processes

MDPI

Article

A Social Group Optimization Algorithm Using the Laplace Operator for the Economic Dispatch Problem

Dinu Calin Secui [1], Cristina Hora [1,*], Florin Ciprian Dan [1], Monica Liana Secui [2], Horea Nicolae Hora [3] and Emil Gligor [4]

[1] Department of Energy Engineering, Faculty of Energy Engineering and Industrial Management, University of Oradea, 410058 Oradea, Romania; csecui@uoradea.ro (D.C.S.); florin.dan@uoradea.ro (F.C.D.)
[2] Department of Psychology, Faculty of Social-Humanistic Sciences, University of Oradea, 410058 Oradea, Romania; msecui@uoradea.ro
[3] Department of Mechanical Engineering and Vehicles, Faculty of Management and Technological Engineering, University of Oradea, 410058 Oradea, Romania; horahorea@yahoo.com
[4] Department of Civil Engineering, Faculty of Construction, Cadastre and Architecture, University of Oradea, 410058 Oradea, Romania; emil.gligor@uoradea.ro
* Correspondence: chora@uoradea.ro; Tel.: +40-259408206

Abstract: The economic dispatch (ED) problem focuses on the optimal scheduling of thermal generating units in a power system to minimize fuel costs while satisfying operational constraints. This article proposes a modified version of the social group optimization (SGO) algorithm to address the ED problem with various practical characteristics (such as valve-point effects, transmission losses, prohibited operating zones, and multi-fuel sources). SGO is a population-based metaheuristic algorithm with strong exploration capabilities, but for certain types of problems, it may stagnate in a local optimum due to a potential imbalance between exploration and exploitation. The new version, named SGO-L, retains the structure of the SGO but incorporates a Laplace operator derived from the Laplace distribution into all the iterative solution update equations. This adjustment generates more effective search steps in the solution space, improving the exploration–exploitation balance and overall performance in terms of solution stability and quality. SGO-L is validated on four power systems of small (six-unit), medium (10-unit), and large (40-unit and 110-unit) sizes with diverse characteristics. The efficiency of SGO-L is compared with SGO and other metaheuristic algorithms. The experimental results demonstrate that the proposed SGO-L algorithm is more robust than well-known algorithms (such as particle swarm optimization, genetic algorithms, differential evolution, and cuckoo search algorithms) and other competitor algorithms mentioned in the study. Moreover, the non-parametric Wilcoxon statistical test indicates that the new SGO-L version is more promising than the original SGO in terms of solution stability and quality. For example, the standard deviation obtained by SGO-L shows significantly lower values (6.02×10^{-9} USD/h for the six-unit system, 7.56×10^{-5} USD/h for the 10-unit system, 75.89 USD/h for the 40-unit system, and 4.80×10^{-3} USD/h for the 110-unit system) compared to SGO (0.44 USD/h for the six-unit system, 50.80 USD/h for the 10-unit system, 274.91 USD/h for the 40-unit system, and 1.04 USD/h for the 110-unit system).

Keywords: thermal units; valve-point effects; transmission losses; prohibited operating zone; human behavior-based techniques

check for updates

Received: 10 December 2024
Revised: 24 December 2024
Accepted: 16 January 2025
Published: 4 February 2025

Citation: Secui, D.C.; Hora, C.; Dan, F.C.; Secui, M.L.; Hora, H.N.; Gligor, E. A Social Group Optimization Algorithm Using the Laplace Operator for the Economic Dispatch Problem. *Processes* **2025**, *13*, 405. https://doi.org/10.3390/pr13020405

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The ED problem involves the optimal allocation of load among the generating units of a power system to minimize fuel costs over a given time period (typically one hour) while respecting the constraints imposed by the units and the system [1]. Furthermore, the solution provided by the ED problem must balance supply and demand in an efficient and sustainable manner. Moreover, solving the ED problem represents a measure of resource conservation, reduction of atmospheric pollution (by controlling the combustion of fossil fuels), and lowering operational costs without requiring significant investments [2].

The ED problem considers a series of practical constraints and characteristics (such as the production capacity of units, ramp-up and ramp-down rates, valve-point effects, transmission losses, multi-fuel sources, etc.), offering a feasible solution that can be implemented in reality [3]. Each generating unit has a cost power characteristic that is modeled in different ways depending on the desired accuracy (thus, there is a wide range of characteristics, namely linear, quadratic, cubic, or cubic with an added sinusoidal term, to highlight valve-point effects) [4]. The mathematical optimization model of the ED problem is nonlinear and discontinuous, comprising non-smooth functions with multiple local minima [5].

Over time, a wide variety of optimization methods have been used to solve the ED problem, grouped into three major categories [2], which are classical methods, artificial intelligence (AI)-based methods, and hybrid methods. From the first category, we mention techniques based on linear programming [6], quadratic programming [7], the lambda iteration algorithm [8], the Lagrange relaxation method [9], and the Newton–Raphson method [10]. These methods have several disadvantages, making them less attractive, such as the difficult handling of non-convex objective functions due to the need to evaluate the first and/or second ordinal derivatives, challenges in managing large-scale problems, and convergence to a local minimum/maximum depends on the starting solution's position [2]. AI-based methods are currently very popular and address some of the difficulties encountered by classical methods. In the case of AI methods, the functions involved in the optimization model can have high complexity (such as implicit models, non-differentiable, or black-box functions), do not require checks for continuity or differentiability, and do not involve derivative calculations. These methods are relatively easy to implement, are flexible, allow hybridization, and have the capability to identify the global solution [11]. For non-convex objective functions, AI-based methods are generally preferred over classical methods because they do not rely on gradient evaluations, have reduced sensitivity to the problem structure, and are equipped with multiple mechanisms (mutation, crossover, introduction of random elements, etc.) and strategies (learning, adaptation, selection, etc.) to avoid local optima. Despite these advantages, AI methods do not guarantee the attainment of a global solution, requiring multiple runs and tuning of specific parameters. The advantages/disadvantages of applying AI methods compared to classical methods also hold true in the particular case of solving the ED problem.

Among the AI methods used over time to solve the ED problem, we mention particle swarm optimization [12,13], genetic algorithms [14], differential evolution [15], cuckoo search algorithms [16], bat algorithms [17], backtracking search algorithms [18], firefly algorithms [19], Jaya algorithms [20], sine–cosine algorithms [21], slime mold algorithms [22], supply–demand optimizers [23], pelican optimization algorithms [24], social network search algorithms [3], and growth optimizer algorithms [25]. From the category of hybrid methods, we highlight the differential harmony search algorithm [1], the eagle-strategy supply–demand optimizer [23], the conglomerated ion motion and crisscross search optimizer [26], the kernel search optimizer with slime mold foraging [27], the hybrid gray wolf optimization algorithm [28], etc.

The next sections describe the main features of recent algorithms applied to the study of the ED problem. In [22], an improved version of the slime mold algorithm (ISMA) was developed, where solutions are updated based on two relations borrowed from the sine–cosine algorithm. The ISMA is applied to solve the ED problem with various characteristics, with results showing superior performance compared to well-known algorithms, such as particle swarm optimization, the Harris Hawk optimizer, tunicate swarm algorithms, jellyfish search optimizers, and the original slime mold algorithm.

To improve population diversity and reduce the likelihood of solutions being trapped in local optima, ref. [20] proposes a Jaya algorithm with self-adaptive multi-population combined with Lévy flights (Jaya-SML). The efficiency of Jaya-SML is tested on five power systems with varying practical characteristics, with results showing that Jaya-SML outperforms the standard Jaya algorithm in terms of cost, computational time, and robustness. It also surpasses other Jaya-based variations, such as Jaya with multi-populations or Jaya with multi-populations and self-adaptive strategies.

In [29], a new population-based algorithm (HcSCA) is proposed, which effectively combines the exploration capability of the sine–cosine algorithm with the local search ability of the hill-climbing algorithm. The HcSCA maintains a good balance between exploration and exploitation, being successfully applied to ten small- and large-scale power systems. A new optimization technique, called the hybrid Jaya optimization algorithm (Jaya–TLBO), is suggested in [30]. It leverages the advantages of the Jaya and teaching–learning-based optimization (TLBO) algorithms to improve population diversity, balance local–global search, and enhance the convergence of the original Jaya algorithm. Jaya–TLBO is used to determine the minimum cost in the ED problem for power systems with 6, 13, 20, and 40 units.

In [31], a competitive swarm optimizer equipped with an orthogonal learning strategy (OLCSO) is developed to quickly identify the most promising search direction, guiding the iterative process toward the global optimum. The new algorithm is tested on numerical functions and the ED problem with various complexities, showing that the orthogonal learning strategy improves CSO's convergence, robustness, and solution stability.

To improve the performance and efficiency of the white shark optimizer (WSO), a leader-based mutation selection mechanism is utilized in [32]. This mechanism enhances the exploitation phase, and the new algorithm (LWSO) is applied to optimize several non-convex mathematical functions and solve the ED problem. Results show that the LWSO delivers high-quality and stable solutions, particularly for small- and medium-sized power systems.

Building on the potential advantages of incorporating chaos into an algorithm's structure (e.g., better convergence, higher solution quality, etc.), ref. [33] introduces a new version of the slime mold algorithm (SMA), called the CSMA. Tests conducted on five power systems with varying practical complexities show that the CSMA outperforms the original SMA and other well-known algorithms, such as the moth flame optimizer, gray wolf optimizer, biogeography-based optimizer, grasshopper optimization algorithm, and multi-verse optimizer.

In [34], the improved fruit fly optimization algorithm is used to generate high-quality solutions for the economic dispatch problem in multi-area power systems. Recently, in [35], a disruption operator was integrated into the symbiotic organism search (SOS) algorithm to escape suboptimal points and enhance the exploration and exploitation capabilities of the original SOS during the optimization process. The new algorithm (DSOS) was tested on a wide range of power systems with different characteristics (e.g., transmission line losses, valve-point loading effects, ramp rate limits, prohibited operating zones). It demonstrates good convergence and high-quality solutions.

In [36], a new and effective optimization algorithm, called the turbulent flow of water-based optimization (TFWO), was developed for finding the global optimum of complex

numerical functions and real-world engineering problems. TFWO is characterized by good convergence and robustness, with the capability to identify high-quality and stable solutions for the ED problem.

Social group optimization (SGO) is a metaheuristic algorithm [37]—population-based and inspired by the social behavior of human groups—introduced relatively recently (2016) for minimizing test mathematical functions (unimodal, multimodal, composite). In [37,38], it was demonstrated that SGO outperforms several popular algorithms (GA, DE, PSO, or ABC) or their variants, such as cooperative PSO (CPSO), comprehensive learning PSO (CLPSO), fully informed PSO (FIPS-PSO), best-guided ABC (GABC), self-adaptive DE (SADE), and DE with an ensemble of parameters and mutation strategies (EPSDE), on certain mathematical functions.

Despite its advantages (good convergence and exploration capacity, simplicity of implementation, etc. [39]), SGO has limitations. As indicated in [38,40] SGO exhibits an inability to achieve the optimal solution for certain mathematical functions, likely due to an imbalance between exploration and exploitation. To address these shortcomings, several approaches to improving the algorithm's performance have been proposed, including enhancing the exploration–exploitation balance by introducing a self-awareness probability factor [40], hybridizing SGO with the whale optimization algorithm (SGO-WOA) [39], equipping SGO with various strategies based on inertia weight [41], developing chaotic SGO variants [42,43], etc.

Another way to improve the performance of a metaheuristic algorithm is the use of the Laplace operator, derived from the Laplace distribution, which was introduced by Deep and Thakur [44]. This operator has been successfully incorporated as a crossover operator in various algorithms (gravitational search algorithm [45], Harris Hawk optimization [46], salp swarm algorithm [47], Nelder–Mead hunger games search [48], sine–cosine algorithm [49], etc.) to enhance the exploration–exploitation balance.

Building on the positive experience obtained with the algorithms mentioned earlier, this article proposes a new approach to improving the performance of SGO by incorporating the Laplace operator into the structure of SGO's solution update relations. The new algorithm is referred to by the acronym SGO-L. The Laplace operator is introduced to generate more appropriate and efficient search steps for new solutions compared to SGO (an algorithm that uses only uniformly distributed random numbers in the interval (0, 1) to generate search steps).

