators, 20 generators, 40 generators, and Indonesian 10-unit generators, the suggested hybrid Jaya algorithm is tested. The outcomes of the hybrid Jaya are contrasted with those of newly proposed supply–demand-based optimization (SDO) [35], the eagle-strategy supply–demand–based optimization algorithm [35], the hunger games search (HGS) [36], and other widely used techniques as well as classical algorithms such as GA, PSO, and so on.

The following is a summary of the main contributions of this work:

- The novel hybrid Jaya–TLBO optimization algorithm is evaluated using IEEE test system data and has been proven efficient in minimizing generation fuel costs.
- The hybrid Jaya algorithm is suggested to solve the ELD with VPLE, power loss, and POZ.
- The simulation results demonstrate the supreme efficacy of the suggested hybrid technique and the solution strategy for handling the ELD problem.
- In terms of strength, solution precision, and efficacy, the proposed hybrid Jaya is compared to several recommended algorithms in the literature.

Summary of the literature and the proposed solution are illustrated in Table 1.

The rest of the article is organized as follows: Section 2 describes the mathematical model of the ELD problem. Section 3 describes the proposed hybrid Jaya algorithm for solving the ELD problem. Section 4 presents the simulation findings from many case studies. Section 5 describes the sensitivity analysis, and finally, Section 6 concludes the paper.

## 2. Mathematical Model of the ELD

The ELD problem is an optimization problem that aims to reduce fuel costs while taking into account inequality restrictions such as generating capacity limit, ramp rate limit, VPLE, and prohibited operating zones.

The objective function for the ELD problem is determined as follows: To generate the most power, the fuel cost function is optimized [33]. The following functions are used to represent overall fuel costs:

$$F_T = \sum_{i=1}^{n} F(P_i) = \sum_{i=1}^{n} a_i P_i^2 + b_i P_i + c_i. \tag{1}$$

$F_T$ is regarded as the overall cost of producing electricity in dollars per hour. The fuel cost coefficients are $a_i$, $b_i$, and $c_i$, and $P_{Gi}$ is the output unit $i$ of real power generation.

The equation above can be restated as follows when the generator's valve point impact is considered:

Table 1: Summary of literature review [36].

| Research gap | Proposed solution |
|---|---|
| Optimization challenges exist in the ELD problem in power systems due to valve point loading, power loss, ramp rate limits, and POZs | Development of a novel hybrid Jaya–TLBO optimization algorithm to address ELD with improved efficiency |
| Limited effectiveness of existing optimization algorithms (e.g., GA, PSO) in solving nonconvex and nonsmooth ELD problems in power systems | Integration of Jaya and TLBO algorithms to enhance convergence rate and solution accuracy in ELD problem-solving |
| Lack of a comprehensive approach to minimize generation fuel costs while considering complex generator constraints | Proposal of a hybrid Jaya algorithm that addresses ELD problems with generator limitations, valve point loading effect, power loss, ramp rate limits, and POZ |
| The proposed hybrid Jaya algorithm needs empirical testing and comparison against existing techniques on various test systems, including IEEE cases and Indonesian power units | Evaluation of the hybrid Jaya algorithm's effectiveness in minimizing fuel costs, its solution strategy, and comparison with other recommended algorithms |
| The importance of demonstrating the algorithm's strength, solution precision, and efficacy through simulation results and sensitivity analysis | Presentation of simulation results from multiple case studies, sensitivity analysis, and a conclusion summarizing the findings |

Abbreviations: ELD, economic load dispatch; GA, genetic algorithm; IEEE, Institute of Electrical and Electronics Engineers; POZs, prohibited operation zones; PSO, particle swarm optimization; TLBO, teaching–learning-based optimization.

$$F_T = \sum_{i=1}^{n} F(P_i) = \sum_{i=1}^{n} a_i P_i^2 + b_i P_i + c_i + |e_i \sin(f_i(P_{Gi\min} - P_{Gi}))|. \tag{2}$$

The cost function $F_T$ is a quadratic function of the cost coefficients $a_i, b_i, c_i$, $e_i$, and $f_i$ concerning the output power and $P_i^{\min}$ is the minimum active power generation from the $i$th generating unit.

### 2.1. Constraints

*2.1.1. Power Balance Constraints.* Total power generated must equal the sum of total power demand $P_D$ and total transmission losses $P_L$, which is expressed as follows [37]:

$$\sum_{i=1}^{n} P_i - P_D - P_L = 0, \tag{3}$$

where $P_L$ is calculated using the Kroon's loss formula for power dispatch [38]:

$$P_L = \sum_{i=1}^{n}\sum_{j=1}^{n} P_i B_{ij} P_j + \sum_{i=1}^{n} B_{i0} P_i + B_{00}. \tag{4}$$

Loss coefficient matrices are denoted with $B_{ij}$, $B_{i0}, B_{00}$ in normal operating conditions.

*2.1.2. Power Generation Constraints.* The minimum and maximum power generation limits are denoted by the following equation [39]:

$$P_i^{\min} \le P_i \le P_i^{\max}, \tag{5}$$

where $P_i^{\min}$ is the minimum limit of the generator, and $P_i^{\max}$is the maximum.

*2.1.3. Ramping Rate Constraints.* Practical limitations prevent instantaneous changes to output power $P_i$ since each generator's working limit is constrained by its ramp rate limit. The following are the up- and down-ramp limits [40]:

$$\begin{aligned} P_i - P_i^0 &\le \mathrm{UR}_i, \\ P_i^0 - P_i &\le \mathrm{DR}_i, \end{aligned} \tag{6}$$

where current output power is denoted with $P_i$, $P_i^0$ is the previous output power, $\mathrm{UR}_i$ is the up-ramping limit of the $i$th generator, and $\mathrm{DR}_i$ is the down-ramping limit.

*2.1.4. Prohibited Operating Zones.* Generators cannot operate in the prohibited operating zone [41]:

$$P_i \varepsilon \begin{cases} P_{i,\min} \le P_i \le P_{i,1}^L \\ P_{i,k-1}^U \le P_i \le P_{i,k}^L k = 2, 3, \ldots, pz_i \\ P_{i,pz_i}^U \le P_i \le P_{i,\max} \end{cases}. \tag{7}$$

Since each generator's operating limit is bound by its ramp rate limit, practical restrictions prevent instantaneous changes to the output power $P_i$.

## 3. Methodology

*3.1. Jaya Algorithm.* The Jaya method is a population-based optimization technique for problem-solving. Rao [8] proposed it in 2016. The method works by keeping a population of possible solutions and improving them repeatedly through a series of updates. In each iteration, the method takes two randomly picked solutions from the population and seeks to develop a new solution that is better than both of them. This is accomplished by modifying each variable of the new solution toward the better solution, with the amount of adjustment equal to the difference between the two solutions.

Let $f(x)$ represents the reduced (or maximized) objective function. Assume that at any iteration $i$, there is $m$ number of design variables i.e., $j = 1, 2, \ldots., m$ and $n$ number of potential solutions *population size*, $k = 1, 2, \ldots., n$. Let the best candidate receive the best value of $f(x)$ in the entire set of candidates, that is, $C$ best, and the worst candidate obtain the worst value of $f(x)$ in all candidate solutions, that is, $f(x)$ worst. If $X_{j,k,i}$ is the value of the $j$th variable for the $k$th

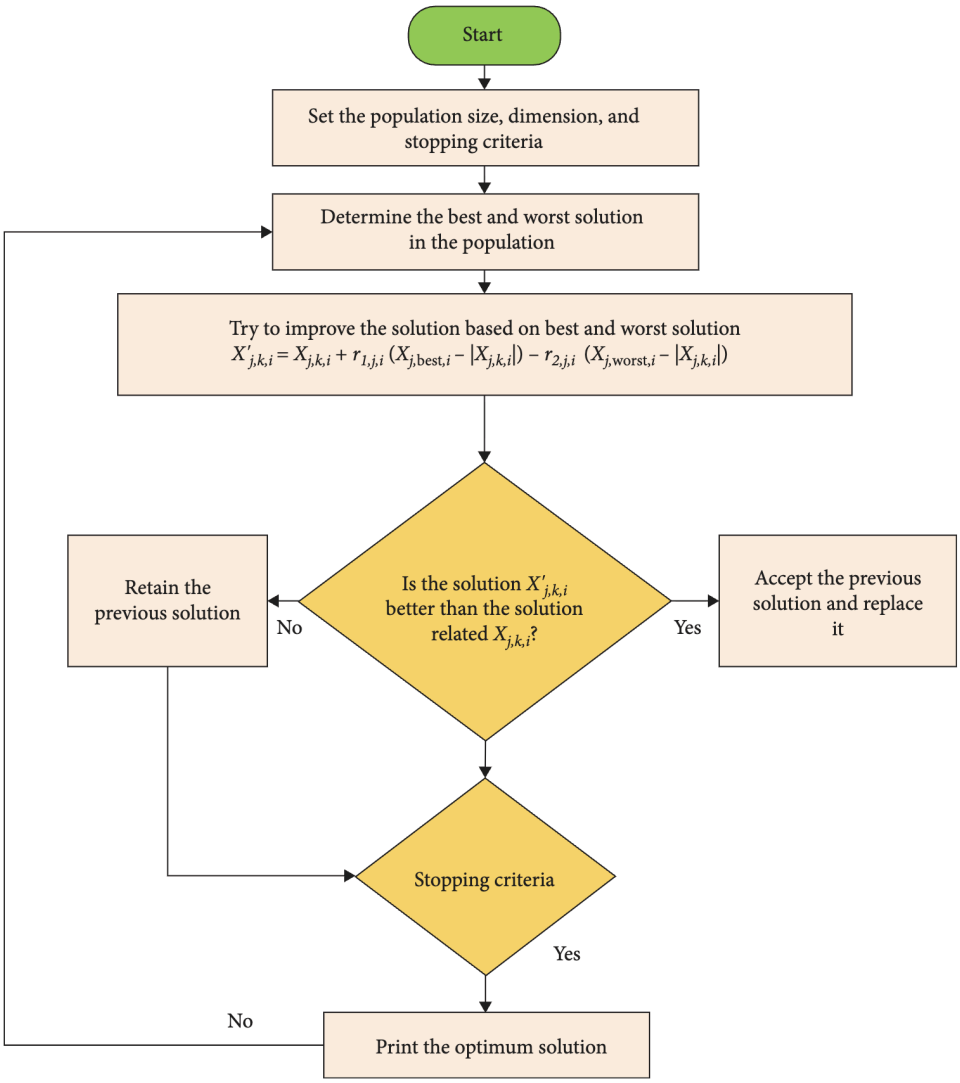


FIGURE 1: Jaya algorithm.

candidate during the $i$th iteration, then the following equation updates this value [33]:

$$X'_{j,k,i} = X_{j,k,i} + r_{1,j,i}\left(X_{j,\text{best},i} - \left|X_{j,k,i}\right|\right) - r_{2,j,i}\left(X_{j,\text{worst},i} - \left|X_{j,k,i}\right|\right), \tag{8}$$

where $X_{j,\text{best},i}$ denotes the $j$ value for the best candidate. On the other hand, the worst candidate is denoted by $X_{j,\text{worst},i}$. During $i$th iteration, the two random numbers $r_{1,j,i}$ and $r_{2,j,i}$ range from 0 to 1. The terms $r_{1,j,i}(X_{j,\text{best},i} - |X_{j,k,i}|)$ can be used to calculate how near the solution is to the best answer. At the same time, the solution attempts to avoid the worst solution. "$-r_{2,j,i}(X_{j,\text{worst},i} - |X_{j,k,i}|$" shows how far the solution tries to avoid the worst solution. The term $X'_{j,k,i}$ is supposed to produce improved function value each time. It is approved for calculation if it delivers a superior function value. All acceptable function values are saved until the conclusion of each iteration and carried forward to the next. Figure 1 shows the flowchart of Jaya algorithm.

*3.2. TLBO Algorithm.* TLBO is a population-based optimization technique that has been shown to be a successful optimization algorithm like other nature-inspired algorithms. The algorithm mimics what is being taught and learned in a classroom. It was proposed by Venkata Rao in 2011 [42]. The algorithm is quite simple to use. There are two phases to the TLBO process. The first section is called the "teacher phase," and the second is the "learner phase." In the "teacher phase," the teacher instructs the class while the students learn from him or her; in the "learner phase," the students engage with one another in the classroom to learn. In TLBO, the population is viewed as a class of students. Every learner has the potential to be the answer to the optimization problem.

*3.2.0.1. Teacher Phase.* In this phase, the learners enhance their understanding of the teacher and class means. Let the teacher $T$ be the population's optimal solution in terms of fitness, $M$ be the mean, and $T_F$ be the teaching factor. For the $i$th learner $X_i$, *new* $X_i$ is the updated state of the learner $X_i$. The process of updating techniques is described in the

following equation [42]:

$$new\,X_i = X_i + r_i \times (T - T_F \times M), \tag{9}$$

where $r_i$ denotes a random number, and $T_F$ determines the amount by which the mean will be altered as a teaching element. Each component of the random vector rand, which has the range [0, 1], is a random number. $T_F$ is determined by Equation (10):

$$T_F = \text{round}[1 + \text{rand}(0, 1)]. \tag{10}$$

Following the teaching phase, the better learner among the learner and the newly created learner will be accepted and begin the learning phase.

*3.2.0.2. Learner Phase.* Students may actively participate in this phase through activities, including presentations, formal communications, and group debates, to broaden their knowledge. If the other student is more skilled than them, the learner picks up new information. The updated state of the learner $X_i$ is represented by $new\,X_i$ for the $i$th learner, which is determined by Equation (11) [42]:

$$new\,X_i = \begin{cases} X_i + r_i \times \left(X_i - X_j\right), \text{if } f(X_i) < f\left(X_j\right) \\ X_i + r_i \times \left(X_i - X_j\right), \text{otherwise} \end{cases}, \tag{11}$$

where $X_i$ and $X_j$ are arbitrarily chosen from the class, and their respective fitness scores are $f(X_i)$ and $f(X_j)$. $r_i$ is a random vector with a range of [0, 1].

Like the teaching phase, the improved learner among the learner and the newly created learner will be approved and will proceed toward the next teaching phase after the learning phase. Figure 2 illustrates the detailed flowchart of the TLBO algorithm.

*3.3. Hybrid Jaya Algorithm.* The Jaya–TLBO meta-heuristic method addresses the ELD problem. It combines two well-known meta-heuristic algorithms: TLBO and Jaya. TLBO is used for improved exploration, while the Jaya algorithm is used to use the search space better.

Appendix G provides the pseudocode and computational stages for the hybrid Jaya algorithm. As the ELD technique is used to estimate the generator's highest output at the lowest cost to satisfy the total demand while considering the system limitation, the Jaya–TLBO algorithm is an excellent option for solving the ELD problem precisely. Both the algorithms are run inside the loop simultaneously and it stores the greedy selection to find out the best results found from the cost function. The stored value is then compared with the previously existing value to find out the best solution. The flowchart of this hybrid algorithm is shown in Figure 3. This study demonstrates how the hybrid Jaya technique can be used to manage the ELD problem. The nonconvex and nonlinear features of the ELD issue can be resolved using this method, and it quickly finds the optimal solution. The hybrid Jaya technique, like other meta-heuristic algorithms like PSO, GA, BAT algorithm (BA), and differential evolution (DE), is effective for solving the ELD problem. Figure 3 illustrates the flowchart of the hybrid Jaya algorithm.

By balancing exploration and exploitation, the hybrid algorithm that combines the TLBO and Jaya algorithms performs exceptionally well regarding fairness for the ELD issue. The Jaya algorithm avoids local optima, but TLBO improves solutions through phases of teaching and learning. Premature convergence is less likely when a varied solution pool is maintained. It exhibits resilience in various situations, guaranteeing equitable outcomes irrespective of the magnitude or configuration of the issue. A major asset is the ability to effectively minimize expenditures, which regularly results in decreased expenses. The superiority of the algorithm is confirmed through benchmarking against other well-established techniques, including GA, PSO, and DE. Adjusting parameters dynamically enables flexibility in response to changing load demands and circumstances. The hybrid strategy fully addresses economic objectives while ensuring equitable and optimal load allocation. All in all, it guarantees a just assessment and optimization procedure.

## 4. Simulation and Optimization Results

*4.1. Case Studies.* This subsection conducts experiments on ELD problems involving 6-, 13-, 20-, 40-, and 10-unit real-world Indonesian power system, under various operational constraint functions, making it the best choice as it closely resembles real-world power systems encompassing small and medium-scale power plants. To examine the robustness, convergence behavior, and computational effectiveness of applying the suggested hybrid Jaya in solving the ELD issues, four test cases of ELD problems with varying sizes and limitations were considered. Table 2 gives an overview of these cases' characteristics.

Table 2 shows the characteristics of different test cases. The "✔" symbol signifies that the constraint is one of the constraints taken into account for solving the ELD problem, whereas the "—" symbol signifies that the constraint is ignored as the data are not available in the literature.

The suggested hybrid Jaya is implemented in the Windows 10 operating system's MATLAB programing language (R2021a). A computer with an Intel i7 central processing unit (CPU) clocked at 2.8 GHz and 16 GB of random access memory (RAM) was used for this study. All these parameters are provided in tabular format in Table 3. The findings are compared with those from several current algorithms and other published literature. The population sizes are 10 times the respective dimensions for cases 1, 2, 3, 4, and 5. Therefore, 60, 130, 200, 400, and 100 population sizes are used for the respective cases. The maximum iteration is set to 100 for test case 1 and 500 for the rest. The iteration numbers are kept the same or fewer when compared to the other algorithms being contrasted. This demonstrates that this hybrid algorithm reaches optimal results even with fewer iterations. This paper diligently contrasts our algorithm's performance with contemporary state-of-the-art solutions, as evidenced by the detailed comparisons presented in Tables 3–6, showcasing the relevance of our findings against recent advancements in the field. In all these case studies, NA stands for not applicable.