By generating suitable search steps, the search space is better covered during the exploration and exploitation phases, positively influencing the convergence speed and the ability of the SGO-L algorithm to achieve higher-quality solutions. It is worth noting that to the best of the authors' knowledge (based on consultations with Elsevier and Scopus databases), the SGO algorithm has not previously been combined with the Laplace operator or applied to solve the ED problem. The main contributions of the research are as follows:

- The development of an efficient optimization technique (called SGO-L), in terms of cost and computational time, based on the inclusion of the Laplace operator into the structure of the original SGO algorithm.
- The implementation of the SGO-L algorithm to solve the economic dispatch (ED) problem, considering various practical complexities, such as valve-point loading effects, ramp-up and ramp-down rates, transmission line losses, prohibited operating zones, and multi-fuel sources.
- The validation of SGO-L's efficiency through testing on five power systems of varying sizes, with small (six-unit), medium (10-unit), and large (40-unit, and 110-unit) sizes.
- Highlighting through results and statistical testing that the new version, SGO-L, is more promising than the original SGO.

The paper is organized as follows: in Section 2, the mathematical optimization model for the ED problem is described. The following two sections present the SGO and SGO-L algorithms. Section 5 details the implementation of the proposed SGO-L algorithm for solving the ED problem. Section 6 presents and discusses the results obtained by SGO-L, while the final section summarizes the conclusions.

2. Statement of the ED Problem

The ED problem aims to determine the power outputs of thermal units and the type of fuel (in cases where units operate with multiple fuel sources) that minimize the fuel cost across a power system while adhering to a set of technical constraints imposed on the units and the system. The mathematical optimization model for the ED problem is presented below, encompassing the decision variables, the objective function, and the constraint relationships. We consider a power system comprising m thermal generating units, with their operating powers represented as a vector $[P] = [P_1, P_2, \ldots, P_j, \ldots, P_m]$ and the corresponding fuel types denoted as a vector $[FT] = [FT_1, FT_2, \ldots, FT_j, \ldots, FT_m]$, where P_j is the power output of unit j and TF_j is the type of fuel utilized by unit j.

The optimization variables are the power outputs of the thermal units $[P]$. Additionally, for units with multiple fuel sources, such as gas, oil, or coal (if the system includes such units), the type of fuel [FT] used by these units must also be specified.

The objective function $F[P]$, expressed in USD/h, aims to minimize the fuel cost for all units in the analyzed system:

$$minF[P] = \sum_{j=1}^{m} F_j(P_j) \tag{1}$$

where $F_j(P_j)$ represents the cost power characteristic of unit j, expressed in USD/h.

Mathematically, the cost power dependency $F_j(P_j)$ is traditionally expressed as a quadratic function in the form [50] of

$$F_j(P_j) = c_j P_j^2 + b_j P_j + a_j \tag{2a}$$

For a more realistic modeling approach, the characteristic $F_j(P_j)$ from Equation (2a) is complemented by a sinusoidal term that accounts for valve-point effects (VPE), resulting in the following form [50]:

$$F_j(P_j) = c_j P_j^2 + b_j P_j + a_j + \left| e_j sin(f_j(P_{j,min} - P_j)) \right| \tag{2b}$$

where a_j, b_j, and c_j are the fuel cost coefficients for unit j; e_j and f_j are the coefficients of the thermal unit j when the valve-point effects are considered; and $P_{j,min}$ is the minimum real power for unit j in MW.

If the units operate with multiple fuel sources (p types of fuel), the cost power dependency $F_j(P_j)$ is mathematically expressed in the form [14] of

$$F_j(P_j) = \begin{cases} a_{jt} P_j^2 + b_{jt} P_j + c_{jt} + \left| e_{jt} sin(f_{jt}(P_{jt,min} - P_j)) \right| \\ if\ P_{jt,min} \le P_j \le P_{jt,max}\ for\ fuel\ type\ t = 1, 2, \ldots, T_j \end{cases} \tag{2c}$$

where $P_{jt,min}$ and $P_{jt,max}$ are the minimum and maximum power limits of unit j operating with fuel type t (t = 1, 2, …, T_j) in MW; T_j represents the number of fuel types with which unit j can operate; and a_{jt}, b_{jt}, c_{jt}, e_{jt}, and f_{jt} are the fuel cost coefficients of unit j using fuel type t.

The constraint relationships define the range of values within which the optimal solution is sought, and they are presented below [5,12].

$$P_{j,min} \le P_j \le P_{j,max} \tag{3}$$

$$P_j^0 - DR_j \le P_j \le P_j^0 + UR_j \tag{4}$$

$$P_{j,z}^l \le P_j \le P_{j,z}^u, j = 1, 2, \ldots, m, z = 1, 2, \ldots, nz_j \tag{5}$$

$$P_G - P_L - P_D = 0 \tag{6}$$

where $P_{j,min}$ and $P_{j,max}$ represent the minimum and maximum power limits of unit j; P_j^0 is the previous hour active power of unit j; DR_j and UR_j are the down-ramp and up-ramp limits of the j unit; $P_{j,z}^l$ and $P_{j,z}^u$ are the lower bound and upper bound of the zth prohibited operating zone for unit j; nz_j represents the number of prohibited zones of unit j; P_D is the power demand of the system; P_G represents the total generated power; and P_L is the transmission loss in the system, usually calculated by Kron's formula [12].

$$P_L = \sum_{i=1}^{m} \sum_{j=1}^{m} P_i B_{ij} P_j + \sum_{i=1}^{m} B_{0i} P_i + B_{00} \tag{7}$$

where B_{ij}, B_{0i}, and B_{00} are the loss coefficients, considered constant and known.

Equation (3) defines the limits within which the generating units can operate. Equation (4), known as ramp rate constraints, indicate by how much the power of unit j can increase (UR_j) or decrease (DR_j) at a given moment when the power from the previous hour (P_j^0) is fixed. Generating units may have zones in which they cannot operate, known as prohibited operating zones (POZ), due to vibrations or constraints imposed by the system operator. These restrictions are defined by inequalities (5). The final relationship in the set of constraints—Equation (6)—refers to the power balance across the entire system, which must be satisfied within the imposed error ε. The accuracy of solutions [P] is determined using the following equation:

$$Accuracy = \mid P_G([P]) - P_L[P] - P_D \mid < \varepsilon \tag{8}$$

3. Social Group Optimization (SGO)

SGO is a metaheuristic algorithm proposed relatively recently by Satapathy and Naik [37] for solving complex optimization or engineering problems. This algorithm mimics the behavior of individuals within a social group aiming to solve real-life problems. Conceptually, achieving this objective is realized through interactions and the exchange of knowledge among members of the social group.

The mathematical modeling of the social group concept involves the following notations: N represents the number of individuals (the population) in the social group and m is the number of traits corresponding to an individual in the group. Each individual i in the social group corresponds to a solution $X_i = [x_{1,i}, x_{2,i}, \ldots, x_{j,i}, \ldots x_{m,i}]$, $i = 1, 2, \ldots, N$ from the population, $x_{j,i}$ is the component j of the solution X_i, and $f_i(X_i)$ is the fitness function for the solution X_i.

The SGO algorithm progresses through three phases [37], namely the initialization phase, the improving phase, and the acquiring phase.

In the initialization phase, solutions X_i are randomly generated within the imposed minimum (X_{min}) and maximum (X_{max}) limits using the following equation:

$$X_i = X_{min} + r_1 \cdot (X_{max} - X_{min}), i = 1, 2, \ldots, N \tag{9}$$

In the improving phase, to enhance the traits of individual i ($X_{i,new}$), the interaction between individual i (X_i) and the individual with the best knowledge in the group (considered the best individual, X_{Best}) is modeled. Solutions are updated using the following equation:

$$X_{i,new} = c \cdot X_i + r_2 \cdot (X_{Best} - X_i), i = 1, 2, \ldots, N \tag{10}$$

The acquiring phase models the exchange of knowledge between an individual i (solution X_i) and the best individual in the group (X_{Best}), as well as between individual i and a randomly selected individual r from the group (solution X_r). In this phase, solutions are updated based on the performance of the current solution X_i compared to the randomly selected solution X_r, using the following equations:

$$\text{If } f(X_i) < f(X_r) \text{ Then, } X_{i,new} = X_i + \mathrm{r}_3 (X_i - X_r) + r_4 (X_{Best} - X_i) \quad (11a)$$

$$\text{Else } X_{i,new} = X_i + r_3 (X_r - X_i) + \mathrm{r}_4 (X_{Best} - X_i), \; i = 1, 2, \ldots, N \quad (11b)$$

where

$X_{i,new}$ is the new solution obtained through Equations (10) and (11);
X_{Best} represents the individual with the best knowledge in the group (the best solution);
X_r is a randomly selected solution from the current population;
c is the self-introspection factor ($0 < c < 1$).

In this study, the value of c is set according to the recommendations in [37,38,40], specifically $c = 0.2$; r_1, r_2, r_3, and r_4 are m-dimensional vectors, with their components being uniformly distributed random numbers in the interval (0, 1).

The SGO algorithm starts by generating an initial population using Equation (9), followed by an iterative cycle that sequentially performs the improving phase and the acquiring phase, where solutions are updated using Equations (10) and (11). At the end of each iteration, it is checked whether the predefined maximum number of iterations (k_{max} has been reached. If k_{max} has not been reached, the iterative cycle repeats; otherwise, the algorithm terminates, retaining the best solution (X_{Best}) and its corresponding fitness $f(X_{Best})$. Pseudo-code of the SGO algorithm is presented below [37] (Algorithm 1).

Algorithm 1: SGO algorithm.

Set the parameters of the SGO algorithm: the population size (N), maximum number of iterations (k_{max})
{*Initialization phase*}
Initialize the population X_i, $i = 1, 2, \ldots, N$ in the social group using Equation (9).
Calculate the fitness $f(X_i)$ for each solution X_i, then find X_{Best} and $f(X_{Best})$
while $k \leq k_{max}$
  for $i = 1$ to N **do** {*Improving phase*}
    Calculate the new solution $X_{i,new}$ using Equation (10)
    Calculate the fitness $f(X_{i,new})$
    Update the solution X_i and $f(X_i)$ using greedy selection
  end for
  Find X_{Best} and $f(X_{Best})$ from the updated population
  for $i = 1$ **to** N **do** {*Acquiring phase*}
    Randomly select a solution X_r, where $i \neq \mathrm{r}$
    Calculate the new solution $X_{i,new}$ using Equations (11a) and (11b)
    Calculate the fitness $f(X_{i,new})$
    Update the solution X_i and $f(X_i)$ using greedy selection
  end for
    Find X_{Best} and $f(X_{Best})$ from the updated population
    $k = k + 1$
end while
Return X_{Best}, and the fitness $f(X_{Best})$.

4. Modified SGO Operator Laplace (SGO-L)

The proposed SGO-L algorithm progresses through the same phases as the SGO algorithm; however, certain characteristics of SGO are modified to improve the performance of SGO-L. In SGO, the updating of solutions is performed using search steps dependent on the generation of random numbers (r_1–r_4 from Equations (9)–(11)) within the range (0, 1). For certain problems with higher complexity, generating search steps solely using random numbers (r_1–r_4) is insufficient to perform an efficient search starting from various initial points. To address some of the shortcomings resulting from a less efficient search (e.g., stagnation in local minima, low global/local search capability, poor and unstable solution quality), the Laplace operator has been integrated into the structure of SGO-L. This operator is based on the Laplace distribution, which has a probability density function $f(x)$ and a distribution function $F(x)$ defined by the following equations [44]:

$$f(x) = \frac{1}{2b} exp\left(-\frac{|x-a|}{b}\right), \mathrm{x} \in \mathrm{R} \tag{12}$$

$$F(x) = \begin{cases} \frac{1}{2} exp\left(-\frac{|x-a|}{b}\right), & x \leq a \\ 1 - \frac{1}{2} exp\left(-\frac{|x-a|}{b}\right), & x > a \end{cases} \tag{13}$$

where a is the location parameter ($a \in$ R) and b is the scale parameter ($b > 0$).