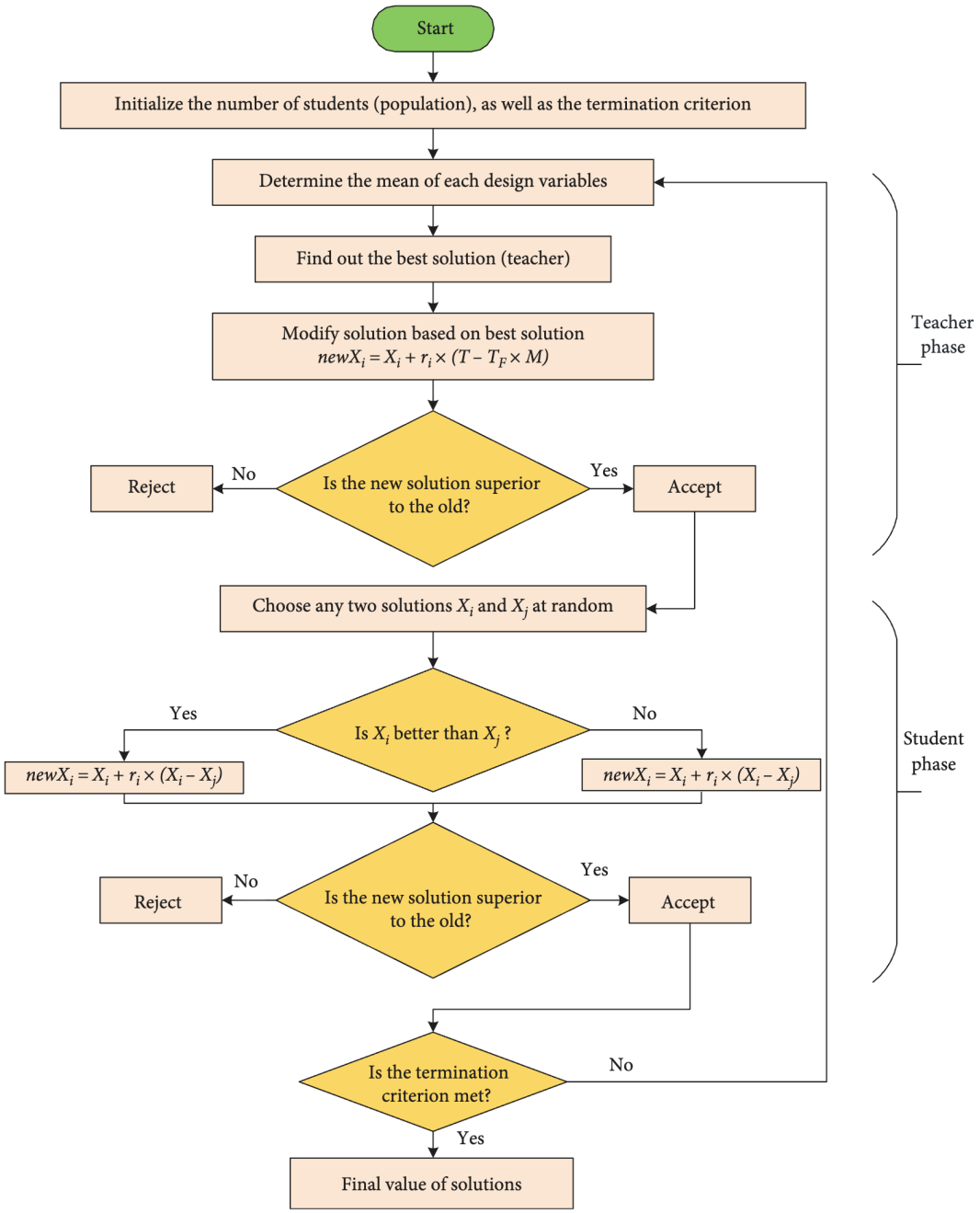


FIGURE 2: TLBO algorithm [8]. TLBO, teaching–learning-based optimization.

*4.1.1. Case 1: Six Thermal Units and 1263 MW Load Demand.* This scenario's system comprises six thermal units, 26 buses, and 46 transmission lines. The load demand is 1263 MW. In addition, ramp rate limits and power losses are taken into account, and all thermal units have POZs. The cost coefficients, loss coefficients B, and the upper and lower ramp limits, POZs, and initial power output of the units are given in Appendix A, where the maximum and minimum power limit cost coefficients, ramp rates, initial power output, and POZs are given in Appendix A, and the B loss coefficients are taken from Appendix B.

Table 2 shows the optimal scheduling results of the proposed hybrid Jaya compared to existing approaches. The results shown in Table 3 show that the proposed strategy gives better solution quality at a lower operating cost than existing techniques. The cost is 15,444.186310 $/h, less than the optimal cost reported in earlier publications while satisfying all restrictions such as ramp rate limits and POZs, load demand, and power generation boundaries. This improvement in performance can be attributed to the inherent capabilities of the hybrid Jaya algorithm, which effectively combines the strengths of both optimization techniques, allowing for a more comprehensive exploration of the solution space and thus leading to the identification of better solutions that fulfill the operational constraints and minimize the cost more efficiently than

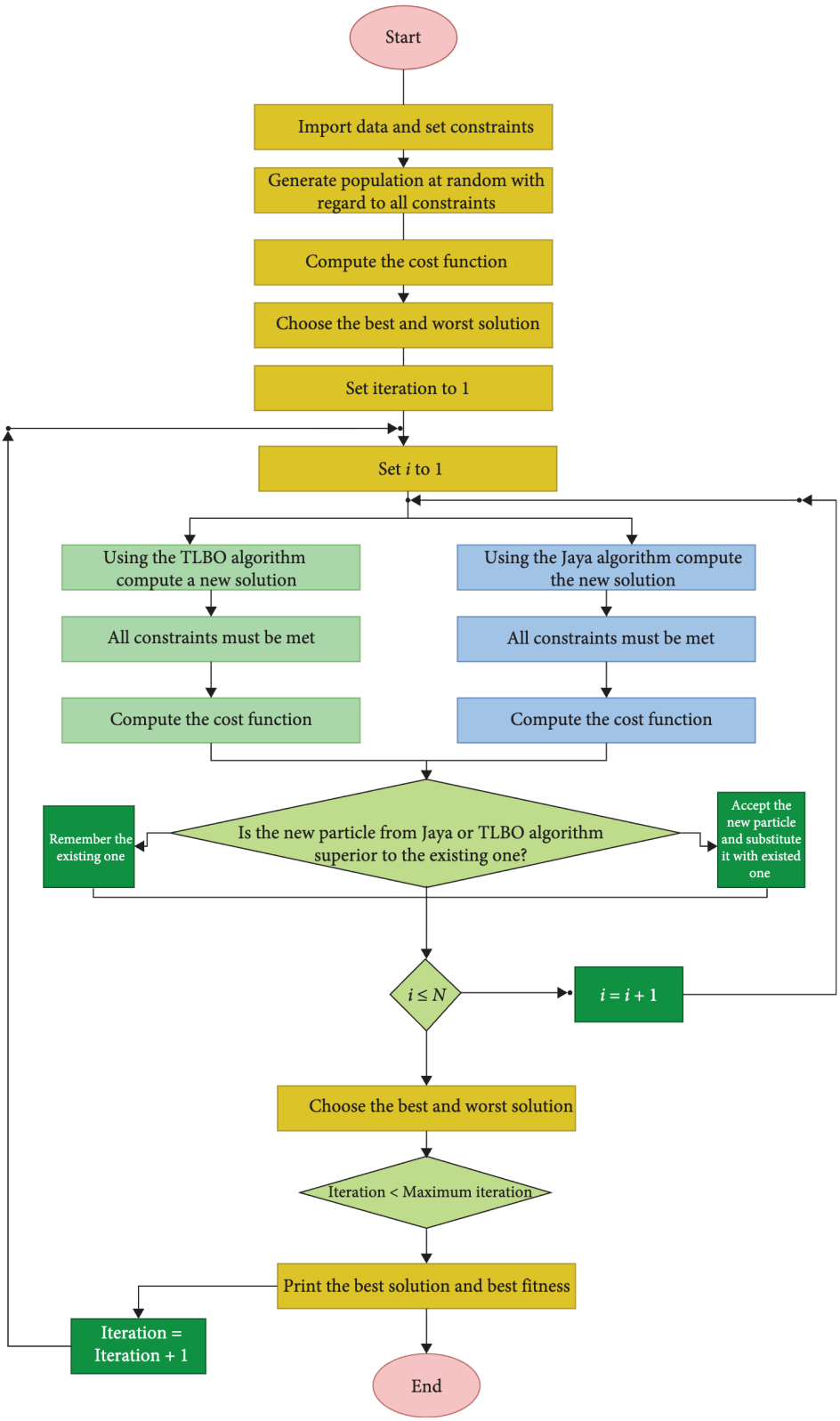


FIGURE 3: Hybrid Jaya algorithm. TLBO, teaching–learning-based optimization.

previous approaches. They can also clearly manage the generation and demand equality constraints. To explain this, consider the power inequality error as $error(V) = P_i - P_D - P_L$. The convergence characteristic curves of Jaya, TLBO, and hybrid Jaya in the ELD problem for optimal performance in the test system are shown in Figure 4, suggesting that the hybrid Jaya algorithm has greater convergence than the others. The optimal generation schedule from unit 1 to unit

Table 2: Characteristics of different test cases for ELD problem.

| Test case | Generating unit | Load demand (MW) | Transmission loss | Power balance | Generating capacity | Ramp rate limit | Prohibited operating zone |
|---|---|---|---|---|---|---|---|
| 1 | 6 | 1263 | ✓ | ✓ | ✓ | ✓ | ✓ |
| 2 | 13 | 1800 | — | ✓ | ✓ | ✓ | ✓ |
| 3 | 20 | 2500 | ✓ | ✓ | ✓ | — | — |
| 4 | 40 | 10,500 | — | ✓ | ✓ | ✓ | ✓ |

Abbreviation: ELD, economic load dispatch.

Table 3: The parameters of algorithm used.

| Parameter | Setting |
|---|---|
| Operating system | Windows 10 |
| MATLAB version | R2021a |
| CPU | Intel i7, 2.8 GHz |
| RAM | 16 GB |
| Population sizes | Case 1 : 60, Case 2 : 130, Case 3 : 200, Case 4 : 400, Case 5 : 100 |
| Maximum iterations | Case 1 : 100, Cases 2–5 : 500 |
| Iteration comparison | Iteration numbers are the same or fewer than other algorithms |

Abbreviations: CPU, central processing unit; RAM, random access memory.

Table 4: Fuel costs ($/h) for case 1 by various algorithms.

| Algorithm | Best | Worst | Average | STD |
|---|---|---|---|---|
| GA [43] | 15,459 | 15,524 | 15,469 | NA |
| PSO [43] | 15,450 | 15,492 | 15,454 | NA |
| MABC [44] | 15,449.9000 | 15,449.9000 | 15,449.9000 | $6.04 \times 10^{-8}$ |
| CBA [45] | 15,450.2400 | 15,518.6600 | 15,454.7600 | 2.9650 |
| NPSO-LRS [46] | 15,450 | 15,454 | 15,452 | NA |
| MSSA [47] | 15,449.9000 | 15,453.5500 | 15,449.9400 | 0.3647 |
| DE [48] | 15,449.7700 | 15,449.8700 | 15,449.7800 | NA |
| ST-IRDPSO [49] | 15,449.8900 | NA | 15,450.7000 | 1.4160 |
| SOH-PSO [50] | 15,446.0200 | 15,609.6400 | 15,497.3500 | NA |
| MCSA [51] | 15,449.1700 | 15,449.3900 | 15,449.2400 | 0.2681 |
| MTS [52] | 15,450.0600 | 15,453.6400 | 15,451.1700 | NA |
| HHS [53] | 15,449 | 15,453 | 15,450 | NA |
| SDO [35] | 15,444.1863 | 15,444.1863 | 15,444.1863 | 3.42E − 06 |
| ESSDO [35] | 15,444.1863 | 15,444.1863 | 15,444.1863 | 3.42E − 06 |
| ESCSDO10 [35] | 15,444.1863 | 15,444.1863 | 15,444.1863 | 4.03E − 06 |
| Rao-1 [54] | 15,445.8082 | 15,445.8082 | 15,445.8082 | 1.5099e-13 |
| Rao-2 [54] | 15,445.8082 | 15,445.8082 | 15,445.8082 | 1.4210e-14 |
| Rao-3 [54] | 15,445.8082 | 15,445.8082 | 15,445.8082 | 1.2434e-13 |
| TLBO | 15,293.0463 | 15,293.3239 | 15,293.0803 | 0.084799655 |
| Jaya | 15,293.0500 | 15,294.7100 | 15,293.2000 | 0.393441 |
| Hybrid Jaya | 15,293.0463 | 15,293.0463 | 15,293.0463 | 4.48128E-06 |

Abbreviations: CBA, classification based on association rules; DE, differential evolution; ESSDO, eagle-strategy supply–demand optimizer; GA, genetic algorithm; HHS, hybrid harmony search; MABC, multispecies artificial bee colony; MCSA, modified crow search algorithm; MSSA, multivariate singular spectrum algorithm; MTS, moving target search; NPSO-LRS, new PSO local random search; PSO, particle swarm optimization; SDO, supply–demand-based optimization; SOH-PSO, self-organizing hierarchical particle swarm optimization; STD, standard deviation; ST-IRDPSO, improved random drift particle swarm optimization algorithm; TLBO, teaching–learning-based optimization.

6 ($U_1 - U_6$) in MW of various algorithms for case study 1 is stated in Table 7.

The overall cost values of different optimization techniques are similar when dealing with tiny systems because of the relatively lesser complexity. The best-performing algorithm stands out brightly, but as the system size grows, performance disparities become more noticeable. The exponential increase in the complexity of the optimization problem, the size of the solution space, and the difficulties presented by local optima are the leading causes of this. An algorithm's scalability,

Table 5: Comparison of the simulation results for test case 2 (13-unit system).

| Algorithm | U1 | U2 | U3 | U4 | U5 | U6 | U7 | U8 | U9 | U10 | U11 | U12 | U13 | Cost($/h) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SDO [35] | 448.7990 | 152.9354 | 224.3995 | 159.7331 | 60.0001 | 159.7331 | 109.8665 | 109.8669 | 109.8666 | 40.0000 | 77.3999 | 55 | 92.3999 | 18,041.4900 |
| ESSDO [35] | 448.7990 | 152.9302 | 224.4018 | 109.8673 | 109.8665 | 109.8668 | 159.7331 | 109.8665 | 109.8688 | 40.0002 | 40.0002 | 92.3998 | 92.4000 | 18,028.5600 |
| ESCSDO10 [35] | 538.5587 | 75.6427 | 224.3995 | 109.8666 | 109.8666 | 109.8666 | 109.8666 | 109.8666 | 109.8666 | 40 | 77.3999 | 92.3999 | 92.3999 | 18,028.1900 |
| HGS [35] | 360.8537 | 360 | 224.4438 | 180 | 129.6487 | 60 | 112.2345 | 60 | 60 | 40 | 40 | 117.8193 | 55 | 18,552.1900 |
| NN-EPSO [55] | 490 | 189 | 214 | 160 | 90 | 120 | 103 | 88 | 104 | 13 | 58 | 66 | 55 | 18,442.5900 |
| PSO [46] | 538.5610 | 299.3550 | 75.0370 | 159.7340 | 60.0780 | 109.8640 | 109.9130 | 109.8700 | 60.0690 | 40.0350 | 77.5610 | 55.0420 | 55 | 18,014.1600 |
| $\theta$-PSO [56] | 628.3482 | 149.5978 | 222.7512 | 109.8665 | 109.8665 | 109.8665 | 109.8665 | 60 | 109.8665 | 40 | 40 | 55 | 55 | 17,963.8297 |
| MTVPSO [57] | 628.3185 | 149.5995 | 222.7491 | 109.8666 | 109.8666 | 109.8666 | 109.8666 | 60 | 109.8666 | 40 | 40 | 55 | 55 | 17,960.3659 |
| Jaya | 511.1597 | 256.2348 | 247.6432 | 100.7954 | 100.0629 | 101.2559 | 93.2996 | 98.8472 | 100.6672 | 40.0154 | 40.0173 | 55.0008 | 55.0007 | 17,932.7603 |
| TLBO | 512.0295 | 253.0990 | 248.0280 | 98.4799 | 101.5352 | 100.9320 | 99.0455 | 99.1925 | 97.6022 | 40.0061 | 40.0055 | 55.0224 | 55.02360 | 17,932.6903 |
| Hybrid Jaya | 500.8437 | 250.7712 | 259.3479 | 99.2483 | 100.2738 | 101.1728 | 99.0763 | 98.9119 | 100.3148 | 40.0174 | 40.0179 | 55.0028 | 55.0028 | 17,932.6558 |

Abbreviations: $\theta$-PSO, theta-PSO; ESSDO, eagle-strategy supply–demand optimizer; HGS, hunger games search; MTVPSO, modified time varying PSO; NN-EPSO, neural network enhanced particle swarm optimization; PSO, particle swarm optimization; SDO, supply–demand-based optimization; TLBO, teaching–learning-based optimization.

Table 6: Best solution for case 3 (20-unit system considering transmission loss).

| Generating unit | Lambda method [58] | NR [59] | HM [58] | EHNN [59] | BBO [60] | GSO [61] | BSA [62] | CQGSO [61] | Jaya | TLBO | Hybrid Jaya |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $U_1$ | 512.7805 | 524.0166 | 512.7804 | 403.3043 | 513.0892 | 512.6382 | 510.4477 | 512.730 | 512.7814 | 512.7817 | 512.7814 |
| $U_2$ | 169.1035 | 160.9879 | 169.1035 | 134.4348 | 173.3533 | 169.1817 | 168.3973 | 169.0263 | 169.1009 | 169.1015 | 169.1009 |
| $U_3$ | 126.8898 | 130.2468 | 126.8897 | 134.4348 | 126.9231 | 126.8581 | 125.9721 | 126.8806 | 126.8910 | 126.8907 | 126.8910 |
| $U_4$ | 102.8657 | 100.4129 | 102.8656 | 134.4348 | 103.3292 | 102.8404 | 103.5291 | 102.8723 | 102.8673 | 102.8673 | 102.8673 |
| $U_5$ | 113.6386 | 115.2559 | 113.6836 | 107.5478 | 113.7741 | 113.6931 | 113.8212 | 113.6836 | 113.6826 | 113.6829 | 113.6826 |
| $U_6$ | 73.5710 | 78.7385 | 73.5709 | 67.2174 | 73.0669 | 73.4903 | 73.7901 | 73.5741 | 73.5726 | 73.5720 | 73.5726 |
| $U_7$ | 115.2878 | 118.1765 | 115.2876 | 84.0217 | 114.9843 | 115.1345 | 115.0664 | 115.3037 | 115.2905 | 115.2900 | 115.2905 |
| $U_8$ | 116.3994 | 118.9390 | 116.3994 | 100.8261 | 116.4238 | 116.4033 | 116.3401 | 116.4090 | 116.3998 | 116.3998 | 116.3998 |
| $U_9$ | 100.4062 | 104.7037 | 100.4063 | 134.4348 | 100.6948 | 100.4915 | 100.7093 | 100.4303 | 100.4050 | 100.4049 | 100.4050 |
| $U_{10}$ | 106.0267 | 113.7706 | 106.0267 | 100.8261 | 99.9998 | 106.0393 | 107.1366 | 106.0581 | 106.0279 | 106.0273 | 106.0279 |
| $U_{11}$ | 150.2394 | 148.7055 | 150.2395 | 201.6522 | 148.9770 | 150.3287 | 150.7060 | 150.2337 | 150.2389 | 150.2385 | 150.2389 |
| $U_{12}$ | 292.7648 | 2,959,623 | 292.7647 | 336.0869 | 294.0207 | 292.7353 | 291.1304 | 292.7813 | 292.7658 | 292.7658 | 292.7658 |
| $U_{13}$ | 119.1154 | 118.0200 | 119.1155 | 107.5478 | 119.5754 | 119.1552 | 119.1528 | 119.1165 | 119.1139 | 119.1142 | 119.1139 |
| $U_{14}$ | 30.8340 | 35.4054 | 30.8342 | 87.3826 | 30.5479 | 30.8990 | 32.4521 | 30.8431 | 30.8313 | 30.8315 | 30.8313 |
| $U_{15}$ | 115.8057 | 121.3720 | 115.8056 | 124.3522 | 116.4546 | 115.8099 | 116.1479 | 115.8179 | 115.8057 | 115.806 | 115.8057 |
| $U_{16}$ | 36.2545 | 36.0465 | 36.2545 | 53.7739 | 36.2279 | 36.2584 | 36.2816 | 36.2542 | 36.2543 | 36.2543 | 36.2543 |
| $U_{17}$ | 66.8590 | 72.4530 | 66.8590 | 57.1348 | 66.8594 | 66.8621 | 67.7355 | 66.8611 | 66.8595 | 66.8592 | 66.85952 |
| $U_{18}$ | 87.9720 | 42.2129 | 87.9720 | 80.6609 | 88.5470 | 87.9949 | 87.2547 | 87.9696 | 87.9708 | 87.9712 | 87.9708 |
| $U_{19}$ | 100.8033 | 102.6087 | 100.8033 | 80.6609 | 100.9802 | 100.8210 | 101.5359 | 100.8088 | 100.8025 | 100.8027 | 100.8025 |
| $U_{20}$ | 54.3050 | 55.7560 | 54.3050 | 67.2174 | 54.2725 | 54.3340 | 54.2861 | 54.3106 | 54.3047 | 54.3048 | 54.3047 |
| Total generation (MW) | 2591.9670 | 2593.7615 | 2591.9670 | 2597.9520 | 2592.1011 | 2591.9687 | 2591.8930 | 2591.9650 | 2591.9666 | 2591.9666 | 2591.9666 |
| Power loss | 91.9670 | 93.7615 | 91.9669 | 97.9520 | 92.1011 | 91.9687 | 91.8930 | 91.9650 | 91.9663 | 91.96666 | 91.9666 |
| Total cost ($/h) | 62,456.6391 | 62,489.5000 | 62,456.6341 | 62,610.0000 | 62,456.7793 | 62,456.6332 | 62,456.6925 | 62,456.6330 | 62,456.63309 | 62,456.63309 | 624,56.63309 |
| Average cost ($/h) | NA | NA | NA | NA | 62,456.7928 | 62,456.6336 | 62,457.1517 | 62,456.6330 | 62,457.36592 | 62,456.7400 | 62,456.6671 |
| Maximum cost ($/h) | NA | NA | NA | NA | 62,456.7928 | 62,456.6353 | 62,458.1272 | 62,456.6334 | 62,459.5557 | 62,457.2600 | 62,456.8696 |

Abbreviations: BBO, biography-based optimization; BSA, backtracking spiral algorithm; CQGSO, continuous quick group search algorithm; EHNN, enhanced hopfield neural network; GSO, glowworm swarm optimization; HM, harmony search; NR, Newton–Raphson; TLBO, teaching–learning-based optimization.