By applying the inverse operation on the function $F(x)$ from Equation (13), we obtain the Laplace operator, denoted as $L(u,a,b)$, which follows a Laplace distribution. $L(u,a,b)$ depends on the parameters of the Laplace distribution (a, b) and a random number u uniformly distributed in the interval (0, 1) [44].

$$L(u,a,b) = \begin{cases} a - b \cdot ln(u), & u \leq a \\ a + b \cdot ln(u), & u > a \end{cases} \tag{14}$$

The Laplace operator $L(u,a,b)$ is incorporated into both Equations (10) and (11), with the role of perturbing each new component of the vector $X_{i,new}$ during the improving phase and the acquiring phase. Depending on the parameters a and b, the values generated by the Laplace operator $L(u,a,b)$ from Equation (14) can sometimes be excessively large, frequently resulting in inadequate search steps that cause significant jumps in the search space. To limit the frequency of these large jumps and improve the exploration-to-exploitation transition, the values generated by the Laplace operator $L(u,a,b)$ are gradually reduced during each iteration k if they exceed a predefined maximum value L_{Max}. The resulting Laplace operator, denoted $L\text{-}lim(u,a,b)$, is defined by Equation (15).

$$L - lim(u,a,b) = \begin{cases} \left(1 - \frac{k-1}{k_{max}}\right) \cdot L(u,a,b), & L(u,a,b) > L_{Max} \\ L(u,a,b), & otherwise \end{cases} \tag{15}$$

Figure 1 shows the first 10,000 values simulated using the Laplace operators $L(u,a,b)$ and $L\text{-}lim(u,1,1)$ considering the Laplace distribution parameters a = 1 and b = 1. The operator L(u,1,1), represented in blue, exhibits high values throughout the range of simulations, which could lead to excessively large search steps toward the end of the search process. In contrast, $L\text{-}lim(u,1,1)$, represented in red, initially shows higher values (allowing for predominantly extensive exploration to identify favorable regions). However, as the number of simulations increases, the values are attenuated and plateau at the predefined maximum limit L_{Max}. Thus, the attenuated values of $L\text{-}lim(u,1,1)$ favor the exploitation phase, enabling intensive searching around the current solutions.

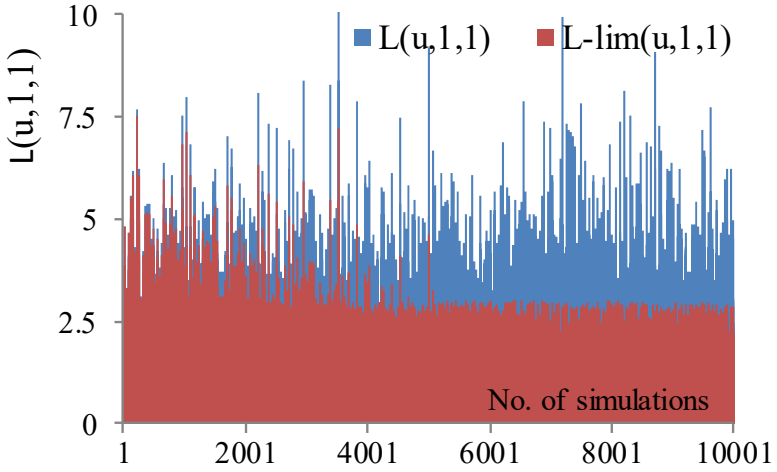


Figure 1. Values generated by the operators L(u,1,1) and L-lim(u,1,1).

Modifications made to the proposed SGO-L algorithm compared to SGO are as follows:
(i). In the improving phase, Equation (10) is replaced with Equation (16) in the followig form:

$$X_{i,new} = c \cdot X_i + L\text{-}lim(u,a,b) \cdot r_2 \cdot (X_{Best} - X_i),\ i = 1, 2, \ldots, N \quad (16)$$

(ii). In the acquiring phase, Equation (11) is replaced with Equation (17) in the following form:

$$\text{If } f(X_i) < f(X_r) \text{ Then, } X_{i,new} = X_i + L\text{-}lim(u,a,b) \cdot r_3\ (X_i - X_r) + L\text{-}lim(u,a,b) \cdot r_4\ (X_{Best} - X_i) \quad (17a)$$

$$\text{Else } X_{i,new} = X_i + L\text{-}lim(u,a,b) \cdot r_3\ (X_r - X_i) + L\text{-}lim(u,a,b) \cdot r_4\ (X_{Best} - X_i),\ i = 1, 2, \ldots, N \quad (17b)$$

(iii). The verification and update of the best solution (X_{Best}) will be performed after generating each solution ($X_{i,new}$) in both the improving phase and the acquiring phase. Additionally, the value of the factor c will be considered as $c = 1$.

The specific improvements of the SGO-L algorithm compared to SGO result from the modifications included in the structure of SGO-L, as presented in points (i)–(iii) mentioned above. The main improvement comes from the inclusion of the Laplace operator with bounded values, *L-lim*(*u*,*a*,*b*), in the structure of the update relationships of the solutions, both in the improving phase and in the acquiring phase. This operator was introduced to generate suitable search steps during the exploration and exploitation phases of the solution space to maintain a good exploration–exploitation balance, which ultimately allows for the attainment of high-quality solutions and good stability with rapid convergence. Additionally, according to modification (iii), the update–evaluation cycle of the solutions is carried out dynamically, thus allowing for the immediate exploitation of an improved solution, resulting in an increase in the rate of convergence.

Considering the modifications (i)–(iii), the steps for applying the proposed SGO-L algorithm are presented below (Algorithm 2).

The main steps of the suggested SGO-L algorithm are summarized in Figure 2 as a flowchart. Also, the modifications made to the proposed SGO-L are highlighted.

Algorithm 2: SGO-L algorithm.

Set the general parameters (N, k_{max}) and specific parameters (a, b, and L_{Max}) for SGO-L
{*Initialization phase*}
Initialize the population X_i, $i = 1, 2 \ldots, N$ in the social group using Equation (9)
Calculate the fitness $f(X_i)$, then find X_{Best} and $f(X_{Best})$
while $k \leq k_{max}$
  for $i = 1$ **to** N **do** {*Improving phase*}
    Generate a value using the Laplace operator *L-lim*(u,a,b) using (15)
    Calculate the new solution $X_{i,new}$ using Equation (16)
    Calculate the fitness $f(X_{i,new})$
    Update the solutions X_i and X_{Best} and their corresponding fitness values $f(X_i)$ and $f(X_{Best})$ using greedy selection
  end for
  for $i = 1$ **to** N **do** {*Acquiring phase*}
    Randomly select a solution X_r, $i \neq$ r
    Generate a value using the Laplace operator *L-lim*(u,a,b) based on Equation (15)
    Calculate the new solution $X_{i,new}$ using Equation (17)
    Calculate the fitness $f(X_{i,new})$
    Update the solutions X_i and X_{Best} and their corresponding fitness values $f(X_i)$ and $f(X_{Best})$ using greedy selection
  end for
$k = k + 1$
end while
Return X_{Best}, and the fitness $f(X_{Best})$.

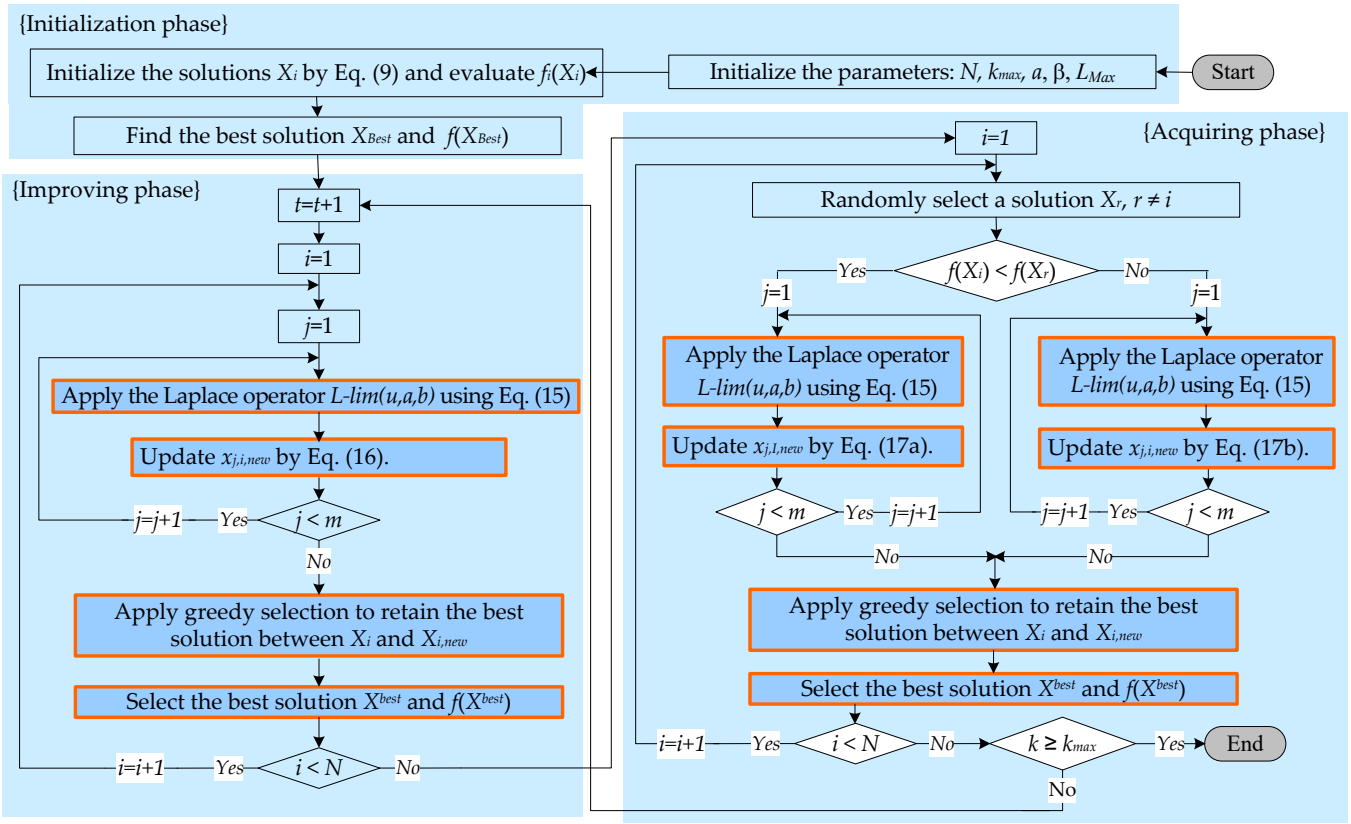


Figure 2. Flowchart of the SGO-L algorithm.

5. Implementing the SGO-L for the ED Problem

To implement the SGO-L or SGO algorithm, we consider that any solution i is represented by the power vector $[P_i] = [P_{1,i}, P_{2,i}, \ldots, P_{j,i}, \ldots P_{m,i}]$ and the vector of the fuel types used $[TF_i]$, $i = 1, 2, \ldots, N$ (if there are units with multiple fuel sources).

To solve the ED problem using the proposed SGO-L algorithm, Algorithm 2 (presented in Section 4) is applied, with the following additions related to handling inequality and equality constraints—relations (3)–(6) presented in the optimization model [5,12]:

(i). If the power of a unit j (P_j, $j = 1, 2, \ldots, m$) is outside the operating limits ($P_{j,min}$, $P_{j,max}$) defined by relation (3), it is set to the exceeded limit ($P_j = P_{j,min}$ or $P_j = P_{j,max}$);

(ii). If the power of a unit j (P_j, $j = 1,2, \ldots, m$) is outside the operating limits (Max($P_{j,min}$, ${P^0}_j - DR_j$), Min($P_{j,max}$, ${P^0}_j + UR_j$)) defined by the combination of relations (3)–(4), it is set to the exceeded limit (P_j = Max($P_{j,min}$, ${P^0}_j - DR_j$) or P_j = Min($P_{j,max}$, ${P^0}_j + UR_j$));

(iii). If the power of a unit j (P_j, $j = 1, 2, \ldots, m$) is within a prohibited operating zone (${P^l}_{j,z}$, ${P^u}_{j,z}$) defined by relation (5), it is set to the upper limit of zone z ($P_j = {P^u}_{j,z}$);

(iv). To handle the equality constraint (6), we use the solution adjustment procedure described in [5].