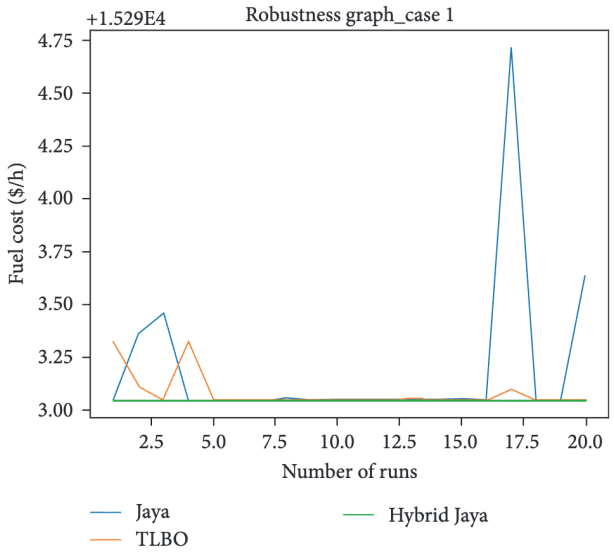


FIGURE 4: Robustness characteristics hybrid Jaya. TLBO, teaching–learning-based optimization.

TABLE 7: Optimal generations schedule (MW) of various algorithms for case study 1.

| Algorithm | $U_1$ | $U_2$ | $U_3$ | $U_4$ | $U_5$ | $U_6$ | $P_L$ | Total cost ($/h) |
|---|---|---|---|---|---|---|---|---|
| GA [43] | 474.8066 | 178.6363 | 262.2089 | 134.2826 | 151.9039 | 74.1812 | 13.0217 | 15,459 |
| PSO [43] | 447.4970 | 173.3221 | 263.4745 | 139.0594 | 165.4761 | 87.1280 | 12.9584 | 15,450 |
| MABC [44] | 447.5039 | 173.3187 | 263.4622 | 139.0650 | 165.4731 | 87.1350 | 12.9582 | 15,449.8995 |
| CBA [45] | 447.4187 | 172.8255 | 264.0759 | 139.2469 | 165.6526 | 86.7652 | 12.9840 | 15,450.2381 |
| NPSO-LRS [46] | 446.9600 | 173.3944 | 262.3436 | 139.5120 | 164.7089 | 89.0162 | 12.9361 | 15,450 |
| MSSA [47] | 447.5029 | 173.3186 | 263.4630 | 139.0656 | 165.4730 | 87.1349 | 12.9580 | 15,449.8995 |
| DE [48] | 447.7440 | 173.4070 | 263.4110 | 139.0760 | 165.3640 | 86.9440 | 12.9570 | 15,449.7660 |
| ST-IRDPSO [49] | 447.5131 | 173.297500 | 263.4668 | 139.0360 | 165.4843 | 87.1605 | 12.9580 | 15,449.8945 |
| SOH-PSO [50] | 438.2100 | 172.5800 | 257.4200 | 141.0900 | 179.3700 | 86.8800 | 12.5500 | 15,446.0200 |
| MCSA [51] | 444.6373 | 174.6410 | 265 | 139.2251 | 165.7121 | 86.6807 | 12.9576 | 15,449.1672 |
| HGS [35] | 446.7435 | 174.2509 | 261.7852 | 143.8329 | 162.0692 | 86.70681 | 12.3885 | 15,444.25963 |
| MTS [52] | 448.1277 | 172.8082 | 262.5932 | 136.9605 | 168.2031 | 87.3304 | 13.0205 | 15,450.0600 |
| HHS [53] | 447.4960 | 173.3140 | 263.4450 | 139.0550 | 165.4750 | 87.1250 | 12.9500 | 15,449 |
| SDO [35] | 446.7266 | 173.1453 | 262.7876 | 143.4916 | 163.9212 | 85.3496 | 12.4220 | 15,444.1863 |
| ESSDO [35] | 446.7121 | 173.1520 | 262.7918 | 143.4897 | 163.9107 | 85.3656 | 12.4219 | 15,444.1863 |
| ESCSDO10 [35] | 446.7116 | 173.1449 | 262.8022 | 143.4911 | 163.9180 | 85.35426 | 12.4220 | 15,444.18631 |
| Rao-1 [54] | 438.5000 | 172.7700 | 266.0100 | 130.0700 | 178.8600 | 89.5090 | 12.7140 | 15,445.8082 |
| Rao-2 [54] | 438.5000 | 172.7700 | 266.0100 | 130.0700 | 178.8600 | 89.50900 | 12.7140 | 15,445.8082 |
| Rao-3 [54] | 438.5000 | 172.7700 | 266.0100 | 130.0700 | 178.8600 | 89.5090 | 12.7140 | 15,445.8082 |
| TLBO | 447.1176 | 171.617400 | 263.5761 | 126.4763 | 171.1611 | 84.3406 | 1.28906 | 15,293.0463 |
| Jaya | 447.1102 | 171.6074 | 263.5453 | 126.5025 | 171.1620 | 84.3615 | 1.2889 | 15,293.0463 |
| Hybrid Jaya | 447.1528 | 171.5934 | 263.6175 | 126.4406 | 171.1284 | 84.3565 | 1.2891 | 15,293.0463 |

Abbreviations: CBA, classification based on association rules; DE, differential evolution; ESSDO, eagle-strategy supply–demand optimizer; GA, genetic algorithm; HHS, hybrid harmony search; MABC, multispecies artificial bee colony; MCSA, modified crow search algorithm; MSSA, multivariate singular spectrum algorithm; MTS, moving target search; NPSO-LRS, new PSO local random search; PSO, particle swarm optimization; SDO, supply–demand-based optimization; SOH-PSO, self-organizing hierarchical particle swarm optimization; ST-IRDPSO, improved random drift particle swarm optimization algorithm; TLBO, teaching–learning-based optimization.

numerical accuracy, and resource efficiency become increasingly crucial in producing superior results for larger systems because algorithms must efficiently navigate a wider search field. Therefore, selecting the best algorithm becomes crucial when handling optimization problems on a large scale. In this specific case, the hybrid Jaya algorithm provides the best possible results.

Figure 5 displays the convergence characteristic curves of Jaya and hybrid Jaya. This figure demonstrates that the hybrid

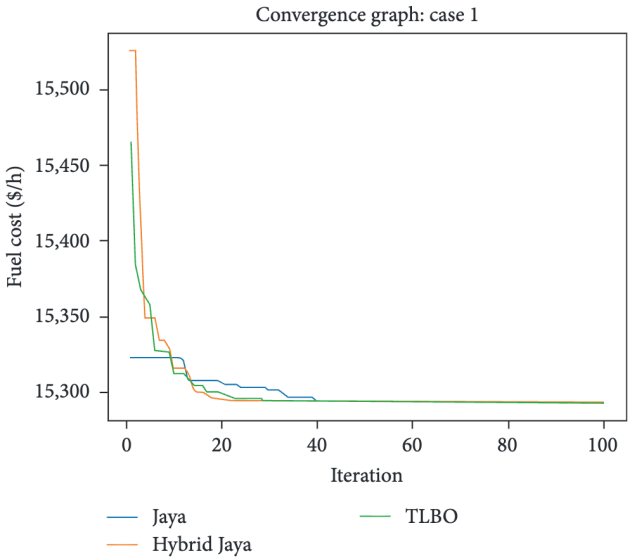


Figure 5: Best case convergence characteristics of Jaya, TLBO, and hybrid Jaya algorithm for 6-unit system. TLBO, teaching–learning-based optimization.

Jaya algorithm converges more than the Jaya algorithm. As a result, it is utilized in the remainder of the paper.

Compared to Jaya, hybrid Jaya regularly reaches a minimum value of 15,293.04631 $/h. This displays hybrid Jaya's outstanding optimization ability. Hybrid Jaya's convergence to a low value also suggests that it has the potential to approach the global minimum, which is evident from the robustness characteristics, shown in Figure 4, which indicates that minimum value is attained in almost every individual run. In contrast, TLBO displays varying values and lacks a distinct convergence pattern. Thus, hybrid Jaya's resilience is demonstrated in Figure 5 since it converges to a lower minimum value faster and exhibits a steady convergence pattern, outperforming both basic Jaya and TLBO regarding convergence properties.

Fuel costs from 20 individual runs of the Jaya, hybrid Jaya, and TLBO algorithms make it clear that the hybrid Jaya algorithm is reliable. Throughout the 20 individual runs, the fuel cost for hybrid Jaya ranges from 15,293.04631 to 15,293.04646 $/h. Jaya, on the other hand, shows minor changes with a range of 15,293.04646 to 15,294.7127 $/h, and TLBO has more variability with a range of 15,293.04631 to 15,293.32391 $/h. Additionally, hybrid Jaya constantly offers the lowest fuel cost of 15,293.04631 $/h, demonstrating its capacity to discover the best minimal fuel cost. Due to its stability and low fuel cost variability, the hybrid Jaya algorithm outperforms Jaya and TLBO in fuel cost optimization, demonstrating its resilience and dependability. While the Jaya algorithm's fuel cost values vary, hybrid Jaya's remain consistent throughout, showing better stability and reliability. This shows that hybrid Jaya is more stable regarding fuel cost than the Jaya algorithm, as it keeps almost the same value without variation.

Moreover, a complete examination demonstrates the superiority of the hybrid Jaya algorithm over TLBO and Jaya in the optimization issue involving six thermal units and a 1263 MW load demand. Statistical analyses demonstrated a statistically significant difference in favor of hybrid Jaya in Table 8. The sensitivity study demonstrated the algorithm's robustness and

Table 8: Best solution values obtained from total fuel cost for the 40-unit system with a load demand of 10,500 MW.

| Algorithm | Best fuel cost ($/h) | Average fuel cost ($/h) |
|---|---|---|
| HHO [1] | 121,731.9500 | 122,343.1700 |
| ABC [63] | 121,441.0300 | 121,995.8200 |
| HHO-A βHC [1] | 121,414.8300 | 121,468.7400 |
| ABOMDE [64] | 121,414.8700 | 121,487.8500 |
| ARCGA [65] | 121,415.5000 | 121,462.1500 |
| βHC [66] | 121,414.6800 | 121,496.8400 |
| CBPSO-RVM [67] | 121,555.3200 | 122,281.1400 |
| CLCS-CLM [68] | 121,412.5400 | 121,412.9900 |
| CSOMA [69] | 121,414.7000 | 121,415.0500 |
| CS [68] | 121,457.1600 | 121,512.3000 |
| CS-CLM [68] | 121,450.4100 | 121,503.5500 |
| FAPSO [70] | 121,712.4000 | 121,778.2500 |
| FCASO-SQP [71] | 121,456.9800 | 122,026.2100 |
| FAPSO-NM [70] | 121,418.3000 | 121,418.8000 |
| FFA [72] | 121,415.0500 | 121,416.5700 |
| GWO [73] | 121,488.4000 | NA |
| HGWO [74] | 121,412.0000 | 121,419.0000 |
| HCASO [71] | 121,865.6300 | 122,100.7400 |
| HCPSO [75] | 121,865.2300 | 122,100.8700 |
| HMAPSO [76] | 121,586.9000 | 121,586.9000 |
| HQIPSO [77] | 121,418.6000 | 121,427.4700 |
| HSSA [78] | 121,960.2700 | 122,239.5300 |
| NDS [79] | 121,647.4000 | 121,647.4000 |
| NSABC [80] | 122,270.9100 | 123,675.4000 |
| NUHS [81] | 121,412.7400 | 121,549.9500 |
| RCGA [77] | 121,418.7200 | NA |
| SOMA [69] | 121,418.7900 | 121,449.8800 |
| THS [82] | 121,425.1500 | 121,528.6500 |
| TSARGA [83] | 121,463.0700 | 122,928.3100 |
| TLA [84] | 122,009.7700 | 122,074.9000 |
| TLBO | 119,167.9000 | 120,529.5000 |
| Jaya | 119,000.4000 | 120,410.3000 |
| Hybrid Jaya | 118,666.1000 | 119,198.3000 |

Abbreviations: ABC, artificial bee colony; ABOMDE, accelerated biogeography-based optimization differential evolution; ARCGA, adaptive real coded genetic algorithm; CLCS-CLM, comprehensive learning cuckoo search with chaos-lambda method; CSOMA, cultural self-organizing migrating algorithm; FAPSO, fuzzy adaptive particle swarm optimization; FAPSO-NM, fuzzy adaptive particle swarm optimization with Nelder–Mead; FCASO-SQP, fuzzy adaptive chaotic ant swarm optimization-sequential quadratic programing; FFA, firefly algorithm; GWO, gray wolf optimization; HCASO, hybrid chaotic ant swarm optimization; HCPSO, hybrid coding particle swarm optimization; HGWO, hybrid gray wolf optimizer; HHO, Harris hawk optimizer; HHO-A βHC, Harris hawk optimizer with adaptive beta hill climbing; HMAPSO, hybrid multiagent particle swarm optimization; HQIPSO, hybrid quantum inspired particle swarm optimization; HSSA, hybrid salp swarm algorithm; NDS, novel direct search; NSABC, nondominated sorting based artificial bee colony; NUHS, natural updated harmony search; RCGA, real coded genetic algorithm; SOMA, self-organizing migrating algorithm; THS, tournament-based harmony search; TLA, teaching–learning algorithm; TLBO, teaching–learning-based optimization; TSARGA, Taguchi self-adaptive real-coded genetic algorithm.

flexibility to parameter modifications. Across more test situations, hybrid Jaya continuously beat the other methods, demonstrating its generalizability. The convergence graph in Figure 5 further revealed that hybrid Jaya converges faster and achieves better solutions. Its ability to utilize the strengths of both optimization techniques, successfully explore solution spaces, and manage equality constraints further strengthens its dominance. Furthermore, hybrid Jaya regularly outperformed traditional optimization techniques. This collected data, supported by current research, indisputably validates hybrid Jaya as the ideal solution for addressing the optimization problem, demonstrating its practical effectiveness and dependability.

*4.1.2. Test Case 2: 13 Units With an 1800 MW Load Demand.* In this instance, a test system with 13 thermal units is used to implement the hybrid Jaya, Jaya, standard SDO, ESSDO, HGS, and other well-known algorithms to solve the ELD problem with an 1800 MW load requirement. Without considering power losses, the fuel cost using VPLE is calculated. Appendix C contains the system data.

Table 5 provides the best-scheduled powers (MW) for generating unit 1 to generating unit 13, as determined by the hybrid Jaya algorithm. Table 5 makes it evident that the proposed hybrid Jaya offers a solution more affordably (and with higher quality) than the other methods. A comparative analysis between Jaya, TLBO, hybrid Jaya, and some other recent algorithms is listed in Table 9. In Figure 6, the suggested hybrid Jaya, Jaya, and TLBO algorithms' convergence characteristic curves are shown for this situation. The suggested algorithm converges in comparatively fewer cycles, demonstrating good convergence qualities and resulting in a cheap generation cost, as demonstrated in the convergence characteristic curves in Figure 6. The hybrid Jaya algorithm achieves the global minimal fuel compared to the other algorithms.

Figure 6 shows the best-case convergence characteristics comparison of Jaya, TLBO, and the hybrid Jaya algorithm. The figure shows that the TLBO method differs from the other two algorithms in terms of cost reduction and achieving global minima. The TLBO algorithm continuously has more significant fuel costs throughout the iterations than the Jaya and hybrid Jaya algorithms. Hybrid Jaya reaches global optima faster than the Jaya algorithm. After 500 iterations, the Jaya algorithm gives a result of 17,932.76036 $/h, while hybrid Jaya produced a value of 17,932.65587 $/h.

According to the robustness characteristics shown in Figure 7, the hybrid Jaya algorithm performs better than the Jaya algorithm at minimizing fuel costs. Throughout the 20 runs, the hybrid Jaya algorithm's average fuel cost of 17,932.77432 $/h is cheaper than the basic Jaya algorithm's 17,933.10838 $/h. This shows that the hybrid Jaya algorithm consistently produces output with lower fuel costs. Thus, it can be inferred from the data that the hybrid Jaya algorithm outperforms the basic Jaya algorithm in reducing fuel costs. The hybrid Jaya algorithm is more reliable than the basic Jaya algorithm. The lower mean fuel cost achieved by the hybrid Jaya algorithm suggests more excellent performance in fuel cost minimization. In this test situation, TLBO underperforms Jaya and hybrid Jaya in terms of minimizing fuel costs. According to the average fuel costs over 20 runs, hybrid Jaya has the lowest average of 17,933.16 $/h, Jaya is second at 17,932.83 $/h, and TLBO has the highest average of 17,936.61 $/h.

The hybrid Jaya algorithm performs better than the Jaya algorithm. Compared to the basic Jaya algorithm, the hybrid Jaya algorithm has a minimum fuel cost value of 17,932.65587 $/h. With a maximum value of 17,932.95598 $/h, the hybrid Jaya algorithm performs noticeably better than the Jaya algorithm, which has a worst value of 17,935.86917 $/h. The median value of the hybrid Jaya algorithm is better at 17,932.77367 $/h, while the median value of the Jaya algorithm is marginally higher at 17,933.19303 $/h. The hybrid Jaya algorithm's superiority is further highlighted by the standard deviation (STD), which contrasts the hybrid Jaya algorithm's value of 0.084509078 with the basic Jaya algorithm's value of 0.938524044.

The findings from Tables 4 and 7 shed important light on how well different optimization strategies perform in solving the ELD problem for a 13-unit system with an 1800 MW load demand. The overall expenses for TLBO, Jaya, and hybrid Jaya do appear to be relatively similar. However, to prove the superiority of the hybrid Jaya algorithm, it is necessary to look more closely at the study.

Take into account the "best" column in Table 4, which shows the most cost-effective algorithm overall. When compared to other algorithms, hybrid Jaya regularly has one of the lowest total costs, matching their top performance. According to this, hybrid Jaya is one of the best algorithms for solving this issue.

The average overall cost for hybrid Jaya across several runs is competitive, falling within a constrained range comparable to the top methods, as shown by the "mean" column in Table 4. This demonstrates how hybrid Jaya consistently offers the best solutions that are close to ideal.

Additionally, hybrid Jaya's STD is remarkably low (0.084509078), demonstrating excellent reliability and accuracy in continually providing cost-effective solutions. In comparison, TLBO and Jaya have STD values that are comparatively higher (3.481836968 and 0.938524, respectively), indicating that their performance is more variable.