It should be noted that the corrections specified in points (i)–(iii) are applied to adjust each component $P_{j,i}$ of the power vector $[P_i]$ in all phases of SGO-L (initialization, improving, acquiring). The correction from point (iv) is applied to each vector generated through Equations (9), (16), and (17), resulting in a new power vector that satisfies relation (6) with the predefined accuracy ε. The corrections specified in points (i)–(iv) are used similarly when applying the SGO algorithm to solve the ED problem. The type of fuel used, represented by the vector $[FT_i]$, is determined based on the power intervals (known values from the problem's input data) within which the components $P_{j,i}$ of the power vector $[P_i]$ are placed after applying the corrections (i)–(iv).

6. Case Studies

In this section, the performance of the SGO-L algorithm is tested on four power systems with various practical characteristics, such as the valve-point effect, transmission line losses, sources with multiple fuel options, prohibited zones of generating units, and the operating limits of the generators. Based on the size and characteristics of the power systems, we study four cases mentioned in Table 1 and described below.

Table 1. Characteristics of the analyzed systems and values set for the SGO and SGO-L parameters.

Cases	SGO and SGO-L (General Parameters)				SGO-L Specific Parameters			P_L	VPE	POZ	MFS
	m	N	k_{max}	NE	a	b	L_{Max}				
C_1	6	7	100	1400	0	0.5	3	√	-	√	-
C_2	10	7	100	1400	1	0.5	3	√	√	-	-
C_3	10	20	400	16,000	1.5	1.0	3	-	√	-	√
C_4	40	50	400	40,000	0.1	0.35	3	-	√	-	-
C_5	110	70	150	21,000	2	1.25	3	-	-	-	-

P_L: transmission line losses; VPE: valve-point effects; POZ: prohibited operating zones; MFS: multiple fuel sources.

Case 1 (C_1): Case C_1 represents an IEEE power system with six thermal units, considering POZs, ramp rate limits, and transmission losses. The power demand is 1263 MW. The cost coefficients (a_j, b_j, c_j, $j = 1, 2, \ldots, 6$) and B-loss coefficients are taken from [12] with the mention that the B_{00} term in the loss expression has been corrected according to the details in [51].

Case 2 (C_2): This case studies a 10-unit system that considers transmission line losses and the valve-point effect. The power demand is 2000 MW. The cost coefficients (a_j, b_j, c_j, e_j, f_j, j = 1, 2, . . ., 10) and loss coefficients (B_{ij}) are taken from [52].

Case 3 (C_3): The third case analyzes a 10-unit system that considers multi-fuel sources and the valve-point effects. The power demand is 2700 MW. Input data for the generating units are taken from [14].

Case 4 (C_4): Case C4 is a large-scale system with 40 units, which considers the valve-point effects. The power demand is 10,500 MW, and the cost coefficients are taken from [50].

Case 5 (C5): The last case is a system with 110 units, which does not consider valve-point effects or transmission line losses. The power demand is 15,000 MW, and the input system data are available in [22]. For each case (C_1–C_4), 50 independent runs of the SGO and SGO-L algorithms were performed, and a set of statistical indicators was determined as follows: the best cost obtained for the objective function F (*MinCost*), the average cost (*MeanCost*), the maximum cost (*MaxCost*), and the standard deviation (*SD*). Additionally, the average execution time (*Time*) is presented for both the SGO and SGO-L algorithms. The accuracy of the calculations was set to $\varepsilon = 10^{-5}$ MW.

Both algorithms (SGO and SGO-L) were implemented in MathCAD and executed on a computer with the following specifications: Intel i5 processor, 2.2 GHz CPU, and 4 GB of RAM.

6.1. Setting the Parameters

The tuning of metaheuristic algorithm parameters is crucial, as it significantly influences their performance. Therefore, the general parameters (k_{max} and N) and specific parameters (a, b, and L_{Max} for SGO-L) were set by conducting experimental trials. The adjustment of the parameters for the SGO and SGO-L algorithms was carried out based on the following rule: when one parameter is varied, the other parameters are kept constant. The best combination of parameter values was chosen based on the "mean cost" criterion (MeanCost item) for the objective function $F[P]$ defined in Equation (1). The selected values for the general parameters (N, and k_{max}) and specific parameters (a, b, and L_{Max} in the case of SGO-L) are presented in Table 1.

For a fair comparison between the original SGO and SGO-L algorithms, as shown in Table 1, the values of the general parameters (k_{max} and N) and the number of evaluations ($NE = 2N \cdot k_{max}$) were chosen to be equal for each case considered (C_1–C_4). Furthermore, it was ensured that both algorithms started from the same initial points (solutions).

6.2. Results for Case C_1 (Six Units)

The best solutions for SGO and SGO-L: Table 2 shows the best solutions obtained by the original SGO and SGO-L algorithms, as well as by several competitor algorithms (MSSA [53], DHS [1], PSO [12], CBA [17], GA [12], and FA [19]). Additionally, the values of the statistical metrics (*MinCost*, *MeanCost*, *MaxCost*, and *SD*), the transmission losses (P_L), total generated power (P_G), the power demand (P_D), accuracy (ε), average execution time (*Time*), and the number of evaluations (*NE*) for the objective function defined by relation (1) are indicated. Observing the best solutions identified by SGO and SGO-L, it can be noted that they are nearly identical, with the generating units operating close to their maximum power limit (P_{max}). For both algorithms (SGO and SGO-L), the best operating cost of the system (corresponding to the best solution) is approximately 15,449.89952 USD/h, a slightly better value than that obtained by other algorithms listed in Table 2, such as the MSSA [53], DHS [1], and PSO [12], among others. It is worth mentioning that the best solutions obtained by SGO-L have very high accuracy, with *Accuracy* < 1.0×10^{-8} MW.

Table 2. Generation schedule obtained by SGO and SGO-L for case C_1 (6 units, P_D = 1263 MW).

Outputs	SGO	SGO-L	MSSA [53]	DHS [1]	PSO [12]	CBA [17]	GA [12]	FA [19]
P_1 (MW)	447.5044797	447.5032965	447.5029	447.5285	447.4970	-	474.8066	439.293500
P_2 (MW)	173.3176530	173.3183787	173.3186	173.2791	173.3221	-	178.6363	175.0455644
P_3 (MW)	263.4629599	263.4630573	263.463	263.4772	263.4745	-	262.2089	265.00000
P_4 (MW)	139.0645127	139.0654381	139.0656	139.0291	139.0594	-	134.2826	140.7740665
P_5(MW)	165.4738233	165.4732240	165.473	165.4864	165.4761	-	151.9039	167.042084
P_6 (MW)	87.13483200	87.13484260	87.1349	87.1587	87.1280	-	74.1812	88.78165465
P_G (MW)	1275.9582606	1275.9582372	1275.958	1275.9590	1276.01	-	1276.03	1275.9367
P_L (MW)	12.95826052	12.9582372	12.958	12.9590	12.9584	-	13.0217	12.93686954
P_D (MW)	1263	1263	1263	1263	1263	1263	1263	1263
Accuracy (MW)	7.73×10^{-8}	3.24×10^{-9}	14×10^{-12}	8.89×10^{-9}		-	2.21×10^{-3}	1.09×10^{-8}
MinCost (USD/h)	15,449.899525	15,449.8995248	15,449.8995	15,449.8996	15,450	15,450.2381	15,459	15,450.50896
MeanCost (USD/h)	15,450.055154	15,449.899525	15,449.937	15,449.9264	15,454	15,454.76	15,469	15,452.53099
MaxCost (USD/h)	15,452.192472	15,449.899525	15,453.545	15,449.9884	15,492	15,518.6588	15,524	15,458.44268
SD (USD/h)	4.46×10^{-1}	6.02×10^{-9}	0.3647	0.0204	-	2.965	-	2.048
Cost saving * (USD/yr)	1363.31	0	328.28	235.43	35,920.16	42,577.76	167,320.20	23,051.63
Time (s)	0.07	0.07	1.16	0.01	14.89	0.704	41.58	1.965
NE	1400	1400	12,000	3000	20,000	12,000	20,000	60,000

* Represents the cost saved by the SGO-L algorithm compared to competitor algorithms.

Quality and stability of solutions: The solutions obtained by SGO and SGO-L are evaluated through the statistical metrics (*MinCost*, *MeanCost*, *MaxCost*, and *SD*) presented in Table 2. Examining these values reveals that the proposed SGO-L algorithm identifies higher-quality solutions than the competitor algorithms mentioned in Table 2 (SGO, MSSA [53], DHS [1], PSO [12], CBA [17], GA [12], and FA [19]). Additionally, the stability of SGO-L is excellent (with *SD* = 7.56×10^{-5} USD/h) considering that a relatively large number of independent runs—50 in total—were performed. Compared to the competitor algorithms listed in Table 2, SGO-L exhibits superior stability. Furthermore, the maximum value obtained by SGO-L (*MaxCost* = 15,449.899525 USD/h) is better than the average value (*MeanCost*) identified by the competitor algorithms in Table 2.

Computation time: For both algorithms (SGO and SGO-L), the computation time (*Time*) is very good, being approximately 0.07 s per run, which is better than the times achieved by the competitor algorithms. The only exception is the DHS [1] algorithm, which achieves a slightly lower execution time compared to SGO and SGO-L.

6.3. Results for Case C_2 (10 Units)

The best solutions for SGO and SGO-L: Table 3 presents the best generation scheduling solutions obtained by the SGO and SGO-L algorithms, as well as several competitor algorithms (IPOA [24], OMF [54], PSO [22], HHO [22], ISMA [22], and CSCA [21]).

From the solutions obtained by the algorithms listed in Table 3, it is observed that most generators (P_1, P_2, P_7–P_{10}) operate at maximum capacity, with differences between the solutions identified by the mentioned algorithms being attributable to units P_3–P_6. The best operating cost is achieved by the proposed SGO-L algorithm (*MinCost* = 111,497.63013 USD/h), but very similar values are also identified by other algorithms, such as SGO, OMF [54], PSO [22], the ISMA [22], or the CSCA [21]. An exception is observed for the IPOA [24] and HHO [22] algorithms, which result in higher operating costs.

Quality and stability of solutions: From Table 3, it can be seen that the proposed SGO-L algorithm outperforms all competitor algorithms, achieving a better set of quality values for cost indicators (*MinCost*, *MeanCost*, *MaxCost*, and *SD*). The very low values of the *SD* indicator ($SD_{\text{SGO-L}}$ = 7.56×10^{-5} USD/h) indicate excellent stability for the SGO-L algorithm.

Table 3. Generation schedule obtained by SGO and SGO-L for case C_2 (P_D = 2000 MW).