In addition, the convergence characteristic curves shown in the text demonstrate that hybrid Jaya converges faster than TLBO and Jaya, showcasing improved convergence qualities critical in real-world situations where computational efficiency is vital.

The analysis of the "best," "mean," and STD values, along with convergence characteristics, confirms the validity of the claim that hybrid Jaya is indeed a superior algorithm for solving the ELD problem in this 13-unit system with an 1800 MW load demand, even though the total cost values for TLBO, Jaya, and hybrid Jaya may appear similar. It is an appealing option for real-world applications due to its reliable performance, minimal variability, and effective convergence.

*4.1.3. Test Case 3: 20 Units With a Load Demand of 2500 MW.* The demand for this system, which comprises 20 generating units, is 2500 MW. The loss coefficients are given in

Table 9: Comparison of the different metrics for case 2 (13-unit system).

| Algorithm | HGS [35] | SDO [35] | ESSDO [35] | ESCSDO10 [35] | PSO [46] | $\theta$-PSO [56] | MTVPSO [57] | JAYA | TLBO | HYBRID JAYA |
|---|---|---|---|---|---|---|---|---|---|---|
| Best | 18,552.1900 | 18,041.4900 | 18,028.5600 | 18,028.1900 | 18,014.1600 | 17,963.82970 | 17,960.3659 | 17,932.76036 | 17,932.6903 | 17,932.65587 |
| Worst | 19,091.2600 | 18,160.1800 | 18,186.7000 | 18,184.3500 | 18,104.6500 | 17,980.2030 | 17,961.0615 | 17,935.86917 | 17,946.03656 | 17,932.95598 |
| Mean | 18,816.5000 | 18,092.7900 | 18,090.5000 | 18,106.8700 | 18,104.6500 | 17,965.2055 | 17,960.4226 | 17,933.1930 | 17,936.0019 | 17,932.7736 |
| STD | 179.7230 | 26.0366 | 32.1778 | 40.5607 | NA | 4.3807 | 1.9200 | 0.9385 | 3.48183 | 0.0845 |

Abbreviations: $\theta$-PSO, theta-PSO; ESSDO, eagle-strategy supply–demand optimizer; HGS, hunger games search; MTVPSO, modified time varying PSO; PSO, particle swarm optimization; SDO, supply–demand-based optimization; STD, standard deviation; TLBO, teaching–learning-based optimization.

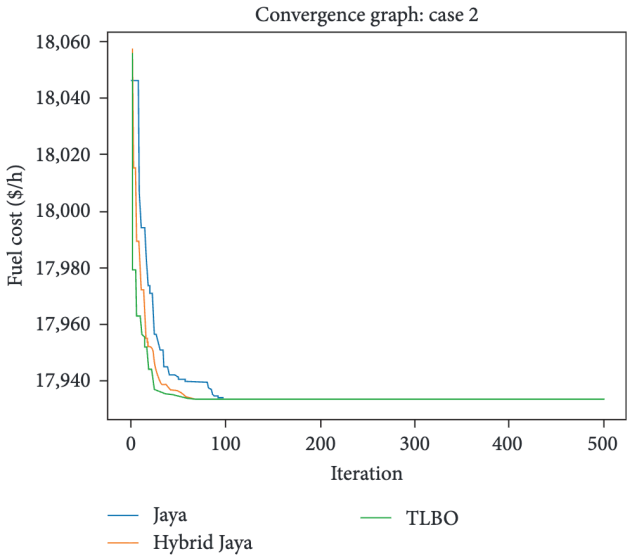


Figure 6: Convergence characteristics of Jaya and hybrid Jaya algorithm. TLBO, teaching–learning-based optimization.

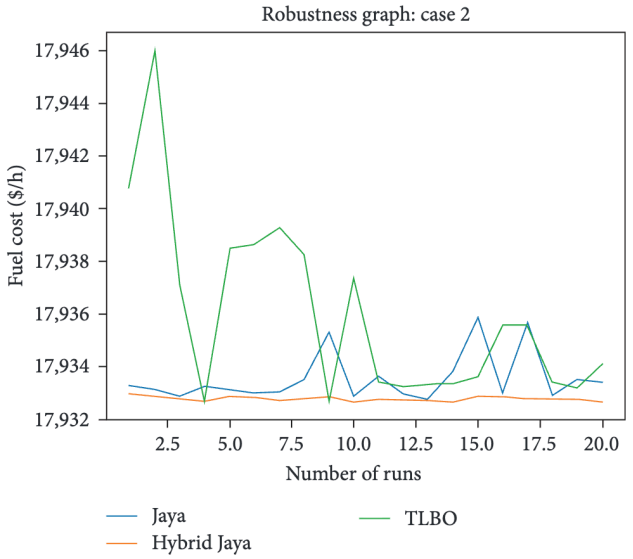


Figure 7: Robustness characteristics comparison between Jaya, TLBO, and hybrid Jaya for 13-unit system. TLBO, teaching–learning-based optimization.

Appendix D and the transmission loss is considered. The data for the unit are summarized in Appendix E.

Table 6 shows the optimal ELD schedule for generating unit 1 to generating unit 20 ($U_1 - U_{20}$) along with some traditional and evolutionary methods (for this particular case study: biography-based optimization [BBO] [60], lambda iteration [58], harmony search [HM] [58], Newton–Raphson [NR] [59], enhanced hopfield neural network [EHNN] [59], glowworm swarm optimization [GSO] [61], and continuous quick group search algorithm [CQGSO] [61]). A comparison reveals that hybrid Jaya converges to the optimal value at the same level as or lower than other approaches.

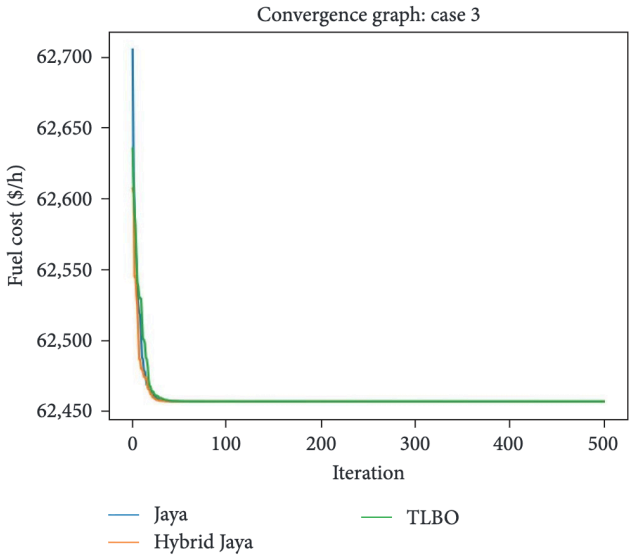


Figure 8: Best case convergence characteristics of Jaya, TLBO, and hybrid Jaya algorithm for the 20-unit system. TLBO, teaching–learning-based optimization.

Figure 8 illustrates the convergence of total generation fuel costs for Jaya, TLBO, and hybrid Jaya among the 20 trial runs of this case study's solutions.

The hybrid Jaya approach stands out from the others in the chart because it offers the lowest total cost and average cost. It results in a 62,456.63309 $/h total cost and a 62,456.66712 $/h average cost. These results are more significant than those from other techniques, including NR, HM, EHNN, BBO, GSO, backtracking spiral algorithm [BSA], CQGSO, Jaya, and TLBO. Additionally, the hybrid Jaya approach performs well when compared to the maximum cost ($/h), which is 62,457.26 $/h, suggesting that it has less cost fluctuation than some other methods like CQGSO and TLBO. The hybrid Jaya method consistently generates lower total cost and average cost values for the 20 generating units than other well-known optimization techniques, demonstrating that it is superior in fuel cost minimization.

Over 500 iterations, the convergence fuel cost values for the three algorithms, Jaya, TLBO, and hybrid Jaya, were examined. Compared to the other two methods, the hybrid Jaya algorithm consistently showed faster convergence to both the minimum value and the global minimum. The fuel cost for the hybrid Jaya algorithm was 62,606.85861 $/h at the first iteration, compared to 62,704.89073 and 62,635.08481 $/h for the Jaya and TLBO algorithms. The hybrid Jaya algorithm consistently outperformed the other algorithms throughout the iterations, and this pattern persisted. For instance, the hybrid Jaya algorithm reached a fuel cost of 62,456.66615 $/h at iteration 50, but the fuel cost for the Jaya and TLBO algorithms were 62,456.71986 and 62,456.80135 $/h, respectively. The final value of the hybrid Jaya algorithm was 62,456.63309 $/h, while the final values of the Jaya and TLBO algorithms were 62,456.63341 and 62,456.63312 $/h, respectively. This indicates that the hybrid Jaya method consistently approached the global minimum more closely than the other

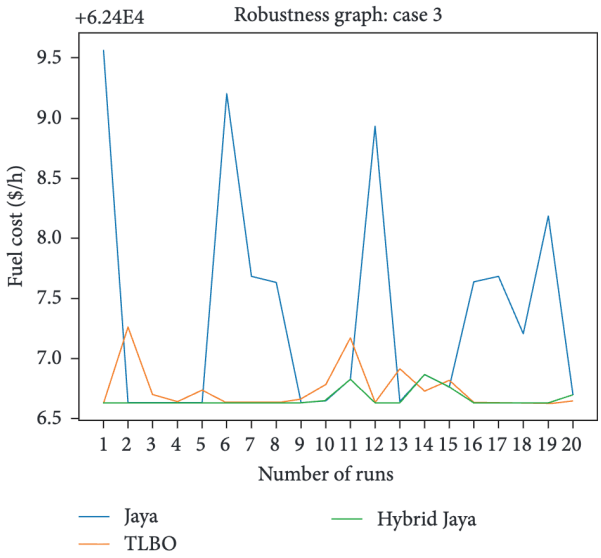


FIGURE 9: Robustness characteristics of hybrid Jaya over 20 trial runs. TLBO, teaching–learning-based optimization.

algorithms. These findings unequivocally demonstrate that the hybrid Jaya algorithm outperforms the Jaya and TLBO algorithms regarding convergence speed and precision. Figure 8 shows the convergence characteristics of the Jaya, TLBO, and hybrid Jaya algorithms for the best-case scenario.

Among the 20 individual trial runs of this case study's solutions, Figure 9 illustrates the robustness of Jaya, TLBO, and hybrid Jaya.

Across all 20 individual runs, hybrid Jaya consistently outperformed the Jaya algorithm and TLBO algorithm in terms of minimizing fuel costs. Its average fuel price, 62,456.67124 $/h, is less than that of TLBO 62,456.82395 and Jaya 62,457.07409 $/h. The limited variance in its fuel cost values (e.g., run 1: 62,456.63314 $/h, run 10: 62,456.64952 $/h, run 20: 62,456.69632 $/h) further demonstrates hybrid Jaya's stability and resilience. TLBO and Jaya, on the other hand, both show higher average fuel costs (e.g., TLBO: 62,456.82395 $/h, basic Jaya: 62,457.07409 $/h) and more notable changes in fuel cost values (e.g., Jaya: run 1: 62,459.55571 $/h, run 10: 62,456.63309 $/h, run 20: 62,456.69632 $/h). Hybrid Jaya performs better than its competitors by constantly reducing fuel cost and preserving stability throughout the course of the 20 runs.

*4.1.4. Test Case 4: 40 Units With a Load Demand of 10,500 MW.* This scenario uses the VPLE to solve the ELD problem on a test system with 40 thermal units without considering power loss. The system has a 10,500 MW total load demand, and the information used is from Appendix F. In the search space for finding a solution, there are more local minima when working with larger power systems. This demonstrates the necessity of optimization methods with improved search capabilities to avoid premature convergence.

Table 10 provides the optimal scheduled powers in MW obtained for generating unit 1 to unit 40 ($U_1 - U_{40}$) through various algorithms such as Jaya, hybrid Jaya, eagle-strategy supply–demand based optimization algorithm with chaotic version 10 (EESCSDO10), SDO, ESSDO, HGS, DE, TLBO, and HSSA. Hybrid Jaya outperforms all other techniques in terms of achieving the best results. The convergence curve and boxplots for hybrid Jaya and the other techniques are illustrated in Figure 10. To further evaluate the performance, Table 8 compares the minimum and average fuel cost values with some of the well-known and recent algorithms reported in the literature. The hybrid Jaya algorithm achieves the best average value among the widely commonly compared techniques in the literature illustrated in Table 8.

As illustrated in the best-case convergence graph of Figure 10. Hybrid Jaya's combination of exploration and exploitation operations helps it continuously explore fruitful search space areas, improving convergence and minimizing objective function values. As a result, the evidence suggests that hybrid Jaya outperforms Jaya and TLBO algorithm in terms of minimization with iteration due to its capacity to combine optimization techniques with adaptability.

The results of 20 individual runs shown in Figure 11 show that the hybrid Jaya algorithm regularly outperforms basic Jaya in terms of cost minimization and robustness. In comparison to basic Jaya, the hybrid Jaya algorithm produced reduced cost values throughout all runs. Cost figures for the Jaya algorithm ranged from 119,000.3672 to 122,501.7593 $/h, while those for hybrid Jaya ranged from 118,666.1433 to 119,763.4359 $/h. Hybrid Jaya's average cost was 119,285.0171 $/h, which was considerably less than Jaya's average cost of 120,290.1795 $/h. In addition, compared to the Jaya algorithm, hybrid Jaya's performance was more consistent and stable, and its cost values varied less. These findings show that, compared to the Jaya algorithm, the hybrid Jaya algorithm consistently lowers costs more successfully and demonstrates better robustness, making it the algorithm of choice for cost optimization for ELD.

*4.1.5. Test Case 5: East Java-Indonesia 10 Units With a Load Demand of 616 MW.* As illustrated in Figure 12, the fifth case study is implemented in the East Java 150 kV power system,

Table 10: Optimal scheduled power obtained through various algorithms for test case 4.

| Algorithm | DE [85] | HSSA [78] | QOPO [86] | HGS [35] | ESSDO [35] | SDO [35] | ESCSDO10 [35] | TLBO | Jaya | Hybrid Jaya |
|---|---|---|---|---|---|---|---|---|---|---|
| U1 | 110.9520 | 113.9994 | 113.7611 | 114 | 111.3958 | 112.1391 | 110.8002 | 113.8627 | 112.4252 | 110.8160 |
| U2 | 113.3000 | 113.9999 | 113.9886 | 114 | 111.7094 | 113.1060 | 110.7814 | 112.8244 | 113.9999 | 109.6472 |
| U3 | 98.6160 | 119.999 | 119.9993 | 60.1394 | 97.4077 | 97.4006 | 97.3987 | 68.0519 | 117.5388 | 71.7319 |
| U4 | 184.1490 | 189.9998 | 189.8949 | 179.7998 | 179.7576 | 179.7355 | 179.7312 | 188.4780 | 187.2487 | 188.2762 |
| U5 | 86.4010 | 97 | 97 | 97 | 89.1816 | 87.8015 | 87.7911 | 96.5550 | 92.7811 | 72.7449 |
| U6 | 140 | 140 | 139.9986 | 140 | 105.5467 | 105.4210 | 139.9984 | 138.6233 | 140 | 111.5046 |
| U7 | 300 | 280.6695 | 300 | 300 | 259.6439 | 259.6137 | 259.5963 | 299.6909 | 297.825 | 299.9319 |
| U8 | 285.4560 | 290.4195 | 299.9992 | 300 | 284.5999 | 284.6095 | 284.5963 | 299.5610 | 299.9433 | 294.8961 |
| U9 | 297.5110 | 292.5067 | 299.9970 | 300 | 284.6077 | 284.5989 | 284.5915 | 298.4900 | 299.9999 | 273.1729 |
| U10 | 130 | 130.0001 | 130 | 130 | 130.0001 | 204.8536 | 130.0107 | 132.4978 | 130.2949 | 196.8798 |
| U11 | 168.7480 | 168.7998 | 94.0011 | 94.0006 | 94.0001 | 168.8005 | 239.0063 | 99.2196 | 94 | 354.2813 |
| U12 | 95.6950 | 168.7998 | 94.3599 | 94 | 168.8010 | 168.8032 | 166.3011 | 95.0155 | 94 | 357.1914 |
| U13 | 125 | 214.7598 | 125.1963 | 125 | 394.2750 | 304.5121 | 214.7574 | 154.9432 | 125 | 125.0505 |
| U14 | 394.3550 | 304.5196 | 304.5059 | 394.2847 | 394.2988 | 394.2813 | 304.5057 | 253.0544 | 291.7584 | 231.2003 |
| U15 | 305.523 | 304.5196 | 394.4782 | 394.2796 | 304.5235 | 304.5089 | 394.2829 | 378.9166 | 287.8407 | 387.0639 |
| U16 | 394.7150 | 394.2793 | 394.2599 | 394.2794 | 394.2794 | 304.5268 | 394.2769 | 257.1037 | 286.9811 | 243.4541 |
| U17 | 489.7970 | 489.2812 | 489.3310 | 489.3126 | 489.2851 | 489.3018 | 489.2760 | 499.8478 | 500 | 499.9992 |
| U18 | 489.3620 | 489.2795 | 489.4125 | 489.2963 | 489.3087 | 489.2864 | 489.2794 | 498.5064 | 499.7137 | 491.1156 |
| U19 | 520.9020 | 511.2795 | 511.2939 | 511.2809 | 511.3006 | 511.2827 | 511.2789 | 549.7678 | 550 | 548.2185 |
| U20 | 510.6410 | 511.2795 | 511.4796 | 511.3354 | 511.2800 | 511.33390 | 511.2788 | 549.4886 | 550 | 382.1751 |
| U21 | 524.5340 | 523.2794 | 525.4610 | 523.5524 | 523.2836 | 523.3314 | 523.2804 | 527.8708 | 549.9867 | 521.9748 |
| U22 | 526.6980 | 523.7119 | 523.5896 | 523.3660 | 523.2759 | 523.2799 | 523.2774 | 549.4911 | 549.9239 | 402.804 |
| U23 | 530.7470 | 523.2794 | 526.5896 | 539.7297 | 523.2844 | 523.2755 | 523.2791 | 548.0739 | 550 | 549.4699 |
| U24 | 526.3270 | 523.2795 | 524.3047 | 523.8539 | 523.2795 | 523.4085 | 523.2744 | 549.9884 | 550 | 536.2256 |
| U25 | 525.6540 | 523.8181 | 524.7312 | 5,233,009 | 523.3598 | 523.2736 | 523.2783 | 528.8443 | 550 | 549.9870 |
| U26 | 522.9500 | 523.2794 | 523.5606 | 523.3843 | 523.3029 | 523.2832 | 523.2836 | 549.8235 | 550 | 548.7541 |
| U27 | 10 | 13.8386 | 10 | 10 | 10.0190 | 10.0630 | 10.0024 | 10.0848 | 10.0002 | 15.8095 |
| U28 | 11.5520 | 11.7804 | 10.22341 | 10 | 10.0149 | 10.0080 | 10.0007 | 15.2925 | 10 | 10.0004 |
| U29 | 10 | 10.1857 | 10.0892 | 10 | 10.00053 | 10.00798 | 10.00303 | 11.4367 | 10 | 17.8544 |
| U30 | 89.9080 | 94.8277 | 97 | 97 | 90.3466 | 89.64376 | 87.7960 | 97 | 48.7476 | 96.5777 |
| U31 | 190 | 189.9999 | 190 | 190 | 161.6513 | 161.1230 | 189.9963 | 189.9987 | 189.9928 | 173.3851 |
| U32 | 190 | 181.8970 | 189.9998 | 190 | 189.9978 | 189.9989 | 189.9999 | 189.2053 | 189.9999 | 189.9927 |
| U33 | 190 | 187.1620 | 190 | 190 | 161.1450 | 189.9791 | 189.9845 | 189.9977 | 190 | 189.9937 |
| U34 | 198.8400 | 164.8096 | 200 | 172.5026 | 164.8925 | 164.7992 | 164.8044 | 196.5762 | 200 | 109.2134 |
| U35 | 174.1780 | 180.4562 | 200 | 200 | 164.9190 | 164.9028 | 164.8001 | 190.6148 | 200 | 180.7268 |
| U36 | 197.1600 | 199.9853 | 199.9999 | 200 | 164.9784 | 164.8301 | 164.7969 | 194.6316 | 200 | 199.8789 |
| U37 | 110 | 108.2447 | 109.9804 | 110 | 95.7665 | 109.9967 | 89.0969 | 106.7947 | 110 | 89.4107 |
| U38 | 109.3570 | 90.2908 | 110 | 110 | 109.9943 | 97.5983 | 89.1136 | 109.9147 | 110 | 108.7641 |
| U39 | 110 | 89.2026 | 110 | 110 | 109.9976 | 109.9998 | 89.1141 | 109.8661 | 109.9987 | 109.8263 |
| U40 | 510.9750 | 511.2793 | 511.4108 | 511.3023 | 511.2872 | 511.2801 | 511.2784 | 550 | 550 | 549.9993 |
| Cost ($/h) | 121.8400 | 121960.3000 | 121789.6000 | 121669.5000 | 121873.6000 | 121750.2000 | 121,627 | 119167.8716 | 119000.3672 | 118666.1000 |

Abbreviations: DE, differential evolution; ESSDO, eagle-strategy supply–demand optimizer; HGS, hunger games search; HSSA, hybrid salp swarm algorithm; QOPO, quasi-oppositional search-based political optimizer; SDO, supply–demand-based optimization; TLBO, teaching–learning-based optimization.

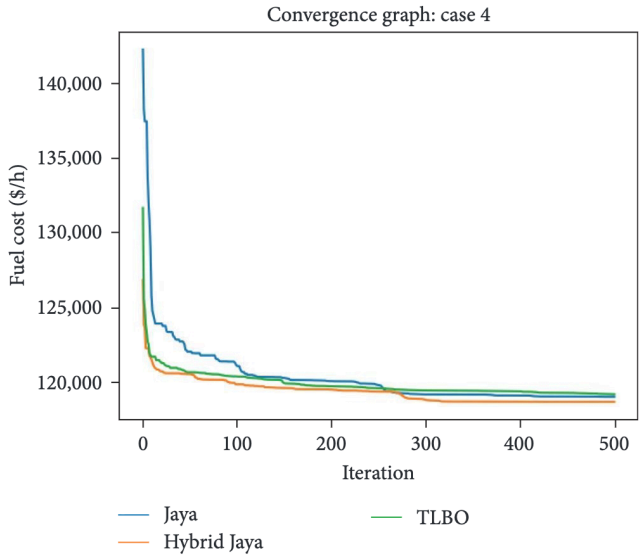


Figure 10: Best case convergence characteristics comparison between Jaya, TLBO, and hybrid Jaya. TLBO, teaching–learning-based optimization.

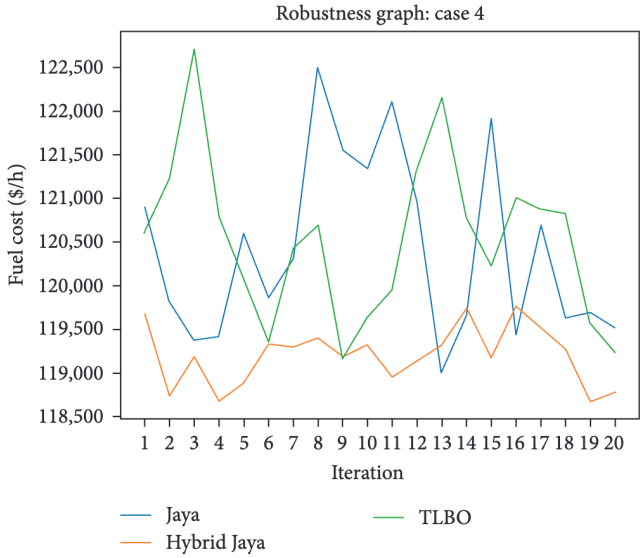


Figure 11: Robustness characteristics of Jaya and hybrid Jaya. TLBO, teaching–learning-based optimization.

which is situated in East Java, Indonesia. This power system consists of 10 generating units that play a crucial role in supplying electricity to the region. The Gresik 1 electric steam power plant, also known simply as Gresik 1, houses units 1–4 and is a key facility within this system. Located strategically to optimize power distribution, Gresik 1 is integral to maintaining the stability and reliability of the power grid in East Java. The implementation of this case study in the East Java power system highlights the practical applicability and effectiveness of the proposed methodologies in real-world power systems, demonstrating their potential for enhancing operational efficiency and meeting the growing energy demands of the region. Units 5 and 6 of the Pembangkit Listrik Tenaga Uap Gresik 1 (PLTU Gresik 1) are situated in the Perak electric steam power plant known as PLTU Perak in Indonesia and Gresik 2 integrated gasification combined cycle plants, which are in Indonesia, are where units 7–10 are situated is known as PLTGU Gresik 2, or pembangkit listrik tenaga gas dan uap Gresik 2. Figure 13 shows the single line diagram of the East Java 150 KV line. Generator unit capacity and cost coefficients are shown in Appendix G.

Table 11 compares the outcomes of the hybrid Jaya with those of other algorithms, including the BA, GA, PSO, directional bat algorithm (dBA) and modified inertial weight (MIW)-based PSO. When compared to all other listed methodologies, the hybrid Jaya offers a cheap fuel cost, 95,633 $/h, proving its viability. This economic fuel cost is also represented in Figure 14 convergence characteristics for the Jaya, TLBO, and hybrid Jaya. The convergence characteristics are drawn for 500 iterations for comparison.

It was discovered that all three algorithms—hybrid Jaya, Jaya, and TLBO—successfully obtained the same minimum value of 95,632.12566 $/h in the comparative convergence

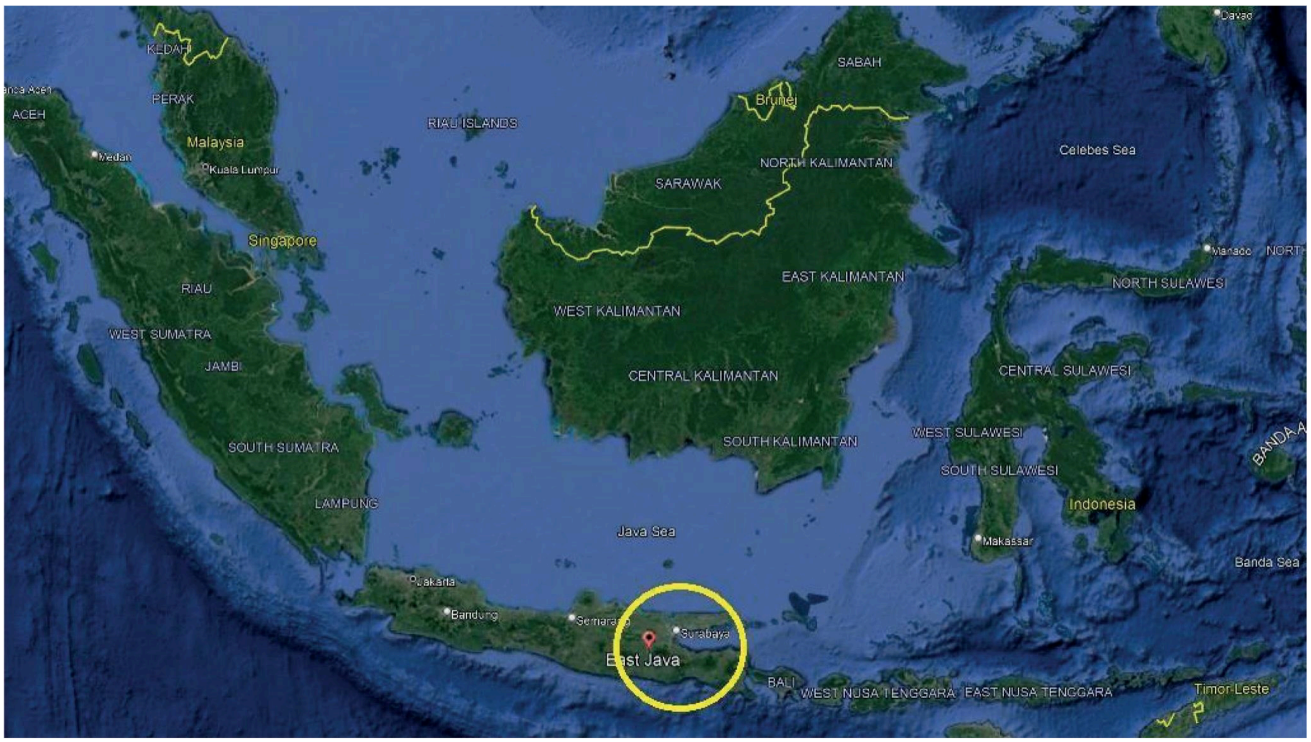


FIGURE 12: The location of the 150 kV electricity system in East Java.

analysis of optimization methods that included these. However, as shown in Figure 14, it took them various amounts of iterations to get this convergence. The TLBO and Jaya algorithms required 227 and 122 iterations, respectively, but the hybrid Jaya algorithm reached the minimal value after just 189 iterations. According to these findings, the hybrid Jaya method can converge more quickly and efficiently to obtain a minimal fuel cost compared to the essential Jaya and TLBO algorithms.

As shown in Figure 15, over 20 runs, the hybrid Jaya algorithm consistently achieves a mean fuel cost of $ 95,632.12652, which is lower than Jaya's mean of $ 95,726.32549 and TLBO's mean of $ 95,633.05032. This performance implies its robust optimization capabilities, which could be attributed to the successful integration of Jaya and TLBO strengths, a balanced exploration–exploitation strategy, domain-specific features, fine-tuned parameters, and consistent convergence. The lower mean cost of the hybrid Jaya algorithm indicates its ability to continuously generate cost-effective solutions, validating its potential as a robust optimization strategy.

Despite producing similar generator contributions and overall costs as TLBO and Jaya, the hybrid Jaya algorithm's efficiency and effectiveness can be ascribed to several vital aspects. First, it shows a far faster convergence rate, achieving the lowest fuel cost in just 189 iterations as opposed to 227 iterations for TLBO and 122 iterations for Jaya, which is a significant advantage in computationally challenging real applications. Its reliability in producing cost-effective solutions while maintaining stability across multiple iterations—a crucial quality for real-world optimization scenarios—is further demonstrated by its consistent achievement of a lower mean fuel cost over 20 runs, $95,632.12652, in comparison to Jaya ($95,726.32549) and TLBO ($95,633.05032).

## 5. Sensitivity and Convergence Analysis

The suggested hybrid Jaya algorithm is applied to the situations above for 20 independent runs to evaluate its sensitivity, and the results are then contrasted with those from Jaya and TLBO separately. The box plot of Jaya, TLBO, and Jaya–TLBO for cases 1–4 is shown in Figures 16–20. A well-known statistical method called the "box plot" summarizes and compares the results visually. The "box plot" aids in locating any hidden patterns among a collection of numbers. For test cases 1–5, respectively, Figures 16–20 show the box plot for the Jaya–TLBO, Jaya, and TLBO algorithms.

The four case study's boxplots are displayed in Figures 16–20. As can be observed, when compared to those of the Jaya and TLBO, the boxplots of the suggested hybrid Jaya algorithms are narrow and have the lowest values.

To determine the efficacy of the proposed hybrid Jaya algorithm, this paper uses a *T*-test analysis. A 5% confidence interval is taken and applied to the sampling data. This data is acquired over 20 individual runs for both the algorithms, Jaya and hybrid Jaya. Tables 12 and 13 display the conclusions drawn from the information. The null hypothesis is "There is no significant difference between the two algorithms." The test statics are translated to a *p*-value, and the results are derived using the following formula:

1. The observed difference is not significant when the *p*-value exceeds 0.10.

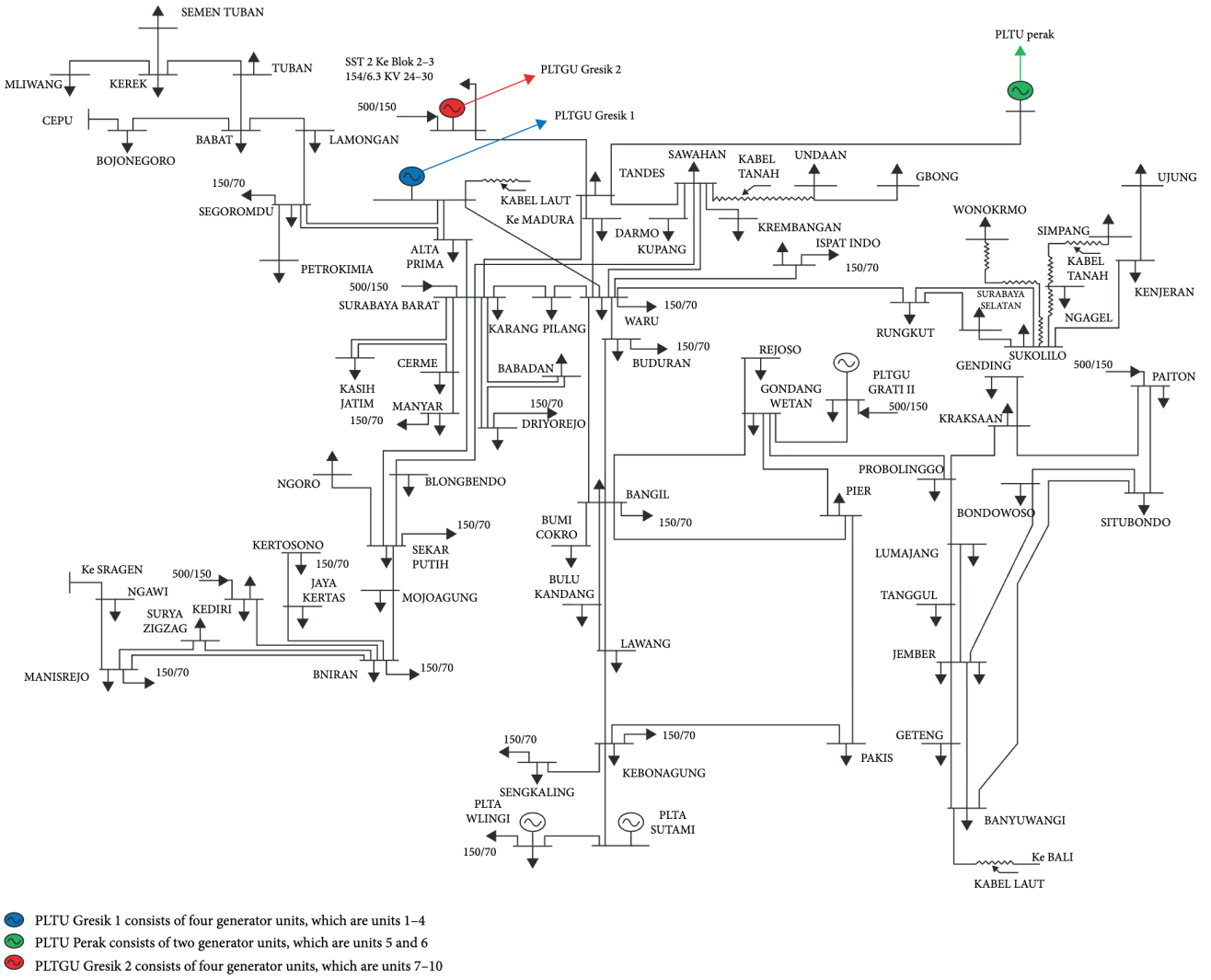


FIGURE 13: East Java power system single line diagram [87]. PLTU, Pembangkit Listrik Tenaga Uap.

TABLE 11: Best solution values obtained from total fuel cost for the 40-unit system with a load demand of 616 MW.

| Generating Unit | BA [88] | dBA [88] | PSO [87] | GA [88] | MIW-PSO [87] | Jaya | TLBO | Hybrid Jaya |
|---|---|---|---|---|---|---|---|---|
| $U_1$ | 23 | 35.8400 | 38.6300 | 42.4690 | 36.3400 | 34.1381 | 34.1380 | 34.1420 |
| $U_2$ | 52.5100 | 44.4700 | 38.9400 | 54.9640 | 46.5800 | 44.7553 | 44.7553 | 44.7414 |
| $U_3$ | 185.9700 | 189 | 178 | 69.7650 | 189 | 189 | 189 | 188.9999 |
| $U_4$ | 150.5600 | 138.4000 | 142.2000 | 73.7550 | 139.1600 | 138.2608 | 138.2608 | 138.2692 |
| $U_5$ | 10.2500 | 10.2500 | 13.4300 | 32.7880 | 11.0600 | 10.2500 | 10.2500 | 10.2500 |
| $U_6$ | 10.2500 | 10.2500 | 13.4200 | 37.7720 | 10.2500 | 10.2500 | 10.2500 | 10.2500 |
| $U_7$ | 23 | 23 | 29 | 23.0090 | 23 | 23 | 23 | 23 |
| $U_8$ | 23 | 31.6200 | 26.8400 | 93.5910 | 29.9000 | 31.8661 | 31.8661 | 31.8651 |
| $U_9$ | 23 | 23 | 29 | 23.0320 | 23 | 23 | 23 | 23 |
| $U_{10}$ | 114.4700 | 110.1700 | 106.5400 | 164.8540 | 107.7100 | 111.4796 | 111.4795 | 111.4823 |
| Total power generation | 616 | 616 | 616 | 616 | 616 | 616 | 616 | 616 |
| Cost ($/h) | 95,745.5400 | 95,633 | 95,840.5700 | 10,0207.1500 | 95,835.5300 | 95,632.1256 | 95,632.1256 | 95,632.1256 |

Abbreviations: dBA, directional bat algorithm; GA, genetic algorithm; MIW-PSO, modified inertial weight-based particle swarm optimization; TLBO, TLBO, teaching–learning-based optimization.

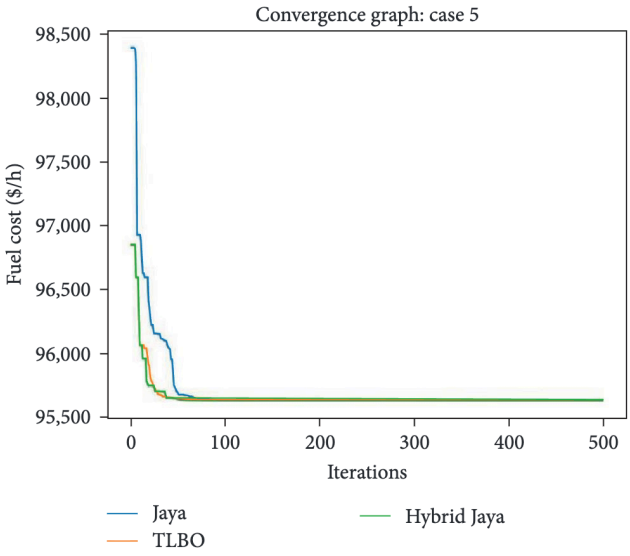


FIGURE 14: Best case convergence characteristics comparison between Jaya, TLBO, and hybrid Jaya for case 5. TLBO, teaching–learning-based optimization.