Outputs	SGO	SGO-L	IPOA [24]	OMF [54]	PSO [22]	HHO [22]	ISMA [22]	CSCA [21]
P_1 (MW)	55	55	-	55	55.000	55.000	55.000	55.0000
P_2 (MW)	80	80	-	80	80.000	80.000	80.000	80.0000
P_3 (MW)	106.93785817	106.93954899	-	106.8771	106.844	108.9857	106.867	106.9274
P_4 (MW)	100.58454530	100.57713786	-	100.7023	100.757	95.69826	100.544	100.585
P_5(MW)	81.49943791	81.50071003	-	81.5370	81.522	85.75969	81.606	81.5182
P_6 (MW)	83.01695688	83.02141094	-	82.9221	82.917	81.62431	83.022	83.0082
P_7 (MW)	300	300	-	300	300.000	300.000	300.000	300.0000
P_8 (MW)	340	340	-	340	340.000	340.000	340.000	340.0000
P_9 (MW)	470	470	-	470	470.000	470.000	470.000	470.0000
P_{10} (MW)	470	470	-	470	469.999	470.000	470.000	470.0000
P_G (MW)	2087.038798	2087.038808	-	2087.0387	2087.0390	2087.0679	2087.0390	2087.0388
P_L (MW)	87.038808	87.038818	87.0778	87.0386	87.0383	87.06791	87.0387	87.0388
P_D (MW)	2000	2000	2000	2000	2000	2000	2000	2000
Accuracy (MW)	9.99×10^{-6}	9.99×10^{-6}	-	-	7.263×10^{-5}	0	1.2×10^{-6}	7.21×10^{-6}
MinCost (USD/h)	111,497.63014	111,497.63013	111,501.2653	111,497.640	111,497.64	111,503.9	111,497.63	111,497.6307
MeanCost (USD/h)	111,533.51883	111,497.63021	-	111,497.64	111,497.71	111,566.9	111,497.66	-
MaxCost (USD/h)	111,667.53617	111,497.63049	-	111,497.642	111,497.83	111,664.6	111,497.72	-
SD (USD/h)	50.80	7.56×10^{-5}	-	3.9×10^{-4}	66.44×10^{-4}	35.06826	283.9×10^{-4}	-
Cost saving (USD/yr)	314,384.3	0.0	-	85.8	699.0	606,803.4	261.0	-
Time (s)	0.09	0.09	2.322	0.031	123.00	181.7545	75.35	-
NE	1400	1400	30,000	-	1,000,000	1,000,000	1,000,000	15,000

Since the *SD* values obtained by SGO-L are lower than those achieved by competitor algorithms, SGO-L demonstrates superior stability. Similarly to Case C_1, the maximum value obtained by SGO-L (*MaxCost* = 111,497.63049 USD/h) is better than the mean cost, *MeanCost* (and very close to the minimum cost, *MinCost*), identified by the competitor algorithms in Table 3.

Computation time: For both algorithms (SGO and SGO-L), the average execution time (*Time*) is excellent, being approximately 0.09 s per run, which is better or significantly better than the time achieved by the competitor algorithms. This can also be observed from the fact that SGO-L requires a small number of evaluations (*NE* = 1400) of the objective function to reach the optimal solution compared to the competitor algorithms in Table 3, which require a much higher number of evaluations, *NE* > 15,000).

6.4. Results for Case C_3 (10 Units with Multi-Fuel)

The best solutions for SGO and SGO-L: The best results obtained by SGO and SGO-L for the scheduling of generating units with multiple fuel sources are presented in Table 4. For this case, the solution consists of the unit power outputs (P_1–P_{10}) and the type of fuel used. Both SGO and SGO-L identify the same fuel types for the obtained solutions. The accuracy of the power balance—relation (8)—corresponding to the solutions identified by SGO and SGO-L is high, with *accuracy* < 2×10^{-6} MW. The best operating cost is achieved by SGO-L (*MinCost* = 623.82940 USD/h), which is better than the cost identified by SGO or other algorithms mentioned in Table 5.

Table 4. Generation schedule obtained by SGO and SGO-L for case C_3 (P_D = 2700 MW).

Outputs [P]	Fuel Type [FT]	SGO	SGO-L
P_1 (MW)	2	218.599049	218.593998
P_2 (MW)	1	210.476474	211.711740
P_3 (MW)	1	279.628226	279.648851
P_4 (MW)	3	240.042488	239.236315
P_5(MW)	1	276.261387	279.934521
P_6 (MW)	3	240.714419	239.639428
P_7 (MW)	1	290.185227	287.727493
P_8 (MW)	3	240.848871	239.908170
P_9 (MW)	3	430.656780	427.730874
P_{10} (MW)	1	272.587078	275.868609
P_D (MW)		2700	2700
Accuracy (MW)		1.61×10^{-6}	1.62×10^{-6}
MinCost (USD/h)		623.87730	623.82940

Table 5. Values of the statistical items obtained by different algorithms for case C_3 (10 units, P_D = 2700 MW).

Algorithm	*MinCost* (USD/h)	*MeanCost* (USD/h)	*MaxCost* (USD/h)	*SD* (USD/h)	*Time* (s)	Cost Saving * (USD/yr)
Jaya-SML [20]	639.9738	624.0468	624.1300	0.03	7.31	1356.7
IGA-MU [14]	624.5178	625.8692	630.8705	–	7.32	17,320.9
PSO [13]	624.2449	624.2543	624.2744	2.8×10^{-3}	1.45	3174.4
GC-Jaya [55]	624.2249	624.6812	628.2811	0.66	2.78	6914.0
MSFLA [56]	624.1156	624.8958	628.3428	–	–	8793.9
M_DE [57]	624.1026	–	–	–	0.49	–
IJaya [55]	623.9383	624.0080	624.0684	0.03	5.18	1016.8
FA [19]	623.9351	623.9939	624.2512	0.10	2.43	893.3
DE [15]	623.9280	624.0068	624.0653	0.02	0.62	1006.3
BSA [18]	623.9016	623.9757	624.0838	–	0.25	733.8
IFA [58]	623.8768	625.2704	629.2765	–	–	12,075.4
CSA [16]	623.8684	623.9495	626.3666	0.24	1.58	504.3
MCSA [59]	623.8558	–	–	–	2.34	–
GHS [56]	623.8491	624.1341	625.3157	–	–	2121.4
SFLA-GHS [56]	623.8406	623.9521	624.7804	–	–	527.1
SGO	623.87730	624.04935	626.52974	0.363	0.62	1379.0
SGO-L	623.82940	623.89193	624.03170	0.043	0.63	0.0

* Represents the cost saved by the SGO-L algorithm compared to competitor algorithms.

Quality and stability of solutions: From Table 5, it can be observed that the statistical indicators (*MinCost*, *MeanCost*, *MaxCost*) obtained by SGO-L have lower values than those obtained by SGO or other competitor algorithms listed in the table. Thus, SGO-L outperforms both well-known algorithms (such as PSO [13], DE [15], CSA [16], BSA [18], or FA [19]) and algorithms obtained through various hybridization techniques (such as Jaya-SML [20], IJaya [55], GHS [56], IFA [58], MCSA [59], etc.) in terms of cost. Additionally, the stability of SGO-L is excellent (*SD* = 0.043 USD/h), as determined from 50 independent runs. Compared to the competitor algorithms listed in Table 5, SGO-L exhibits better stability than most (CSA [16], GC-Jaya [55], FA [19], etc.). However, a few algorithms (PSO [13], DE [15], Jaya-SML [20], and IJaya [55]) demonstrate better stability than SGO-L, but their solution quality in terms of cost is lower.

Computation time: The average computation time achieved by SGO and SGO-L is excellent (approximately 0.6 s/rulare), which is lower than the majority of computation times obtained by competitor algorithms. Exceptions include the BSA [18] and M_DE [57] algorithms, which have slightly better computation times but worse cost indicators (*MinCost*, *MeanCost*, and *MaxCost*).

6.5. Results for Case C_4 (40 Unit)

The best generation scheduling results identified by the SGO and SGO-L algorithms are presented in Table 6. In cases C1-C3, which analyze small- and medium-sized systems, the best solutions obtained by SGO and SGO-L are very close (cases C_1 and C_2) or similar (case C_3); in case C_4 (which studies a large-scale system with 40 units with VPE), the solutions and operating costs identified by the two algorithms differ significantly ($MinCost_{SGO}$ = 121,492.1621 USD/h and $MinCost_{SGO\text{-}L}$ = 121,420.9436 USD/h). The accuracy of satisfying the power balance is very high for both algorithms, with *Accuracy* $< 2 \times 10^{-11}$ MW. Additionally, the best solution obtained by SGO-L is better than the solutions identified by SGO or other competitor algorithms listed in Table 7 (except the ESNS algorithm [3], which achieves a slightly lower minimum cost).

Table 6. Generation schedule obtained by SGO and SGO-L for case C_4 (40 units, P_D = 10,500 MW).

P **(MW)**	**SGO**	**SGO-L**	***P*** **(MW)**	**SGO**	**SGO-L**
P_1	111.210100	110.818159	P_{21}	523.279409	523.279671
P_2	111.079577	110.878681	P_{22}	523.280011	523.279576
P_3	97.400703	97.401031	P_{23}	523.279465	523.279795
P_4	179.734599	179.733223	P_{24}	523.280329	523.279903
P_5	96.998256	97.000000	P_{25}	523.279779	523.279785
P_6	139.999988	139.999990	P_{26}	523.280280	523.279828
P_7	299.999996	259.601664	P_{27}	10.000147	10.000054
P_8	284.602096	284.600557	P_{28}	10.000000	10.000101
P_9	284.599953	284.602673	P_{29}	10.000017	10.000126
P_{10}	130.000001	130.000232	P_{30}	96.999895	87.845007
P_{11}	168.799996	168.800477	P_{31}	189.999999	190.000000
P_{12}	168.800277	94.000000	P_{32}	189.999974	189.999940
P_{13}	214.760088	214.759819	P_{33}	189.999999	189.999999
P_{14}	394.279536	394.279407	P_{34}	199.999964	164.803470
P_{15}	304.519891	394.279334	P_{35}	164.803367	200.000000
P_{16}	304.519402	304.519576	P_{36}	164.815650	199.999997
P_{17}	489.279336	489.279733	P_{37}	109.999930	109.999950
P_{18}	489.279323	489.279966	P_{38}	109.999999	109.999914
P_{19}	511.279646	511.279494	P_{39}	109.999945	109.999980
P_{20}	511.279505	511.279487	P_{40}	511.279574	511.279404
	Accuracy (MW)			1.82×10^{-11}	1.80×10^{-11}
	MinCost F (USD/h)			121,492.1621	121,420.9436

Table 7. Values of the statistical items obtained by different algorithms for case C_4 (40 units, P_D = 10,500 MW).

Algorithm	***MinCost*** **(USD/h)**	***MeanCost*** **(USD/h)**	***MaxCost*** **(USD/h)**	***SD*** **(USD/h)**	***Time*** **(s)**	***NE***	**Cost Saving (USD/yr)**
SOA [60]	125,704.4	128,066.9	127,019.5	667.16	110.32	1,500,000	57,188,108.6
BWO [60]	122,875.8	123,858.9	123,398.6	240.70	45.88	1,500,000	20,326,028.6
MIMO [26]	122,758.7	124,621.8	126,059.2	866.20	–	400,000	27,009,032.6
GWO [3]	121,951.3	122,171.9	122,965.9	299.97	37.25	500,000	5,547,908.6
SDO [23]	121,750.2	122,460.1	123,222.7	405.01	–	4,000,000	8,072,540.6
SMA [61]	121,658.66	–	–	–	–	25,000	–
ESCSDO [23]	121,626.97	122,351.7	123,128.9	412.29	–	4,000,000	7,122,956.6
EBWO [60]	121,600.9	122,180.9	122,012.6	163.42	98.69	1,500,000	5,626,748.6
SCA [62]	121,506.58	121,857.90	122,056.15	347.26	–	30,000	2,797,268.6
TLABC [63]	121,468.38	121,739.44	122,192.32	160.88	–	200,000	1,759,559.0
CSO [64]	121,465.99	121,988.48	122,781.75	275.92	19.77	1,000,000	3,941,149.4
SNS [3]	121,465.7	121,640.8	122,267.9	194.86	52.05	500,000	895,472.6
ESNS [3]	121,415.6	121,742.2	122,155.2	206.70	61.44	500,000	1,783,736.6
SGO	121,492.1621	122,017.6763	122,788.3340	274.91	4.02	40,000	4,196,909.0
SGO-L	121,420.9436	121,538.5771	121,731.1530	75.89	4.07	40,000	0.0

Quality and stability of solutions: Table 7 presents the statistical cost indicators (*MinCost*, *MeanCost*, *MaxCost*, and *SD*) obtained by SGO, SGO-L, and other competitor algorithms. The proposed SGO-L algorithm outperforms all competitor algorithms listed in Table 7 in terms of cost indicators. The stability of SGO-L is good ($SD_{SGO\text{-}L}$ = 75.89 USD/h) compared to the stability achieved by other algorithms, where (*SD* ranges between 160.88 USD/h and 866.20 USD/h). Additionally, the number of evaluations *NE* performed by SGO-L is relatively small compared to the *NE* required by other algorithms, where *NE* > 40,000 (with the exceptions of SMA [61] and SCA [62]).

Computation time: The average computation time (item *Time*) achieved by SGO and SGO-L is excellent (approximately 4 s per run), being better than the algorithms listed in Table 7.

6.6. Results for Case C_5 (110 Unit)

The best solutions for SGO and SGO-L: The best solutions obtained using the SGO and SGO-L algorithms are shown in Table 8. The minimum cost achieved by SGO-L ($MinCost_{\text{SGO-L}}$ = 197,988.1782 USD/h) is more competitive than that identified by SGO ($MinCost_{\text{SGO}}$ = 197,990.3133 USD/h). The accuracy of satisfying the power balance for the best solutions identified by both SGO and SGO-L is very high, with *accuracy* $< 2 \times 10^{-11}$ MW. Additionally, the best solution obtained by SGO-L is better than the solutions identified by SGO or other competitor algorithms listed in Table 9.

Table 8. Generation schedule obtained by SGO and SGO-L for case C_5 (110 units, P_D = 15,000 MW).

***P* (MW)**	**SGO**	**SGO-L**	***P* (MW)**	**SGO**	**SGO-L**
P_1	2.400007	2.4	P_{56}	25.316376	25.200031
P_2	2.400000	2.4	P_{57}	25.221640	25.2
P_3	2.404759	2.4	P_{58}	35.090308	35.000000
P_4	2.400002	2.4	P_{59}	35.084439	35
P_5	2.402293	2.4	P_{60}	45.001934	45
P_6	4.000026	4	P_{61}	45.000201	45
P_7	4.000479	4	P_{62}	45.001104	45
P_8	4.000602	4	P_{63}	184.956082	184.999998
P_9	4.000215	4	P_{64}	184.961643	184.999704
P_{10}	63.109427	64.381310	P_{65}	184.991783	185
P_{11}	62.889623	62.117647	P_{66}	184.999993	185
P_{12}	37.255163	36.300229	P_{67}	70.002750	70
P_{13}	54.805006	56.679235	P_{68}	70.000385	70
P_{14}	25.000000	25	P_{69}	70.002879	70
P_{15}	25.011983	25	P_{70}	359.989844	360
P_{16}	25.000227	25	P_{71}	399.998426	400
P_{17}	154.996375	155	P_{72}	399.995372	400
P_{18}	154.986712	155	P_{73}	103.151748	104.973142
P_{19}	154.999200	155	P_{74}	192.225926	191.507362
P_{20}	154.981305	154.999998	P_{75}	89.953594	90
P_{21}	68.900222	69	P_{76}	49.998792	50
P_{22}	68.900000	69	P_{77}	160.140507	160
P_{23}	68.901006	69	P_{78}	302.074815	295.761286
P_{24}	349.999869	350	P_{79}	174.414560	174.947921
P_{25}	399.995712	400	P_{80}	99.264355	98.085804
P_{26}	399.996573	400	P_{81}	10.000516	10
P_{27}	499.994615	499.999980	P_{82}	12.029907	12.000095
P_{28}	499.988010	500	P_{83}	20.644153	20.000022
P_{29}	199.962997	200	P_{84}	199.980093	200
P_{30}	99.997154	100	P_{85}	324.501508	325
P_{31}	10.000162	10	P_{86}	439.981804	440
P_{32}	19.997876	20	P_{87}	11.069768	14.294975
P_{33}	79.762445	80	P_{88}	29.609869	24.272118
P_{34}	249.999857	250	P_{89}	79.608108	82.429122

Table 8. *Cont.*

P (MW)	SGO	SGO-L	*P* (MW)	SGO	SGO-L
P_{35}	359.924170	360	P_{90}	89.611089	89.368521
P_{36}	399.997151	399.999997	P_{91}	57.588706	57.641984
P_{37}	39.970913	39.999931	P_{92}	98.815536	99.956986
P_{38}	69.935322	70	P_{93}	439.995628	440
P_{39}	99.869878	99.999995	P_{94}	499.997365	500
P_{40}	120.000000	120	P_{95}	599.981214	600
P_{41}	156.666734	157.254488	P_{96}	470.499292	471.474016
P_{42}	219.996854	219.999944	P_{97}	3.600287	3.6
P_{43}	439.999940	440	P_{98}	3.600060	3.6
P_{44}	559.998228	560	P_{99}	4.404941	4.4
P_{45}	659.995985	660	P_{100}	4.405880	4.400000
P_{46}	615.389468	616.454159	P_{101}	10.029742	10.000000
P_{47}	5.400727	5.400000	P_{102}	10.001742	10
P_{48}	5.400256	5.4	P_{103}	20.004665	20
P_{49}	8.400020	8.4	P_{104}	20.000019	20
P_{50}	8.401530	8.4	P_{105}	40.000143	40
P_{51}	8.400557	8.4	P_{106}	40.000241	40
P_{52}	12.000014	12	P_{107}	50.000506	50
P_{53}	12.005638	12	P_{108}	30.002929	30
P_{54}	12.000763	12	P_{109}	40.000063	40
P_{55}	12.000695	12	P_{110}	20.000019	20
	Accuracy (MW)			-1.82×10^{-11}	2.41×10^{-14}
	MinCost F (USD/h)			197,990.3133	197,988.1782

Table 9. Values of the statistical items obtained by different algorithms for case C_5 (110 units, P_D = 15,000 MW).

Algorithm	*MinCost* (USD/h)	*MeanCost* (USD/h)	*MaxCost* (USD/h)	*SD* (USD/h)	*Time* (s)	*NE*	Cost Saving (USD/yr)
PSO [31]	200,272.8409	201,483.4217	203,491.7762	662.2335	35.69	1,100,000	30,618,282.25
DE/eig [31]	197,991.2194	197,992.8192	197,995.7383	0.7643	52.98	1,100,000	40,604.35
DE/BBO [31]	197,988.8705	197,989.6066	197,990.5767	0.4201	37.17	1,100,000	12,461.98
CSO [31]	197,993.3466	197,997.0751	198,001.5873	1.8054	28.61	1,100,000	77,886.04
OLCSO [31]	197,988.8576	197,989.5832	197,990.4551	0.3699	33.55	1,100,000	12,256.99
ISMA [22]	198,565.9	198,782.1	198,949.1	153.465	899.75	5,000,000	6,954,704.16
SMA [22]	198,618.5	198,790.5	198,924.3	144.4451	1159.09	5,000,000	7,028,288.16
HHO [22]	219,950.8	222,320.1	225,741.9	2593.779	1106.04	5,000,000	213,147,584.16
JS [22]	213,120.4	213,389.5	213,982.6	356.9868	479.18	5,000,000	134,915,528.16
TSA [22]	200,832.1	202,375.1	203,923.7	927.5175	551.64	5,000,000	38,429,384.16
ESNS [3]	200,266.5	202,925.5	204,908	1669.812	254.54	7,500,000	43,250,888.16
SNS [3]	207,194.8	209,977.1	211,106.9	1063.516	201.80	7,500,000	105,022,904.16
GBO [3]	202,888.6	207,338.9	215,710.8	3248.177	267.08	7,500,000	81,912,272.16
RUN [3]	202,010.7	204,777.8	206,498.3	1205.59	297.06	7,500,000	59,477,036.16
GWO [3]	201,505.7	202,480.1	203,869.7	598.0938	144.26	7,500,000	39,349,184.16
SOS [35]	201,034.1008	-	-	-	-	5,000,000	-
DSOS [35]	198,007.60256	-	-	-	-	5,000,000	-
TFWO [36]	197,988.1790	197,988.1823	197,988.1904	0.0068	46.22	320,000	≈0
SGO	197,990.3133	197,992.1061	197,994.4319	1.04	4.95	21,000	34,357.60
SGO-L	197,988.1782	197,988.1840	197,988.1950	0.0048	5.21	21,000	0.00

Quality and stability of solutions: From Table 9, the performance of the proposed SGO-L algorithm (for the 110-unit system) can be observed in comparison to SGO and other competing algorithms in terms of cost (*MinCost*, *MeanCost*, *MaxCost*, and *SD*) and execution time (item *Time*).

The proposed SGO-L algorithm outperforms all competitor algorithms listed in Table 9 in terms of cost and execution time (except for the TFWO algorithm [36], which achieves slightly

better values for *MeanCost* and *MaxCost*. Notably, SGO-L demonstrates superior capabilities in extracting high-quality and highly stable solutions compared to well-known algorithms (such as PSO [31], CSO [31], SMA [22], HHO [22], SNS [3], GWO [3], and SOS [35]) and improved versions of these algorithms (e.g., OLCSO [31], ISMA [22], ESNS [3], DSOS [35]).

The stability of SGO-L is very good ($SD_{SGO\text{-}L}$ = 0.0048 USD/h) compared to the stability achieved by the majority of algorithms in Table 9, for which the *SD* ranges between 144.44 USD/h and 3248.17 USD/h. A few algorithms exhibit good stability (*SD* < 1 USD/h), but their cost indicators are higher than those of SGO-L (with the exception of TFWO [36]). Additionally, the number of evaluations (*NE*) performed by SGO and SGO-L is significantly smaller ($NE_{SGO\text{-}L}$ = 21,000) compared to the *NE* (>320,000) required by other algorithms.

Computation time: The average computation time (item *Time*) achieved by SGO and SGO-L is excellent (approximately 5 s per run), being significantly better than the algorithms listed in Table 9. The results highlighted in this paragraph emphasize the potential of the SGO-L algorithm for solving the ED problem in large-scale systems.

6.7. Comparison of SGO-L with SGO and Other Algorithms

SGO versus *SGO-L*: In all studied cases (C_1–C_5), the statistical cost indicators (*MinCost*, *MeanCost*, *MaxCost*, and *SD*) obtained by SGO-L are superior to those obtained by SGO (see Tables 2, 3, 5, 7 and 9 corresponding to cases C_1, C_2, C_3, C_4, and C_5). However, to demonstrate the positive influence of including the Laplace distribution in the solution update equations of SGO-L and to show that SGO-L achieves statistically significantly better results than SGO, a comparison of the two algorithms is performed using the non-parametric Wilcoxon statistical test. The Wilcoxon test is applied with a significance level of 1%, and the number of value pairs being compared is equal to 50. Practically, each algorithm (SGO or SGO-L) performs 50 runs for each case study. The Wilcoxon test is applied for two related samples using the *SPSS*® software package, version 16 (Statistical Package for the Social Sciences).

Table 10 presents the *p*-value items resulting from the application of the Wilcoxon test for comparing the value pairs corresponding to the SGO and SGO-L algorithms. In all cases, the *p*-value is below 0.01, indicating that the costs obtained by SGO-L are significantly better than those obtained by SGO. The test results demonstrate that including the Laplace distribution in SGO-L has positive effects, leading to increased performance of this algorithm compared to SGO.

Table 10. Comparison of SGO and SGO-L by the Wilcoxon test.

Cases	R^-	R^+	z-Values *	*p*-Values	Winner
C_1	26.00	1.00	−6.144	0.000	SGO-L
C_2	26.70	6.67	−5.961	0.000	SGO-L
C_3	27.71	5.60	−5.884	0.000	SGO-L
C_4	26.27	7.00	−6.019	0.000	SGO-L
C_5	25.50	0	−6.154	0.000	SGO-L

R^- and R^+ represent the mean of the negative and positive ranks, respectively. * Based on positive ranks.

Convergence process: The convergence characteristics of the SGO and SGO-L algorithms, as a function of the number of iterations, are shown in Figure 3a–e corresponding to the five studied cases. In all studied cases (C_1–C_5), the convergence of both SGO and SGO-L is rapid. The objective function decreases sharply during the first (15–20)% of the maximum number of iterations, followed by a very slow decrease, and the iterative process stabilizes. From Figure 3a–e, it can be observed that SGO converges slightly faster than SGO-L during the initial iterations (in C_1–C_4 cases) because SGO has good global search capabilities. However, as the iterative process progresses, SGO-L, equipped with the Laplace distribution, performs better

local searches than SGO and ultimately identifies better solutions. The best solutions identified by SGO and SGO-L (*MinCost*) are presented in Tables 2, 3, 5, 7 and 9. It should be noted that the plotted characteristics correspond to the best solutions found in each studied case.