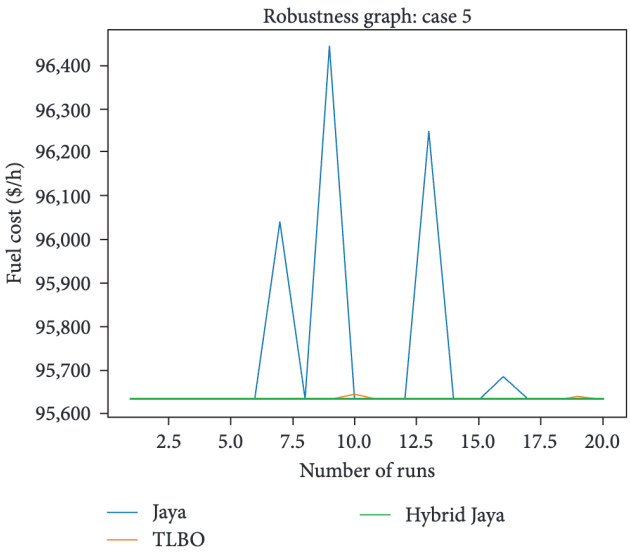


FIGURE 15: Robustness characteristics comparison between Jaya, TLBO, and hybrid Jaya for case 5. TLBO, teaching–learning-based optimization.

2. The observed difference is marginally significant when the $p$-value is less than 0.10.
3. The observed difference is null (the behavior is the same) when the $p$-value is 0.05.
4. The observed difference is significant when the $p$-value is less than 0.05.
5. The observed difference is very significant when the $p$-value is less than 0.01.

Though the main objective of this hybrid Jaya algorithm is to minimize the cost of generation, the simple computation time for each algorithm is presented in Table 14.

The computation time of the hybrid algorithm is awe-inspiring. The main objective of this paper was to study cost minimization, in which this algorithm performed exceptionally well compared to other well-known algorithms in the literature. However, limited literature mentions the computation or CPU time required to solve each test case. In our future work, we will investigate each algorithm's computation and CPU time by running them individually, as this data is only readily available for some comparative algorithms.

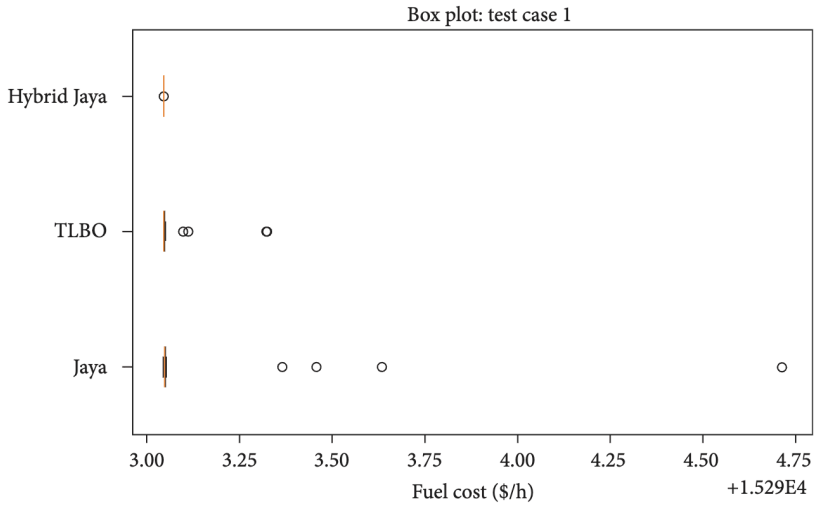


FIGURE 16: Box plot of Jaya, TLBO, and hybrid Jaya for test case 1. TLBO, teaching–learning-based optimization.

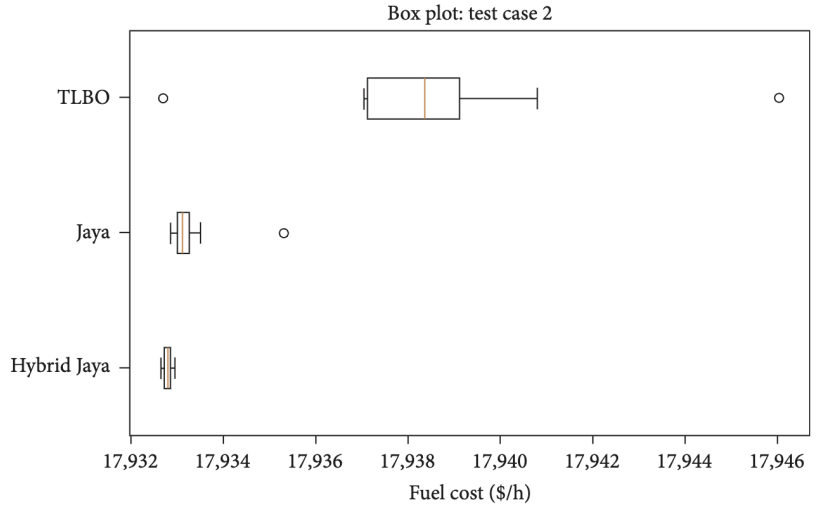


FIGURE 17: Box plot of Jaya, TLBO, and hybrid Jaya for test case 2. TLBO, teaching–learning-based optimization.

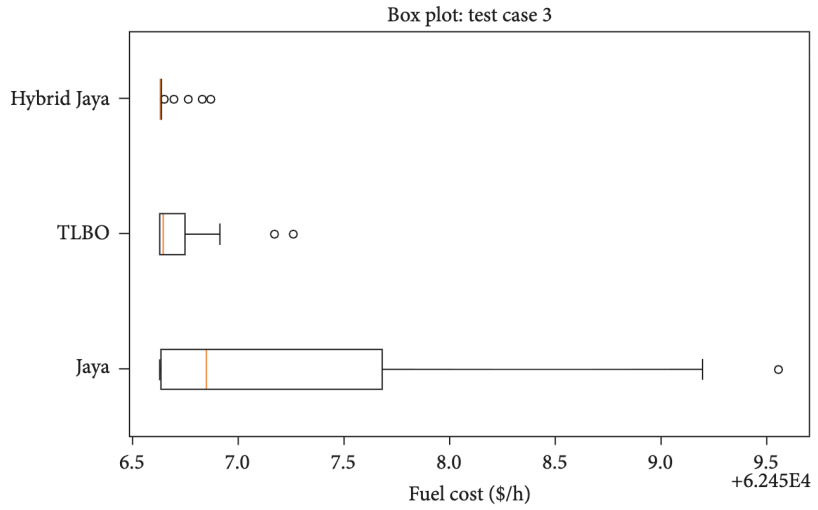


FIGURE 18: Box plot of Jaya, TLBO, and hybrid Jaya for test case 3. TLBO, teaching–learning-based optimization.

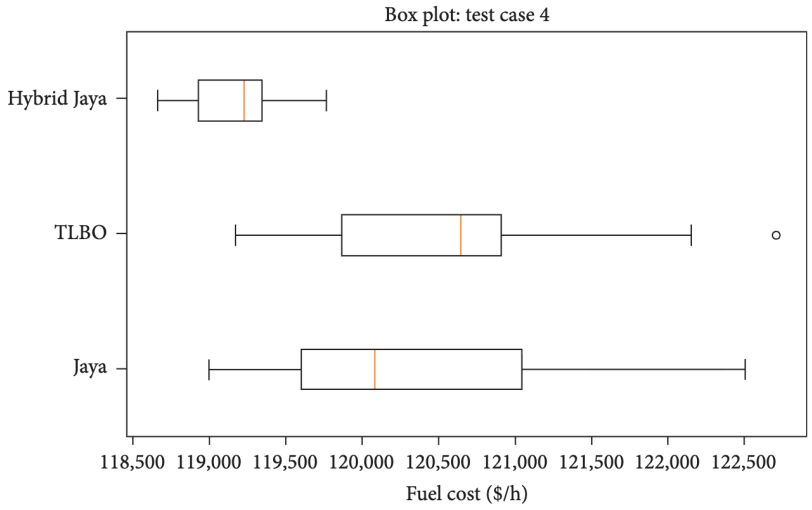


FIGURE 19: Box plot of Jaya, TLBO, and hybrid Jaya for test case 4. TLBO, teaching–learning-based optimization.

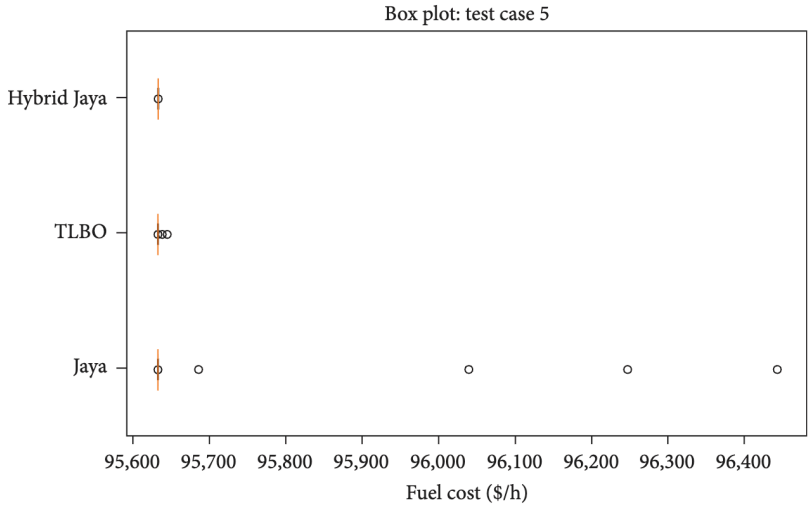


FIGURE 20: Box plot of Jaya, TLBO, and hybrid Jaya for test case 5. TLBO, teaching–learning-based optimization.

TABLE 12: *T*-test analysis of Jaya and hybrid Jaya for ELD problems.

| Serial | Test case | Degree of freedom | *p* value | Conclusion |
|---|---|---|---|---|
| 1 | 1 | 20 | 0.100 | Marginally significant |
| 2 | 2 | 20 | 3.2503E-64 | Highly significant |
| 3 | 3 | 20 | 3.0150E-05 | Highly significant |
| 4 | 4 | 20 | 6.3323E-05 | Highly significant |
| 5 | 5 | 20 | 0.0854 | Marginally significant |

Abbreviation: ELD, economic load dispatch.

TABLE 13: *T*-test analysis of TLBO and hybrid Jaya for ELD problems.

| Serial | Test case | Degree of freedom | *p* value | Conclusion |
|---|---|---|---|---|
| 1 | 1 | 20 | 0.0889 | Marginally significant |
| 2 | 2 | 20 | 0.0004 | Highly significant |
| 3 | 3 | 20 | 0.0676 | Highly significant |
| 4 | 4 | 20 | 0.0004 | Highly significant |
| 5 | 5 | 20 | 0.1000 | Marginally significant |

Abbreviations: ELD, economic load dispatch; TLBO, teaching–learning-based optimization.

Table 14: Computation time of hybrid Jaya algorithm.

| Serial | Test Case | Units | CPU meantime |
|---|---|---|---|
| 1 | 1 | 6 | 0.0630 |
| 2 | 2 | 13 | 0.7012 |
| 3 | 3 | 20 | 0.0935 |
| 4 | 4 | 40 | 103.9565 |
| 5 | 5 | 10 | 0.0652 |

Abbreviation: CPU, central processing unit.

## 6. Conclusion

This paper proposes an efficient hybrid Jaya optimization method. To enhance the functionality of the standard Jaya method, this version is based on the hybridization of Jaya and TLBO algorithms. The new hybrid algorithm outperformed the traditional Jaya in a set of a 6-unit nonconvex search space ELD problem that took into account power losses, power limits, ramp rate limits, and POZ for each unit, a 13-unit system with VPLE on the cost function and power generation boundaries, a 20-unit nonconvex search space ELD problem that took into account power losses, power limits, and in a 40-unit system using VPLE on the cost function and power generation limits, offering superior solutions than the original Jaya and TLBO employed in these four case studies. The simulation results demonstrate the proposed hybrid Jaya optimization algorithm's high accuracy and minimized cost, allowing it to be used in various real-world optimization situations.

However, the proposed method's performance is reliant on the ELD search space. The ELD problem has various variants, and the proposed hybrid approach is only intended for situations with specific constraints addressed in this work.

It is essential to delve into more challenging and significantly larger scale generation problems. Future research will focus on applying these algorithms to solve more demanding constrained optimization problems. Additionally, further exploration of integrating other metrics such as success rates [89] and adapting them with other parameter-less approaches will be conducted in future studies.

## Nomenclature

GA-API: Hybrid ant colony-genetic algorithm
GA: Genetic algorithm
PSO: Particle swarm optimization
MABC: Multispecies artificial bee colony
CBA: Classification based on association rules
NPSO-LRS: New PSO local random search
MSSA: Multivariate singular spectrum algorithm
DE: Differential evolution
ST-IRDPSO: Improved random drift particle swarm optimization algorithm
SOH-PSO: Self-organizing hierarchical particle swarm optimization
MCSA: Modified crow search algorithm
HGS: Hunger games search
MTS: Moving target search
HHS: Hybrid harmony search
ESSDO: Eagle-strategy supply–demand optimizer
NN-EPSO: Neural network enhanced particle swarm optimization
MTV-PSO: Modified time-varying particle swarm optimization
NR: Newton–Raphson
HM: Harmony search
EHNN: Enhanced hopfield neural network
BBO: Biography-based optimization
GSO: Glowworm swarm optimization
BSA: Backtracking spiral algorithm
HHO: Harris hawk optimizer
ABC: Artificial bee colony
HHO-A βHC: Harris hawk optimizer with adaptive beta hill climbing
ABOMDE: Accelerated biogeography-based optimization differential evolution
ARCGA: Adaptive real coded genetic algorithm
βHC: Beta hill climbing
RVMPSO: Real-valued mutation particle swarm optimization
CLCS-CLM: Comprehensive learning cuckoo search with chaos-lambda method
CSOMA: Cultural self-organizing migrating algorithm
FAPSO: Fuzzy adaptive particle swarm optimization
FCASO-SQP: Fuzzy adaptive chaotic ant swarm optimization- sequential quadratic programing
FAPSO-NM: Fuzzy adaptive particle swarm optimization with Nelder–Mead
FFA: Firefly algorithm
GWO: Gray wolf optimization
HGWO: Hybrid gray wolf optimizer
HCASO: Hybrid chaotic ant swarm optimization
HCPSO: Hybrid coding particle swarm optimization
HMAPSO: Hybrid multiagent particle swarm optimization
HQIPSO: Hybrid quantum inspired particle swarm optimization
HMAPSO: Hybrid multiagent particle swarm optimization
HSSA: Hybrid salp swarm algorithm
NDS: Novel direct search
NSABC: Nondominated sorting based artificial bee colony
NUHS: Natural updated harmony search
RCGA: Real coded genetic algorithm
SOMA: Self-organizing migrating algorithm
THS: Tournament-based harmony search
TSARGA: Taguchi self-adaptive real-coded genetic algorithm
TLA: Teaching–learning algorithm.

# Appendix A

Table A1: Data for the 6-unit system: generating unit capacity, coefficients, ramp rate limitations, and banned zones for producing units.

| Unit | $a_i$ ($) | $b_i$ ($/MW) | $c_i$ ($/MW$^2$) | $P_{i\max}$ (MW) | $P_i^0$ | $DR_i$ | Prohibited operating zones |
|---|---|---|---|---|---|---|---|
| 1 | 240 | 0.7 | 0.007 | 500 | 440 | 120 | [210, 240] [350, 380] |
| 2 | 200 | 10 | 0.0095 | 200 | 170 | 90 | [90, 110] [140,160) |
| 3 | 220 | 8.5 | 0.009 | 300 | 200 | 100 | I 150,170] [210. 240J |
| 4 | 200 | 11 | 0.009 | 150 | 150 | 90 | [80, 90] [110, 120] |
| 5 | 220 | 10.5 | 0.008 | 200 | 190 | 90 | [90, 110] [140, 150] |
| 6 | 190 | 12 | 0.0075 | 120 | 110 | 90 | [75, 85] [100,105] |

Abbreviation: down ramp.

# Appendix B

Table A2: The 6-unit test system's actual loss $B$-coefficients.

| $B_{oi}$ = 1.0E-3 | −0.3908 | −0.1297 | 0.7047 | 0.0591 | 0.2161 | −0.6635 |
|---|---|---|---|---|---|---|
| $B_{00}$ | 0.056 | — | — | — | — | — |
| $B_{ij}$ | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 0.0017 | 0.0012 | 0.0007 | −0.0001 | −0.0005 | −0.0002 |
| 2 | 0.0012 | 0.0011 | 0.0009 | 0.0001 | −0.0006 | −0.0001 |
| 3 | 0.0007 | 0.0009 | 0.0031 | 0 | −0.001 | −0.0006 |
| 4 | −0.0001 | 0.0001 | 0 | 0.0024 | −0.0006 | −0.0008 |
| 5 | −0.0005 | −0.0006 | −0.001 | −0.0006 | 0.0129 | −0.0006 |
| 6 | −0.0002 | −0.0001 | −0.0006 | −0.0008 | −0.0002 | 0.015 |

# Appendix C

Table A3: Data for the 13-unit system with the valve point effect: fuel cost coefficients and generation limits.

| Unit | $a_i$ ($) | $b_i$ ($/MW) | $c_i$ ($/MW$^2$) | $d_i$ ($) | $e_i$ (1/MW) | $P_{i\min}$ (MW) | $P_{i\max}$ (MW) |
|---|---|---|---|---|---|---|---|
| 1 | 0.00028 | 8.1 | 550 | 300 | 0.035 | 0 | 680 |
| 2 | 0.00056 | 8.1 | 309 | 200 | 0.042 | 0 | 360 |
| 3 | 0.00056 | 8.1 | 307 | 200 | 0.042 | 0 | 360 |
| 4 | 0.00324 | 7.74 | 240 | 150 | 0.063 | 60 | 180 |
| 5 | 0.00324 | 7.74 | 240 | 150 | 0.063 | 60 | 180 |
| 6 | 0.00324 | 7.74 | 240 | 150 | 0.063 | 60 | 180 |
| 7 | 0.00324 | 7.74 | 240 | 150 | 0.063 | 60 | 180 |
| 8 | 0.00324 | 7.74 | 240 | 150 | 0.063 | 60 | 180 |
| 9 | 0.00324 | 7.74 | 240 | 150 | 0.063 | 60 | 180 |
| 10 | 0.00284 | 8.6 | 126 | 100 | 0.084 | 40 | 120 |
| 11 | 0.00284 | 8.6 | 126 | 100 | 0.084 | 40 | 120 |
| 12 | 0.00284 | 8.6 | 126 | 100 | 0.084 | 55 | 120 |
| 13 | 0.00284 | 8.6 | 126 | 100 | 0.084 | 55 | 120 |

## Appendix D. Transmission Loss Coefficients for Case 2.

$$B = 10^{-5} \times \begin{bmatrix} 8.70 & 0.43 & -4.61 & 0.36 & 0.32 & -0.66 & 0.96 & -1.60 & 0.80 & -0.10 & 3.60 & 0.64 & 0.79 & 2.10 & 1.70 & 0.80 & -3.20 & 0.70 & 0.48 & -0.70 \\ 0.43 & 8.30 & -0.97 & 0.22 & 0.75 & -0.28 & 5.04 & 1.70 & 0.54 & 7.20 & -0.28 & 0.98 & -0.46 & 1.30 & 0.80 & -0.20 & 0.52 & -1.70 & 0.80 & 0.20 \\ -4.61 & -0.97 & 9.00 & -2.00 & 0.63 & 3.00 & 1.70 & -4.30 & 3.10 & -2.00 & 0.70 & -0.77 & 0.93 & 4.60 & -0.30 & 4.20 & 0.38 & 0.70 & -2.00 & 3.60 \\ 0.36 & 0.22 & -2.00 & 5.30 & 0.47 & 2.62 & -1.96 & 2.10 & 0.67 & 1.80 & -0.45 & 0.92 & 2.40 & 7.60 & -0.20 & 0.70 & -1.00 & 0.86 & 1.60 & 0.87 \\ 0.32 & 0.75 & 0.63 & 0.47 & 8.60 & -0.80 & 0.37 & 0.72 & -0.90 & 0.69 & 1.80 & 4.30 & -2.80 & -0.70 & 2.30 & 3.60 & 0.80 & 0.20 & -3.00 & 0.50 \\ -0.66 & -0.28 & 3.00 & 2.62 & -0.80 & 11.80 & -4.90 & 0.30 & 3 & -3 & 0.40 & 0.78 & 6.40 & 2.60 & -0.20 & 2.10 & -0.40 & 2.30 & 1.60 & -2.10 \\ 0.96 & 5.04 & 1.70 & -1.96 & 0.37 & -4.90 & 8.24 & -0.90 & 5.90 & -0.60 & 8.50 & -0.83 & 7.20 & 4.80 & -0.90 & -0.10 & 1.30 & 0.76 & 1.90 & 1.30 \\ -1.60 & 1.70 & -4.30 & 2.10 & 0.72 & 0.30 & -0.90 & 1.20 & -0.96 & 0.93 & -0.30 & 6.50 & 2.30 & 2.60 & 0.58 & -0.10 & 0.23 & -0.30 & 5.30 & 0.80 \\ 0.80 & 0.54 & 3.10 & 0.67 & -0.90 & 3.00 & 5.90 & -0.96 & 0.93 & -0.30 & 6.50 & 2.30 & 2.60 & 0.58 & -0.10 & 0.23 & -0.30 & 1.50 & 0.74 & 0.70 \\ -0.10 & 7.20 & -2.00 & 1.80 & 0.69 & -3.00 & -0.60 & 0.56 & -0.30 & 0.99 & -6.60 & 3.90 & 2.30 & -0.30 & 2.80 & -0.80 & 0.38 & 1.90 & 0.47 & -0.26 \\ 3.60 & -0.28 & 0.70 & -0.45 & 1.80 & 0.40 & 8.50 & 1.60 & 6.50 & -6.60 & 10.70 & 5.30 & -0.60 & 0.70 & 1.90 & -2.60 & 0.93 & -0.60 & 3.80 & -1.50 \\ 0.64 & 0.98 & -0.77 & 0.92 & 4.30 & 0.78 & -0.83 & 0.80 & 2.30 & 3.90 & 5.30 & 8.00 & 0.90 & 2.10 & -0.70 & 5.70 & 5.40 & 1.50 & 0.70 & 0.10 \\ 0.79 & -0.46 & 0.93 & 2.40 & -2.80 & 6.40 & 7.20 & -0.40 & 2.60 & 2.30 & -0.60 & 0.90 & 11.00 & 0.87 & -1.00 & 3.60 & 0.46 & -0.90 & 0.60 & 1.50 \\ 2.10 & 1.30 & 4.60 & 7.60 & -0.70 & 2.60 & 4.80 & 0.23 & 0.58 & -0.30 & 0.70 & 2.10 & 0.87 & 3.80 & 0.50 & -0.70 & 1.90 & 2.30 & -0.97 & 0.90 \\ 1.70 & 0.80 & -0.30 & -0.20 & 2.30 & -0.20 & -0.90 & 0.75 & -0.10 & 2.80 & 1.90 & -0.70 & -1.00 & 0.50 & 11.00 & 1.90 & -0.80 & 2.60 & 2.30 & -0.10 \\ 0.80 & -0.20 & 4.20 & 0.70 & 3.60 & 2.10 & -0.10 & -0.56 & 0.23 & -0.80 & -2.60 & 5.70 & 3.60 & -0.70 & 1.90 & 10.80 & 2.50 & -1.80 & 0.90 & -2.60 \\ -3.20 & 0.52 & 0.38 & -1.00 & 0.80 & -0.40 & 1.20 & 0.80 & -0.20 & 0.38 & 0.93 & 5.40 & 0.46 & 1.90 & -0.80 & 2.50 & 8.70 & 4.20 & -0.30 & 0.68 \\ 0.70 & -1.70 & 0.70 & 0.86 & 0.20 & 2.30 & 0.76 & -0.30 & 1.50 & 1.90 & -0.60 & 1.50 & -0.90 & 2.30 & 2.60 & -1.80 & 4.20 & 2.20 & 0.16 & -0.30 \\ 0.48 & 0.80 & -2.00 & 1.60 & -3.00 & 1.60 & 1.90 & 5.30 & 0.74 & 0.47 & 3.80 & 0.70 & 0.60 & -0.97 & 2.30 & 0.90 & -0.30 & 0.16 & 7.60 & 0.69 \\ -0.70 & 0.20 & 3.60 & 0.87 & 0.50 & 2.10 & 1.30 & 0.80 & 0.70 & -0.26 & -1.50 & 0.10 & 1.50 & 0.90 & -0.10 & -2.60 & 0.68 & -0.30 & 0.69 & 7.00 \end{bmatrix}$$

## Appendix E

Table A4: Data for the 20-unit system with transmission loss and generation limits.

| Generating unit | $a$ | $b$ | $c$ | $P_{\min}$(MW) | $P_{\max}$(MW) |
|---|---|---|---|---|---|
| $U_1$ | 0.00068 | 18.19 | 1000 | 150 | 600 |
| $U_2$ | 0.00071 | 19.26 | 970 | 50 | 200 |
| $U_3$ | 0.0065 | 19.8 | 600 | 50 | 200 |
| $U_4$ | 0.005 | 19.1 | 700 | 50 | 200 |
| $U_5$ | 0.00738 | 18.1 | 420 | 50 | 160 |
| $U_6$ | 0.00612 | 19.26 | 360 | 20 | 100 |
| $U_7$ | 0.0079 | 17.14 | 490 | 25 | 125 |
| $U_8$ | 0.00813 | 18.92 | 660 | 50 | 150 |
| $U_9$ | 0.00522 | 18.27 | 765 | 50 | 200 |
| $U_{10}$ | 0.00573 | 18.92 | 770 | 30 | 150 |
| $U_{11}$ | 0.0048 | 16.69 | 800 | 100 | 300 |
| $U_{12}$ | 0.0031 | 16.76 | 970 | 150 | 500 |
| $U_{13}$ | 0.0085 | 17.36 | 900 | 40 | 160 |
| $U_{14}$ | 0.00511 | 18.7 | 700 | 20 | 130 |
| $U_{15}$ | 0.00398 | 18.7 | 450 | 25 | 185 |
| $U_{16}$ | 0.0712 | 14.26 | 370 | 20 | 80 |
| $U_{17}$ | 0.0089 | 19.14 | 480 | 30 | 85 |
| $U_{18}$ | 0.00713 | 18.92 | 680 | 30 | 120 |
| $U_{19}$ | 0.00622 | 18.47 | 700 | 40 | 120 |
| $U_{20}$ | 0.00773 | 19.79 | 850 | 30 | 100 |

## Appendix F

Table A5: Data for the 40-unit system with the valve-point effect: fuel cost coefficients and generation limits.

| Unit | $a_i$ ($) | $b_i$ ($/MW) | $c_i$ ($/MW$^2$) | $P_{i\min}$ (MW) | $P_{i\max}$ (MW) | $e_i$ (1/MW) | $f_i$ ($) |
|---|---|---|---|---|---|---|---|
| 1 | 0.0069 | 6.73 | 94.705 | 36 | 114 | 0.084 | 100 |
| 2 | 0.0069 | 6.73 | 94.705 | 36 | 114 | 0.084 | 100 |
| 3 | 0.02028 | 7.07 | 309.54 | 60 | 120 | 0.084 | 100 |
| 4 | 0.00942 | 8.18 | 369.03 | 80 | 190 | 0.063 | 150 |
| 5 | 0.0114 | 5.35 | 148.89 | 47 | 97 | 0.077 | 120 |
| 6 | 0.01142 | 8.05 | 222.33 | 68 | 140 | 0.084 | 100 |
| 7 | 0.00357 | 8.03 | 287.71 | 110 | 300 | 0.042 | 200 |
| 8 | 0.00492 | 6.99 | 391.98 | 135 | 300 | 0.042 | 200 |
| 9 | 0.00573 | 6.6 | 455.76 | 135 | 300 | 0.042 | 200 |
| 10 | 0.00605 | 12.9 | 722.82 | 130 | 300 | 0.042 | 200 |
| 11 | 0.00515 | 12.9 | 635.2 | 94 | 375 | 0.042 | 200 |
| 12 | 0.00569 | 12.8 | 654.69 | 94 | 375 | 0.042 | 200 |
| 13 | 0.00421 | 12.5 | 913.4 | 125 | 500 | 0.035 | 300 |
| 14 | 0.00752 | 8.84 | 1760.4 | 125 | 500 | 0.035 | 300 |
| 15 | 0.00708 | 9.15 | 1728.3 | 125 | 500 | 0.035 | 300 |
| 16 | 0.00708 | 9.15 | 1728.3 | 125 | 500 | 0.035 | 300 |
| 17 | 0.00313 | 7.97 | 647.85 | 220 | 500 | 0.035 | 300 |
| 18 | 0.00313 | 7.95 | 649.69 | 220 | 500 | 0.035 | 300 |
| 19 | 0.00313 | 7.97 | 647.83 | 242 | 550 | 0.035 | 300 |
| 20 | 0.00313 | 7.97 | 647.81 | 242 | 550 | 0.035 | 300 |
| 21 | 0.00298 | 6.63 | 785.96 | 254 | 550 | 0.035 | 300 |
| 22 | 0.00298 | 6.63 | 785.96 | 254 | 550 | 0.035 | 300 |
| 23 | 0.00284 | 6.66 | 794.53 | 254 | 550 | 0.035 | 300 |
| 24 | 0.00284 | 6.66 | 794.53 | 254 | 550 | 0.035 | 300 |
| 25 | 0.00277 | 7.1 | 801.32 | 254 | 550 | 0.035 | 300 |
| 26 | 0.00277 | 7.1 | 801.32 | 254 | 550 | 0.035 | 300 |
| 27 | 0.52124 | 3.33 | 1055.1 | 10 | 150 | 0.077 | 120 |
| 28 | 0.52124 | 3.33 | 1055.1 | 10 | 150 | 0.077 | 120 |
| 29 | 0.52124 | 3.33 | 1055.1 | 10 | 150 | 0.077 | 120 |
| 30 | 0.0114 | 5.35 | 148.89 | 47 | 97 | 0.077 | 120 |
| 31 | 0.0016 | 6.43 | 222.92 | 60 | 190 | 0.063 | 150 |
| 32 | 0.0016 | 6.43 | 222.92 | 60 | 190 | 0.063 | 150 |
| 33 | 0.0016 | 6.43 | 222.92 | 60 | 190 | 0.063 | 150 |
| 34 | 0.0001 | 8.95 | 107.87 | 90 | 200 | 0.042 | 200 |
| 35 | 0.0001 | 8.62 | 116.58 | 90 | 200 | 0.042 | 200 |
| 36 | 0.0001 | 8.62 | 116.58 | 90 | 200 | 0.042 | 200 |
| 37 | 0.0161 | 5.88 | 307.45 | 25 | 110 | 0.098 | 80 |
| 38 | 0.0161 | 5.88 | 307.45 | 25 | 110 | 0.098 | 80 |
| 39 | 0.0161 | 5.88 | 307.45 | 25 | 110 | 0.098 | 80 |
| 40 | 0.00313 | 7.97 | 647.83 | 242 | 550 | 0.035 | 300 |

## Appendix G

Table A6: Generator unit capacity and cost coefficient.

| Unit | $a_i$ ($) | $b_i$ ($/MW) | $c_i$ ($/MW$^2$) | $P_{i\min}$ (MW) | $P_{i\max}$ (MW) |
|---|---|---|---|---|---|
| 1 | 0.2162 | 42.5118 | 4088.5375 | 23.00 | 92 |
| 2 | 0.4108 | 20.5021 | 4547.8075 | 23.00 | 92 |
| 3 | 0.05620 | 32.9483 | 4601.9649 | 47.25 | 189 |
| 4 | 0.1266 | 22.2655 | 4316.1074 | 47.25 | 189 |
| 5 | 0.6210 | 50.6244 | 3707.7500 | 10.25 | 41 |
| 6 | 0.1255 | 69.7050 | 3459.6950 | 10.25 | 41 |
| 7 | 3.6454 | 370.6642 | 9045.7750 | 23.00 | 95 |
| 8 | 0.3981 | 31.9013 | 1124.9075 | 23.00 | 95 |
| 9 | 2.3185 | 484.7006 | 8549.5500 | 23.00 | 95 |
| 10 | 0.1142 | 31.8112 | 4486.6174 | 41.25 | 165 |

## Appendix H. Pseudocode for the Proposed Hybrid Jaya Algorithm.

**Start**
**Initialize variables**
Generate initial population randomly;
Evaluate the objective function for each member of the population;
Finding the best (**fmin**) and worst members of the population and its position;
**Hybrid main function loop**
**for** (i: = 1 to maximum_iteration_number)
  **for** (j: = 1 to number_of _populaton_size)
    **Perform TLBO algorithm**
      **Teacher phase**
      Find the mean and minimum value and its position
      Identify the best solution as teacher, Xnew
      Checks the bounds generates by the Xew
      fnew = cost_function (Xnew)
      **if** fnew <fx (i,:)
        pos (i,:) = Xnew;
        fx (i,:) = fnew;
      **End if**
      **Learner Phase**
      Pa = 1: population_size;
      Pa (i) = [];
      Partner = Pa (randi (length (Pa));
      X*p* = Population (Partner,:);
      F*p* = Cost_function (Xp);
      X = Population (I,:);
      **if** fx (I,:)<fp
        Xnew = X + random_value. * (X-Xp);
      **else** Xnew = X-random_value. * (X-Xp);
      **End else**
      **End if**
      **Perform greedy solution**
      Evaluate the fitness with cost function for the new position Xnew;
      **if** the new fitness fnew is better than the current fitness fx (i,:);
        Update the current position population (i,:) with the new position Xnew;
        Update the current fitness fx (i,:) with the new fitness fnew;
      **End if**
      **Find TLBO algorithm optimum value**
      **Perform JAYA algorithm**
      Get the current position of a population i as X;
      Generate a new position Xnew using the equation (Xbest-abs (X))- random_value.* (Xworst abs (X));
      Keep Xnew within the lower and upper bounds
      Evaluate the fitness value with cost function of Xnew
      **Perform Greedy selection for JAYA algorithm**
      If the fitness value of Xnew is better than the current fitness value fx (i), update the particle's position to Xnew and the fitness value to fnew;
      **Find JAYA algorithm optimum value**
      **if** the value of JAYA_optimal_value is less than both TLBO_optimal_value and fmin, then update fmin to JAYA_optimal_value
      **else if** the value of TLBO_optimal_value is less than both JAYA_optimal_value and fmin, then update fmin to TLBO_optimal_value
      **else** keep the value of fmin unchanged
      **End else**
    **End else**
  **End if**

52. **End for**
53. **Show optimal solution for each iteration**
54. **End for**

## Data Availability Statement

Data are publicly available.

## Conflicts of Interest

The authors declare no conflicts of interest.

## Author Contributions

**Khairul Eahsun Fahim:** conceptualization, formal analysis, investigation, visualization, writing–original draft, programing. **Liyanage C. De Silva:** conceptualization, methodology, formal analysis, resources, project administration, supervision, validation. **Sk. A. Shezan:** methodology, investigation, software and editing. **Hayati Yassin:** software, project administration, investigation, formal analysis, validation, review and editing. **Viknesh Andiappan:** formal analysis, validation, review and editing.

## Funding

The authors extend their appreciation to the UGS scholarship at Universiti Brunei Darussalam and Swinburne University of Technology, Malaysia, for funding this work.

## Acknowledgments

The authors express their sincere gratitude to the Faculty of Integrated Technologies, Universiti Brunei Darussalam, for their thorough research support.

## References

[1] M. A. Al-Betar, M. A. Awadallah, S. N. Makhadmeh, et al., "A Hybrid Harris Hawks Optimizer for Economic Load Dispatch Problems," *Alexandria Engineering Journal* 64 (2023): 365–389.

[2] S. Pan, J. Jian, H. Chen, and L. Yang, "A Full Mixed-Integer Linear Programming Formulation for Economic Dispatch With Valve-Point Effects, Transmission Loss and Prohibited Operating Zones," *Electric Power Systems Research* 180 (2020): 106061.

[3] M. H. Mansor, I. Musirin, M. M. Othman, M. K. M. Zamani, S. A. Shaaya, and S. Jelani, "Immune-Commensal-Evolutionary Programming for Solving Non-Smooth/Non-Convex Economic Dispatch Problem," *Energy Reports* 6 (2020): 266–275.

[4] V. Veerasamy, N. I. A. Wahab, R. Ramachandran, et al., "A novel RK4-Hopfield Neural Network for Power Flow Analysis of power system," *Applied Soft Computing* 93 (2020): 106346.

[5] M. Hanif and N. Mohammad, "Artificial Bee Colony and Genetic Algorithm for Optimization of Non-smooth Economic Load Dispatch With Transmission Loss," in *Proceedings of the International Conference on Big Data, IoT, and Machine Learning*, vol. 95 of *Lecture Notes on Data Engineering and Communications Technologies*, (Springer, Singapore, 2022): 271–287.

[6] M. Gholamghasemi, E. Akbari, M. B. Asadpoor, and M. Ghasemi, "A New Solution to the Non-Convex Economic Load Dispatch Problems Using Phasor Particle Swarm Optimization," *Applied Soft Computing* 79 (2019): 111–124.

[7] V. Garg, K. Deep, and N. P. Padhee, "Constrained Laplacian Biogeography-Based Optimization for Economic Load Dispatch Problems," *Process Integration and Optimization for Sustainability* 6, no. 2 (2022): 483–496.

[8] R. Rao, "Review of Applications of TLBO Algorithm and a Tutorial for Beginners to Solve the Unconstrained and Constrained Optimization Problems," *Decision Science Letters* 5, no. 1 (2016): 1–30.

[9] J. K. Pattanaik, M. Basu, and D. P. Dash, "Dynamic Economic Dispatch: A Comparative Study for Differential Evolution, Particle Swarm Optimization, Evolutionary Programming, Genetic Algorithm, and Simulated Annealing," *Journal of Electrical Systems and Information Technology* 6, no. 1 (2019): 1–18.

[10] M. A. Al-Betar, M. A. Awadallah, and M. M. Krishan, "A Non-Convex Economic Load Dispatch Problem With Valve Loading Effect Using a Hybrid Grey Wolf Optimizer," *Neural Computing and Applications* 32, no. 16 (2020): 12127–12154.

[11] F. Z. Alazemi and A. Y. Hatata, "Ant Lion Optimizer for Optimum Economic Dispatch Considering Demand Response as a Visual Power Plant," *Electric Power Components and Systems* 47, no. 6-7 (2019): 629–643.

[12] R. Ramalingam, D. Karunanidy, S. S. Alshamrani, M. Rashid, S. Mathumohan, and A. Dumka, "Oppositional Pigeon-Inspired Optimizer for Solving the Non-Convex Economic Load Dispatch Problem in Power Systems," *Mathematics* 10, no. 18 (2022): 3315.

[13] T. Ahmed, M. Ebeed, A. Refai, and S. Kamel, "Solving Combined Economic and Emission Dispatch Problem Using the Slime Mould Algorithm," *Sohag Engineering Journal* 1, no. 1 (2021): 62–70.

[14] S. H. A. Kaboli and A. K. Alqallaf, "Solving Non-Convex Economic Load Dispatch Problem via Artificial Cooperative Search Algorithm," *Expert Systems With Applications* 128 (2019): 14–27.

[15] M. Ellahi, G. Abbas, G. B. Satrya, M. R. Usman, and J. Gu, "A Modified Hybrid Particle Swarm Optimization With Bat Algorithm Parameter Inspired Acceleration Coefficients for Solving Eco-Friendly and Economic Dispatch Problems," *IEEE Access* 9 (2021): 82169–82187.