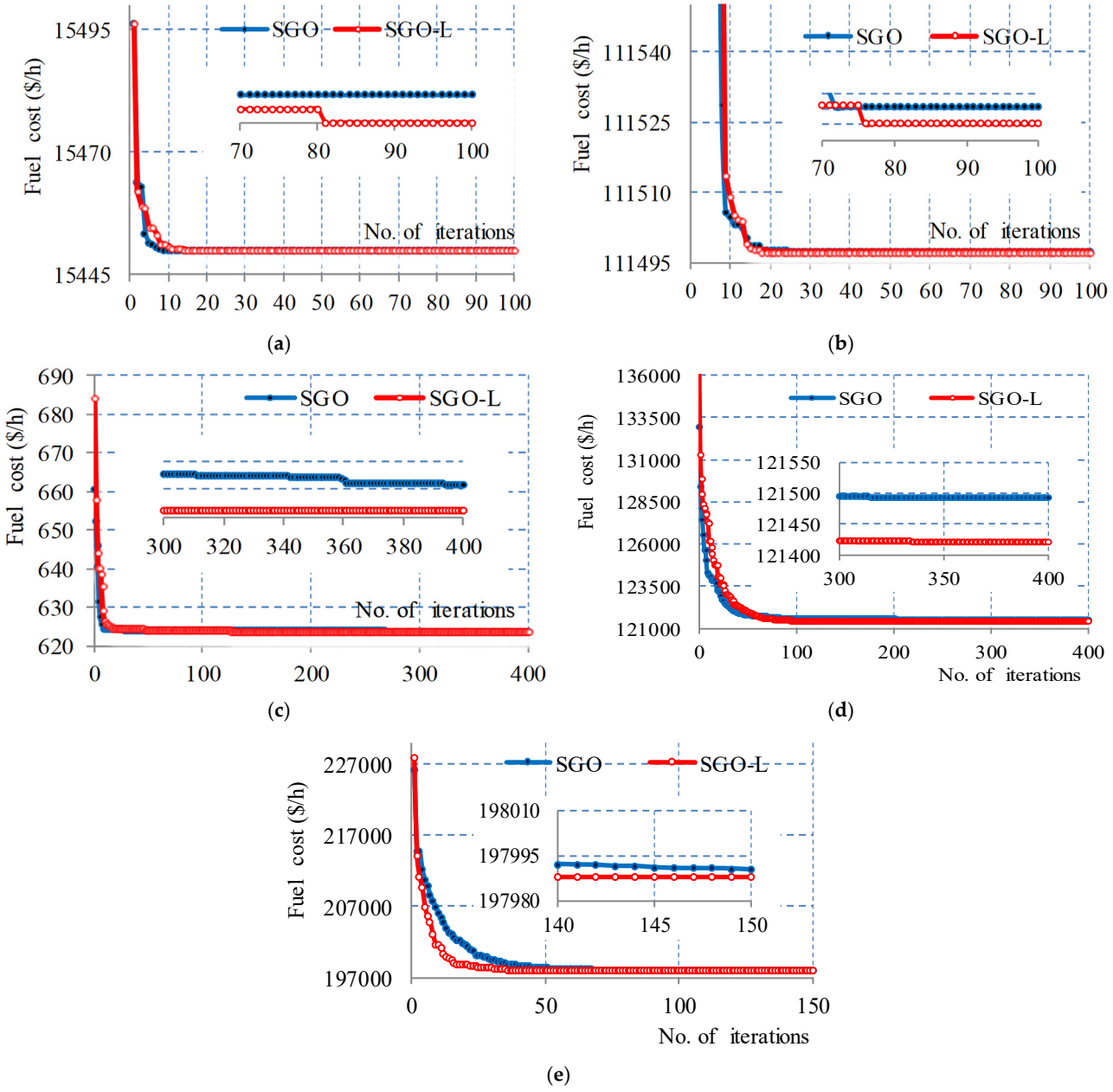


Figure 3. Cost convergence characteristics obtained by SGO and SGO-L: (**a**) for case C_1; (**b**) for case C_2; (**c**) for case C_3; (**d**) for case C_4; (**e**) for case C_5.

To more clearly highlight the superior local search capability of the proposed SGO-L algorithm compared to SGO and the transition from the exploration phase to the exploitation phase, we present the variation in the objective function with the number of evaluations (*NE*) performed (Figure 4a–f) for each case. From Figure 4a–f—the area marked in blue—it can be observed that SGO maintains a high global search capability throughout the iterative process, with the values of the objective function remaining high both in the initial and final iterations. Thus, SGO is unable to focus during each run on a potential area that could later be exploited intensively to extract a near-optimal or optimal solution.

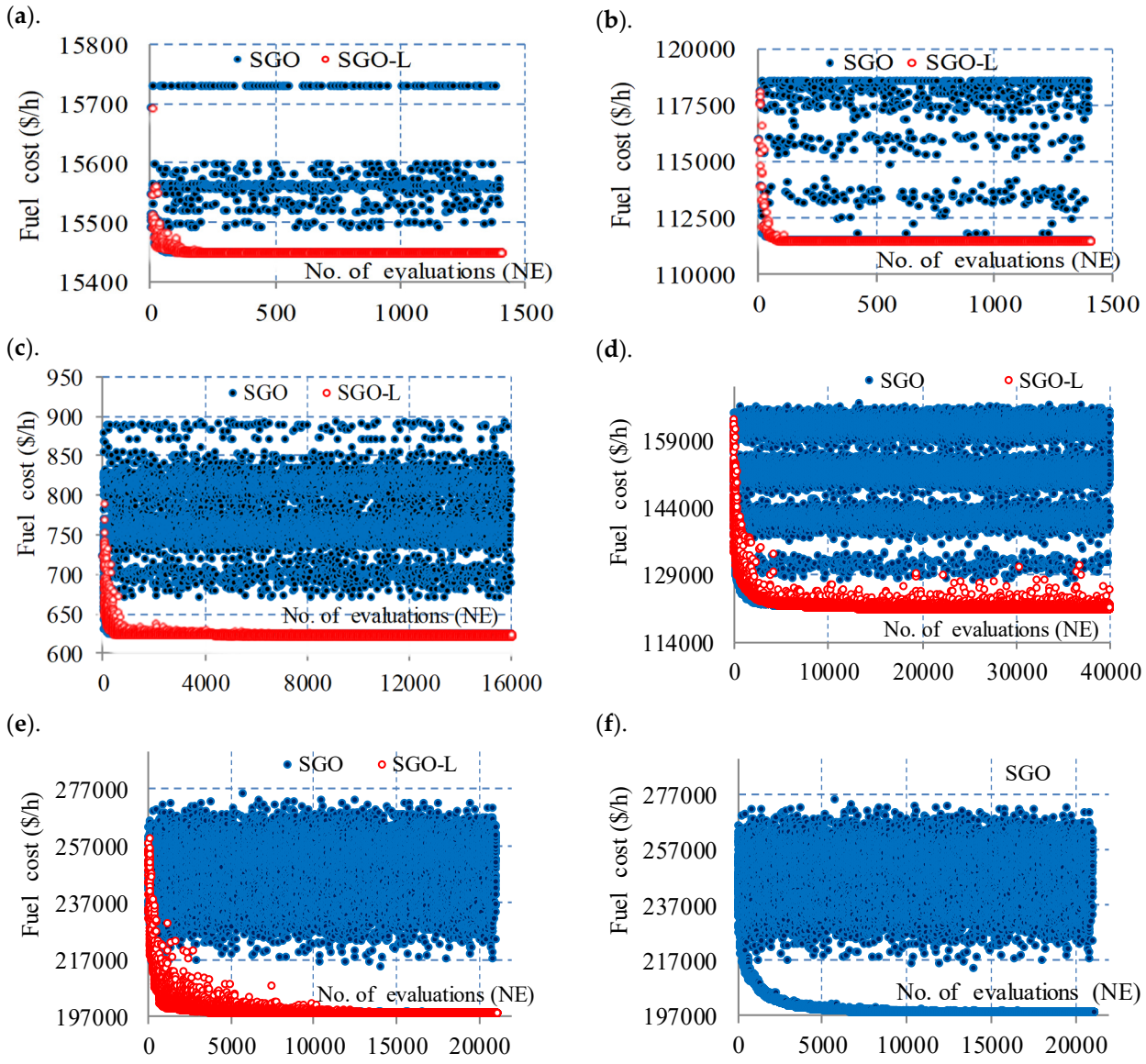


Figure 4. Cost–number of evaluations (*NE*) dependency obtained by SGO and SGO-L: (**a**) for case C_1; (**b**) for case C_2; (**c**) for case C_3; (**d**) for case C_4; (**e**,**f**) for case C_5.

However, it should be noted that SGO does perform an exploitation phase, albeit in a more limited form (in all cases, several solutions are generated around the current best solution—see, for example, Figure 4f, where the lower blue area indicates this observation). Additionally, Figure 4f reveals a large blue area/band that highlights the multitude of high costs generated by solutions potentially that are far from the current best solution. This large blue area in Figure 4f demonstrates that a significant portion of the solutions generated by SGO result in high costs and inefficiencies due to the fact that they provide information with low potential for identifying profitable areas. These blue zones/bands, characteristic of the SGO algorithm, appear in all the studied cases, indicating that SGO does not perform sufficiently intensive local searches. Thus, in case C_1—Figure 4a—three blue areas appear; in C_2—Figure 4b—four distinct blue areas are visible; in C_3—Figure 4c—four distinct areas are observed; in C_4—Figure 4d—five distinct areas are present; and in C_5—Figure 4e—two distinct areas are identified (in each case, the lower blue area—associated with low-cost values—is partially obstructed by the red area corresponding to the costs obtained by the SGO-L algorithm).

In contrast, observing the behavior of SGO-L (the area marked in red in Figure 4a–e), it can be seen that it initially executes successful global searches (exploration phase) and then transitions to an intense local search (exploitation phase) around the current solutions. Thus, for all the studied cases, the SGO-L algorithm can achieve solutions that are very close to or near the optimal solution. From Figure 4a–e (the areas corresponding to the points marked in red), it can be observed that in all cases, the cost values of the solutions generated by SGO-L remain close to the best current solution. As a result, the SGO-L algorithm is characterized by a unique, stable, and narrow or relatively narrow red zone (for case C_4) with a continuous decrease. These features indicate that SGO-L performs an intensive local search throughout the entire iterative process. Practically, SGO-L manages to capture a greater amount of useful information than SGO by translating solutions from the blue zone into the red zone, where the corresponding costs are lower. This is due to the inclusion of values generated by the Laplace distribution in the structure of the solution update equations (values that are limited to a maximum threshold, L_{Max}).

When the parameters of the Laplace distribution are properly tuned, this distribution can generate suitable steps both in the global search phase and in the local search phase, leading to high-quality and stable solutions for small-, medium-, and large-sized ED problems (including convex functions in the case of the 110-unit system). For large-scale ED problems (such as case C_4, 40-unit system), the use of the Laplace distribution in SGO-L results in acceptable solutions. Given the results obtained, the Laplace distribution could be successfully incorporated to enhance the characteristics of other classical or recently proposed metaheuristic algorithms.

Cost saving is calculated based on the mean cost, in USD/year, using the following equation: $CostSaving = (MeanCost_C - MeanCost_{SGO\text{-}L})8760$, where $MeanCost_C$ is the mean cost obtained from 50 runs of a given competitor algorithm C and $MeanCost_{SGO\text{-}L}$ is the mean cost obtained from 50 runs of SGO-L. Tables 2, 3, 5, 7 and 9 present the *Cost Saving* values corresponding to cases C_1, C_2, C_3, C_4, and C_5, respectively. For small-scale systems (case C_1, six-unit system) and medium-scale systems (C_2 and C_3, 10-unit systems), the cost saving is minor (as most competitor algorithms are capable of obtaining solutions relatively close to the optimal solution), with a few exceptions, such as the CBA [17] (in case of C_1), HHO [22] (case C_2), or IGA-MU [14] (in case of C_3). However, in the case of C_4 or C_5, which is a large-scale system, the cost saving is more significant.