[16] V. Kansal and J. S. Dhillon, "Application of Hybrid Artificial Algae Algorithm for Dynamic Economic Load Dispatch Problem," in *2021 IEEE 2nd International Conference On Electrical Power and Energy Systems (ICEPES)*, (IEEE, 2021): 1–6.

[17] D. Singh and J. S. Dhillon, "Ameliorated Grey Wolf Optimization for Economic Load Dispatch Problem," *Energy* 169 (2019): 398–419.

[18] H. T. Kahraman, S. Aras, and E. Gedikli, "Fitness-Distance Balance (FDB): A New Selection Method for Meta-Heuristic Search Algorithms," *Knowledge-Based Systems* 190 (2020): 105169.

[19] H. T. Kahraman, M. Katı, S. Aras, and D. A. Taşci, "Development of the Natural Survivor Method (NSM) for Designing an Updating Mechanism in Metaheuristic Search Algorithms," *Engineering Applications of Artificial Intelligence* 122 (2023): 106121.

[20] B. Ozkaya, H. T. Kahraman, S. Duman, and U. Guvenc, "Fitness-Distance-Constraint (FDC) Based Guide Selection

Method for Constrained Optimization Problems," *Applied Soft Computing* 144 (2023): 110479.

[21] H. T. Kahraman, H. Bakir, S. Duman, M. Katı, S. Aras, and U. Guvenc, "Dynamic FDB Selection Method and Its Application: Modeling and Optimizing of Directional Overcurrent Relays Coordination," *Applied Intelligence* 52 (2022): 1–36.

[22] H. T. Kahraman, M. H. Hassan, M. Katı, M. Tostado-Véliz, S. Duman, and S. Kamel, "Dynamic-Fitness-Distance-Balance Stochastic Fractal Search (dFDB-SFS Algorithm): An Effective Metaheuristic for Global Optimization and Accurate Photovoltaic Modeling," *Soft Computing* 28 (2023): 1–28.

[23] H. T. Kahraman, M. Akbel, S. Duman, M. Kati, and H. H. J. S. Sayan, "Unified Space Approach-Based Dynamic Switched Crowding (DSC): A New Method for Designing Pareto-Based Multi/Many-Objective Algorithms," *Swarm and Evolutionary Computation* 75 (2022): 101196.

[24] B. Ozkaya, H. T. Kahraman, S. Duman, U. Guvenc, and M. Akbel, "Combined Heat and Power Economic Emission Dispatch Using Dynamic Switched Crowding Based Multi-Objective Symbiotic Organism Search Algorithm," *Applied Soft Computing* 151 (2024): 111106.

[25] B. Ozkaya, S. Duman, H. T. Kahraman, and U. Guvenc, "Optimal Solution of the Combined Heat and Power Economic Dispatch Problem by Adaptive Fitness-Distance Balance Based Artificial Rabbits Optimization Algorithm," *Expert Systems With Applications* 238 (2024): 122272.

[26] H. Bakır, S. Duman, U. Guvenc, and H. T. Kahraman, "Improved Adaptive Gaining-Sharing Knowledge Algorithm With FDB-Based Guiding Mechanism for Optimization of Optimal Reactive Power Flow Problem," *Electrical Engineering* 105, no. 5 (2023): 3121–3160.

[27] Y. Sonmez, S. Duman, H. T. Kahraman, M. Kati, S. Aras, and U. Guvenc, "Fitness-Distance Balance Based Artificial Ecosystem Optimisation to Solve Transient Stability Constrained Optimal Power Flow Problem," *Journal of Experimental & Theoretical Artificial Intelligence* 36, no. 5 (2022): 745–784.

[28] H. Bakir, H. T. Kahraman, S. Temel, S. Duman, U. Guvenc, and Y. Sonmez, "Development of An FDB-Based Chimp Optimization Algorithm for Global Optimization and Determination of the Power System Stabilizer Parameters," in *Smart Applications With Advanced Machine Learning and Human-Centred Problem Design*, vol. 1 of *Engineering Cyber-Physical Systems and Critical Infrastructures*, (Springer, Cham, 2021): 337–365.

[29] S. Duman, H. T. Kahraman, B. Korkmaz, H. Bakir, U. Guvenc, and C. Yilmaz, "Improved Phasor Particle Swarm Optimization With Fitness Distance Balance for Optimal Power Flow Problem of Hybrid AC/DC Power Grids," in *Smart Applications With Advanced Machine Learning and Human-Centred Problem Design*, vol. 1 of *Engineering Cyber-Physical Systems and Critical Infrastructures*, (Springer, Cham, 2021): 307–336.

[30] D. A. Taşci, H. T. Kahraman, M. Kati, and C. Yilmaz, "Improved Gradient-Based Optimizer With Dynamic Fitness Distance Balance for Global Optimization Problems," in *Smart Applications With Advanced Machine Learning and Human-Centred Problem Design*, vol. 1 of *Engineering Cyber-Physical Systems and Critical Infrastructures*, (Springer, Cham, 2021): 247–269.

[31] B. Yenipinar, A. Şahin, Y. Sönmez, C. Yilmaz, and H. T. Kahraman, "Design Optimization of Induction Motor With FDB-Based Archimedes Optimization Algorithm for High Power Fan and Pump Applications," in *Smart Applications With Advanced Machine Learning and Human-Centred Problem Design*, Engineering Cyber-Physical Systems and Critical Infrastructures, (Springer, Cham, 2021): 409–428.

[32] R. V. Rao, *Jaya: An Advanced Optimization Algorithm and its Engineering Applications* (Springer, 2019).

[33] K. E. Fahim, H. Yassin, L. C. De Silva, T. Roy, N. M. Rihan, and M. A. Tanvir, "Jaya Algorithm-A Practical Algorithm for Solving Economic Load Dispatch Problems," in *2022 International Conference on Energy and Power Engineering (ICEPE)*, (IEEE, 2022): 1–5.

[34] R. Xue and Z. Wu, "A Survey of Application and Classification on Teaching-Learning-Based Optimization Algorithm," *IEEE Access* 8 (2020): 1062–1079.

[35] M. H. Hassan, S. Kamel, A. Eid, L. Nasrat, F. Jurado, and M. F. Elnaggar, "A Developed Eagle-Strategy Supply-Demand Optimizer for Solving Economic Load Dispatch Problems," *Ain Shams Engineering Journal* 14, no. 5 (2023): 102083.

[36] W. Aribowo, S. Muslim, and B. Suprianto, "A Hunger Game Search Algorithm for Economic Load Dispatch," *IAES International Journal of Artificial Intelligence (IJ-AI)* 11, no. 2 (2022): 632–640.

[37] M. S. Alanazi, "A Modified Teaching—Learning-Based Optimization for Dynamic Economic Load Dispatch Considering Both Wind Power and Load Demand Uncertainties With Operational Constraints," *IEEE Access* 9 (2021): 101665–101680.

[38] S. E. Edagbami, C. Yinka-Banjo, and C. O. Uwadia, "Adaptive Rooted Tree Optimization for Non-Convex Economic Dispatch Cost Function," in *2021 IEEE PES/IAS PowerAfrica*, (IEEE, 2021): 1–5.

[39] A. Kaur, L. Singh, and J. S. Dhillon, "Modified Krill Herd Algorithm for Constrained Economic Load Dispatch Problem," *International Journal of Ambient Energy* 43, no. 1 (2022): 4332–4342.

[40] M. K. Sattar, A. Ahmad, S. Fayyaz, S. S. Ul Haq, and M. S. Saddique, "Ramp rate handling strategies in Dynamic Economic Load Dispatch (DELD) problem using Grey Wolf Optimizer (GWO)," *Journal of the Chinese Institute of Engineers* 43, no. 2 (2020): 200–213.

[41] G. Dhiman, "MOSHEPO: A Hybrid Multi-Objective Approach to Solve Economic Load Dispatch and Micro Grid Problems," *Applied Intelligence* 50, no. 1 (2020): 119–137.

[42] K. Sharma, H. M. Dubey, and M. Pandit, "Teaching-Learning-Based Optimization for Static and Dynamic Load Dispatch," in *Nature Inspired Optimization for Electrical Power*, (Springer, System, 2020): 1–12.

[43] Z.-L. Gaing, "Particle Swarm Optimization to Solving the Economic Dispatch considering the Generator Constraints," *IEEE Transactions on Power Systems* 18, no. 3 (2003): 1187–1195.

[44] D. C. Secui, "A New Modified Artificial Bee Colony Algorithm for the Economic Dispatch Problem," *Energy Conversion and Management* 89 (2015): 43–62.

[45] B. R. Adarsh, T. Raghunathan, T. Jayabarathi, and X.-S. Yang, "Economic Dispatch Using Chaotic Bat Algorithm," *Energy* 96 (2016): 666–675.

[46] A. I. Selvakumar and K. Thanushkodi, "A New Particle Swarm Optimization Solution to Nonconvex Economic Dispatch Problems," *IEEE Transactions on Power Systems* 22, no. 1 (2007): 42–51.

[47] W. T. Elsayed, Y. G. Hegazy, F. M. Bendary, and M. S. El-bages, "Modified Social Spider Algorithm for Solving the Economic Dispatch Problem," *Engineering Science and Technology, An International Journal* 19, no. 4 (2016): 1672–1681.

[48] N. Noman and H. Iba, "Differential Evolution for Economic Load Dispatch Problems," *Electric Power Systems Research* 78, no. 8 (2008): 1322–1331.

[49] W. T. Elsayed, Y. G. Hegazy, M. S. El-bages, and F. M. Bendary, "Improved Random Drift Particle Swarm Optimization With Self-Adaptive Mechanism for Solving the Power Economic Dispatch Problem," *IEEE Transactions on Industrial Informatics* 13, no. 3 (2017): 1017–1026.

[50] K. T. Chaturvedi, M. Pandit, and L. Srivastava, "Self-Organizing Hierarchical Particle Swarm Optimization for Nonconvex Economic Dispatch," *IEEE Transactions on Power Systems* 23, no. 3 (2008): 1079–1087.

[51] F. Mohammadi and H. Abdi, "A Modified Crow Search Algorithm (MCSA) for Solving Economic Load Dispatch Problem," *Applied Soft Computing* 71 (2018): 51–65.

[52] S. Pothiya, I. Ngamroo, and W. Kongprawechnon, "Application of Multiple Tabu Search Algorithm to Solve Dynamic Economic Dispatch considering Generator Constraints," *Energy Conversion and Management* 49, no. 4 (2008): 506–516.

[53] M. Fesanghary and M. M. Ardehali, "A Novel Meta-Heuristic Optimization Methodology for Solving Various Types of Economic Dispatch Problem," *Energy* 34, no. 6 (2009): 757–766.

[54] R. Manam, R. Sangu, L. Pamidi, and M. K. R. Karri, "RA 123s: Three Metaphor-Less Algorithms for Economic Load Dispatch Solution," *Journal of Electrical Engineering & Technology* 17, no. 2 (2022): 835–845.

[55] K. Thanushkodi, "An Evolutionary Programming Based Efficient Particle Swarm Optimization for Economic Dispatch Problem With Valve Point Loading," (2011).

[56] V. Hosseinnezhad and E. Babaei, "Economic Load Dispatch Using $\theta$-PSO," *International Journal of Electrical Power & Energy Systems* 49 (2013): 160–169.

[57] R. P. Parouha, "Nonconvex/Nonsmooth Economic Load Dispatch Using Modified Time-Varying Particle Swarm Optimization," *Computational Intelligence* 35, no. 4 (2019): 717–744.

[58] C.-T. Su and C.-T. Lin, "New Approach With a Hopfield Modeling Framework to Economic Dispatch," *IEEE Transactions on Power Systems* 15, no. 2 (2000): 541–545.

[59] A. Y. Abdelaziz, S. F. Mekhamer, M. A. L. Badr, and M. Z. Kamh, "Economic Dispatch Using An Enhanced Hopfield Neural Network," *Electric Power Components and Systems* 36, no. 7 (2008): 719–732.

[60] A. Bhattacharya and P. K. Chattopadhyay, "Solving Complex Economic Load Dispatch Problems Using Biogeography-Based Optimization," *Expert Systems With Applications* 37, no. 5 (2010): 3605–3615.

[61] M. Moradi-Dalvand, B. Mohammadi-Ivatloo, A. Najafi, and A. Rabiee, "Continuous Quick Group Search Optimizer for Solving Non-Convex Economic Dispatch Problems," *Electric Power Systems Research* 93 (2012): 93–105.

[62] M. Modiri-Delshad and N. Abd Rahim, "Solving Non-Convex Economic Dispatch Problem via Backtracking Search Algorithm," *Energy* 77 (2014): 372–381.

[63] S. Hemamalini and S. P. Simon, "Artificial Bee Colony Algorithm for Economic Load Dispatch Problem With Non-Smooth Cost Functions," *Electric Power Components and Systems* 38, no. 7 (2010): 786–803.

[64] M. R. Lohokare, B. K. Panigrahi, S. S. Pattnaik, S. Devi, and A. Mohapatra, "Neighborhood Search-Driven Accelerated Biogeography-Based Optimization for Optimal Load Dispatch," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42, no. 5 (2012): 641–652.

[65] S. Sayah and A. Hamouda, "A Hybrid Differential Evolution Algorithm Based on Particle Swarm Optimization for Nonconvex Economic Dispatch Problems," *Applied Soft Computing* 13, no. 4 (2013): 1608–1619.

[66] M. A. Al-Betar, M. A. Awadallah, I. Abu Doush, E. Alsukhni, and H. ALkhraisat, "A Non-Convex Economic Dispatch Problem With Valve Loading Effect Using a New Modified $\beta$-Hill Climbing Local Search Algorithm," *Arabian Journal for Science and Engineering* 43, no. 12 (2018): 7439–7456.

[67] H. Lu, P. Sriyanyong, Y. H. Song, and T. Dillon, "Experimental Study of a New Hybrid PSO With Mutation for Economic Dispatch With Non-Smooth Cost Function," *International Journal of Electrical Power & Energy Systems* 32, no. 9 (2010): 921–935.

[68] Z. Huang, J. Zhao, L. Qi, Z. Gao, and H. Duan, "Comprehensive Learning Cuckoo Search With Chaos-Lambda Method for Solving Economic Dispatch Problems," *Applied Intelligence* 50, no. 9 (2020): 2779–2799.

[69] L. dos Santos Coelho and V. C. Mariani, "An Efficient Cultural Self-Organizing Migrating Strategy for Economic Dispatch Optimization With Valve-Point Effect," *Energy Conversion and Management* 51, no. 12 (2010): 2580–2587.

[70] T. Niknam, H. D. Mojarrad, H. Z. Meymand, and B. B. Firouzi, "A New Honey Bee Mating Optimization Algorithm for Non-Smooth Economic Dispatch," *Energy* 36, no. 2 (2011): 896–908.

[71] J. Cai, Q. Li, L. Li, H. Peng, and Y. Yang, "A Hybrid FCASO-SQP Method for Solving the Economic Dispatch Problems With Valve-Point Effects," *Energy* 38, no. 1 (2012): 346–353.

[72] N. Sinha, R. Chakrabarti, and P. K. Chattopadhyay, "Evolutionary Programming Techniques for Economic Load Dispatch," *IEEE Transactions on Evolutionary Computation* 7, no. 1 (2003): 83–94.

[73] M. H. Sulaiman, W. L. Ing, Z. Mustaffa, and M. R. Mohamed, "Grey Wolf Optimizer for Solving Economic Dispatch Problem With Valve-Loading Effects," *ARPN Journal of Engineering and Applied Sciences* 10, no. 21 (2015): 1619–1628.

[74] T. Jayabarathi, T. Raghunathan, B. R. Adarsh, and P. N. Suganthan, "Economic Dispatch Using Hybrid Grey Wolf Optimizer," *Energy* 111 (2016): 630–641.

[75] J. Cai, Q. Li, L. Li, H. Peng, and Y. Yang, "A Hybrid CPSO-SQP Method for Economic Dispatch Considering the Valve-Point Effects," *Energy Conversion and Management* 53, no. 1 (2012): 175–181.

[76] R. Kumar, D. Sharma, and A. Sadu, "A Hybrid Multi-Agent Based Particle Swarm Optimization Algorithm for Economic Power Dispatch," *International Journal of Electrical Power & Energy Systems* 33, no. 1 (2011): 115–123.

[77] S. Chakraborty, T. Senjyu, A. Yona, A. Y. Saber, and T. Funabashi, "Solving Economic Load Dispatch Problem With Valve-Point Effects Using a Hybrid Quantum Mechanics Inspired Particle Swarm Optimisation," *IET Generation, Transmission & Distribution* 5, no. 10 (2011): 1042–1052.

[78] M. S. Alkoffash, M. A. Awadallah, M. Alweshah, R. A. Zitar, K. Assaleh, and M. A. Al-Betar, "A Non-Convex Economic Load Dispatch Using Hybrid Salp Swarm Algorithm," *Arabian Journal for Science and Engineering* 46, no. 9 (2021): 8721–8740.

[79] W.-M. Lin, H.-J. Gow, and M.-T. Tsai, "Combining of Direct Search and Signal-to-Noise Ratio for economic dispatch optimization," *Energy Conversion and Management* 52, no. 1 (2011): 487–493.

[80] M. A. Awadallah, M. A. Al-Betar, A. L. Bolaji, E. M. Alsukhni, and H. Al-Zoubi, "Natural Selection Methods for Artificial Bee Colony With New Versions of Onlooker Bee," *Soft Computing* 23, no. 15 (2019): 6455–6494.

[81] M. A. Al-Betar, M. A. Awadallah, A. T. Khader, A. L. Bolaji, and A. Almomani, "Economic Load Dispatch Problems With

Valve-Point Loading Using Natural Updated Harmony Search," *Neural Computing and Applications* 29, no. 10 (2018): 767–781.

[82] M. A. Al-Betar, M. A. Awadallah, A. T. Khader, and A. L. Bolaji, "Tournament-Based Harmony Search Algorithm for Non-Convex Economic Load Dispatch Problem," *Applied Soft Computing* 47 (2016): 449–459.

[83] P. Subbaraj, R. Rengaraj, and S. Salivahanan, "Enhancement of Self-Adaptive Real-Coded Genetic Algorithm Using Taguchi Method for Economic Dispatch Problem," *Applied Soft Computing* 11, no. 1 (2011): 83–92.

[84] R. Azizipanah-Abarghooee, T. Niknam, A. Roosta, A. R. Malekpour, and M. Zare, "Probabilistic Multiobjective Wind-Thermal Economic Emission Dispatch Based on Point Estimated Method," *Energy* 37, no. 1 (2012): 322–335.

[85] M. Basu, "Economic Environmental Dispatch Using Multi-Objective Differential Evolution," *Applied Soft Computing* 11, no. 2 (2011): 2845–2853.

[86] V. Basetti, S. S. Rangarajan, C. K. Shiva, et al., "Economic Emission Load Dispatch Problem With Valve-Point Loading Using a Novel Quasi-Oppositional-Based Political Optimizer," *Electronics* 10, no. 21 (2021): 2596.

[87] C.-Y. Lee and M. Tuegeh, "An Optimal Solution for Smooth and Non-Smooth Cost Functions-Based Economic Dispatch Problem," *Energies* 13, no. 14 (2020): 3721.

[88] F. Tariq, S. Alelyani, G. Abbas, A. Qahmash, and M. R. Hussain, "Solving Renewables-Integrated Economic Load Dispatch Problem by Variant of Metaheuristic Bat-Inspired Algorithm," *Energies* 13, no. 23 (2020): 6225.

[89] S. Gürgen, H. T. Kahraman, S. Aras, and İ. J. A. T. E. Altın, "A Comprehensive Performance Analysis of Meta-Heuristic Optimization Techniques for Effective Organic Rankine Cycle Design," *Applied Thermal Engineering* 213 (2022): 118687.