Discussion on computation time: The comparison between SGO-L and competitor algorithms in terms of the time metric should be regarded as indicative, as the algorithms were executed on different computing systems or, in some cases, due to the lack of information about these systems. Nevertheless, in most cases, the difference in execution times achieved by SGO-L compared to the competitor algorithms is evident. Additionally, SGO-L requires a significantly smaller number of evaluations (NE) compared to the competitor algorithms (e.g., PSO [12], GA [12], FA [19] in case C1; PSO [22], HHO [22], ISMA [22] in case C2; SOA [60], EBWO [60], ESCSDO [23] in case C4; and all algorithms in case C5). Consequently, the execution time of SGO-L should be shorter than that of the mentioned competitor algorithms provided that the computing systems used have similar characteristics.

7. Conclusions

This article introduces a new variant of the SGO algorithm, denoted as SGO-L, achieved by integrating the Laplace operator into the heuristics implemented in the improving and acquiring phases. The aim is to regulate the search steps in the solution space and enhance the exploration–exploitation balance, positively impacting the quality and stability of the solutions obtained through SGO-L. SGO-L was successfully validated on four systems of small (six units), medium (10 units), and large (40 units) sizes. In all

analyzed cases (C_1–C_4), statistical tests show that SGO-L outperforms the SGO algorithm, achieving higher quality and more stable solutions.

In cases corresponding to small (C_1, six units) and medium (C_2, 10 units) systems, SGO and SGO-L achieve very close values for the *MinCost* indicator, but SGO-L is superior for other indicators (*MeanCost*, *MaxCost*, and *SD*). SGO-L's superiority becomes more evident in larger cases (C_4, 40-unit systems), where the entire set of statistical indicators (*MinCost*, *MeanCost*, *MaxCost*, and *SD*) shows significant improvements over SGO. Moreover, the annual cost savings exceed USD 4 million in the case of C4.

Both algorithms, SGO and SGO-L, achieve excellent average execution times; for instance, in the 40-unit system, the *Time* indicator is below 4.1 s per run compared to other algorithms exceeding 19 s per run. SGO-L demonstrates better average performance and stability than well-known algorithms (e.g., PSO, FA, DE) and modern competitors (e.g., SNS, TLABC, EBWO).

The results confirmed through statistical tests show that equipping SGO-L with the Laplace operator brings significant improvements compared to SGO in solving the ED problem. The main disadvantage of the Laplace operator lies in the need to tune its specific parameters (a, b, and L_{Max}), which can significantly impact the quality of the solutions.

The SGO-L algorithm is characterized by its simplicity, ease of understanding and implementation, rapid convergence, and flexibility for hybridization with other algorithms. Additionally, SGO-L has the ability to identify high-quality and stable solutions for small- and medium-sized systems with non-convex functions and for large-scale systems with convex functions (110 units). However, for large-scale systems with non-convex functions, the performance of SGO-L is moderate in terms of stability, indicating the need for further improvements to the algorithm.

Future work could explore the potential of the Laplace operator within SGO in other forms (as a mutation and/or crossover operator) or in combination with other metaheuristic algorithms. We will also test whether the Laplace operator maintains its efficiency when incorporated into the structure of other metaheuristic algorithms. The new versions obtained can be applied without limitation, for example, in power system optimization (such as multi-area ED, dynamic ED, ED problems with renewable sources, hydrothermal scheduling, etc.), in other fields, or for optimizing complex mathematical functions.

Author Contributions: Conceptualization, D.C.S., C.H. and M.L.S.; software data curation, D.C.S.; investigation, D.C.S., C.H., M.L.S., H.N.H., F.C.D. and E.G.; writing—original draft preparation, D.C.S., M.L.S. and F.C.D.; writing—review and editing, C.H., H.N.H., F.C.D. and E.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the University of Oradea within the Grants Competition "Scientific Research of Excellence Related to Priority Areas with Capitalization through Technology Transfer: INO-TRANSFER-UO", project no. 315/21/12/2021; "Scientific Research of Excellence Related to Priority Areas with Capitalization through Technology Transfer: INO-TRANSFER-UO—2nd Edition", project no. 239/01/11/2022; and the APC was funded by the University of Oradea.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Acknowledgments: The research undertaken was made possible by the equal scientific involvement of all the authors concerned.

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

Nomenclature

a_j, b_j, c_j, e_j, and f_j: fuel cost coefficients of thermal unit j; B_{ij}, B_{0i}, B_{00}: loss coefficients; c: self-introspection parameter from SGO; $f(X_{Best})$: objective function associated with X_{Best}; $f(X_i)$, $f(X_r)$: objective function

associated with X_i and X_r solutions; m: number of thermal units (problem dimension); N: population size; P_D: total power demand; P_L: transmission line losses; P: output power vectors of the thermal units; P_j: power of the thermal unit j; $P_{j,min}$, $P_{j,max}$: real minimum and maximum power of the thermal unit j; k: current iteration; k_{max}: maximum number of iterations; X_i, X_r: m-dimensional vectors associated with solutions i and r; $X_{i,new}$: new solution vector X_i; X_{Best}: vector of the best solution; a, b, L_{Max}: specific parameters of the SGO-L algorithm; $L(u,a,b)$: Laplace operator; $exp(\bullet)$: exponential function; *min*, *max*: minimum and maximum limits of some variables/functions; *Min*, *Max*: mathematical functions that determine the minimum and maximum value in a set. ABC: artificial bee colony; BSA: backtracking search optimization; BWO: beluga whale optimization; CBA: chaotic bat algorithm; CLPSO: comprehensive learning PSO; CPSO: cooperative PSO; CSA: cuckoo search algorithm; CSCA: chaotic SCA; CSO: cuckoo search optimization; DE: differential evolution; DE/eig: DE with eigenvector-based crossover operator; DE/BBO: hybrid DE with biogeography-based optimization; DHS: differential harmony search; DVAC: distance-varying acceleration coefficient; EBWO: enhanced BWO; ED: economic dispatch; ESCSDO: eagle strategy supply–demand-based optimization algorithm with chaotic; ESNS: enhanced SNS; FA: firefly algorithm; FIPS-PSO: fully informed PSO; GA: genetic algorithm; GABC: best-guided ABC; GC-Jaya: Jaya with GCM and DVAC; GCM: Gaussian and Cauchy mutation; GBO: gradient-based optimizer; GHS: global best harmony search algorithm; GWO: gray wolf optimizer; HHO: Harris Hawk optimizer; IFA: improved FA; IGA-MU: improved GA with multiplier updating; IJaya: improved Jaya algorithm; IPOA: improved pelican optimization algorithm; ISMA: improved SMA; Jaya-SML: Jaya algorithm with SAMP and Lévy flights; JS: jellyfish search optimizer; M_DE: momentum DE; MCSA: modified CSA; MIMO: modified ion motion optimization; MSFLA: modified SFLA; MSSA: modified social spider algorithm; OMF: optimization by morphological filter algorithm; POZ: prohibited operating zone; PSO: particle swarm optimization; RUN: Runge–Kutta optimizer; SADE: self-adaptive DE; SAMP: self-adaptive multi-population; SCA: sine–cosine algorithm; SDO: supply-demand-based optimization; SFLA: shuffled frog leaping algorithm; SFLA-GHS: SFLA with GHS; SGO: social group optimization; SGO-L: SGO with Laplace operator; SMA: slime mold algorithm; SNS: social network search; SOA: skill optimization algorithm; SSA: salp swarm algorithm; TLABC: teaching–learning-based artificial bee colony; TSA: tunicate swarm algorithm; VPE: valve-point effect; WOA: whale optimization algorithm.

References

1. Wang, L.; Li, L.P. An effective differential harmony search algorithm for the solving non-convex economic load dispatch problems. *Int. J. Electr. Power Energy Syst.* **2013**, *44*, 832–843. [CrossRef]
2. Mahdi, F.P.; Vasant, P.; Kallimani, V.; Watada, J.; Fai, P.Y.S.; Abdullah-Al-Wadud, M. A holistic review on optimization strategies for combined economic emission dispatch problem. *Renew. Sustain. Energy Rev.* **2018**, *81*, 3006–3020. [CrossRef]
3. Hassan, M.H.; Kamel, S.; Jurado, F.; Desideri, U. Global optimization of economic load dispatch in large scale power systems using an enhanced social network search algorithm. *Electr. Power Energy Syst.* **2024**, *156*, 10971. [CrossRef]
4. Askarzadeh, A.; Gharibi, M. Accurate estimation of cost function parameters for thermal power plants using a novel optimization approach. *Energy Sources Part A: Recovery Util. Environ. Eff.* **2018**, *40*, 2986–2999. [CrossRef]
5. Park, J.-B.; Jeong, Y.-W.; Shin, J.-R.; Lee, K.Y. An improved particle swarm optimization for nonconvex economic dispatch problems. *IEEE Trans. Power Syst.* **2010**, *25*, 155–166.
6. Farag, A.; Al-Baiyat, S.; Cheng, T.C. Economic load dispatch multiobjective optimization procedures using linear programming techniques. *IEEE Trans. Power Syst.* **1995**, *10*, 731–738. [CrossRef]
7. Fan, J.Y.; Zhang, L. Real-time economic dispatch with line flow and emission constraints using quadratic programming. *IEEE Trans. Power Syst.* **1998**, *13*, 320–325. [CrossRef]
8. Zhan, J.P.; Wu, Q.H.; Guo, C.X.; Zhou, X.X. Fast lambda-iteration method for economic dispatch. *IEEE Trans. Power Syst.* **2014**, *29*, 990–991. [CrossRef]
9. Shalini, S.P.; Lakshmi, K. Solution to economic emission dispatch problem using lagrangian relaxation method. In Proceedings of the 2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE), Coimbatore, India, 6–8 March 2014.

10. Chen, S.D.; Chen, J.F. A direct Newton–Raphson economic emission dispatch. *Int. J. Electr. Power Energy Syst.* **2003**, *25*, 411–417. [CrossRef]
11. Zhou, G.; Zhou, Y.; Deng, W.; Yin, S.; Zhang, Y. Advances in teaching–learning-based optimization algorithm: A comprehensive survey (ICIC2022). *Neurocomputing* **2023**, *561*, 126898. [CrossRef]
12. Gaing, Z.-L. Particle swarm optimization to solving the economic dispatch considering the generator constraints. *IEEE Trans. Power Syst.* **2003**, *18*, 1187–1195. [CrossRef]
13. Barisal, A.K. Dynamic search space squeezing strategy based intelligent algorithm solutions to economic dispatch with multiple fuels. *Int. J. Electr. Power Energy Syst.* **2013**, *45*, 50–59. [CrossRef]
14. Chiang, C.-L. Genetic-based algorithm for power economic load dispatch. *IET Gener. Transm. Distrib.* **2007**, *1*, 261–269. [CrossRef]
15. Niu, Q.; Zhou, Z.; Zhang, H.-Y.; Deng, J. An improved quantum-behaved particle swarm optimization method for economic dispatch problems with multiple fuel options and valve-points effects. *Energies* **2012**, *5*, 3655–3673. [CrossRef]
16. Vo, D.N.; Schegner, P.; Ongsakul, W. Cuckoo search algorithm for non-convex economic dispatch. *IET Gener. Transm. Distrib.* **2013**, *7*, 645–654. [CrossRef]
17. Adarsh, B.R.; Raghunathan, T.; Jayabarathi, T.; Yang, X.-S. Economic dispatch using chaotic bat algorithm. *Energy* **2016**, *96*, 666–675. [CrossRef]
18. Modiri-Delshad, M.; Aghay Kaboli, S.H.; Taslimi-Renani, E.; Rahim, N.A. Backtracking search algorithm for solving economic dispatch problems with valve-point effects and multiple fuel options. *Energy* **2016**, *116*, 637–649. [CrossRef]
19. Yang, Y.; Wei, B.; Liu, H.; Zhang, Y.; Zhao, J.; Manla, E. Chaos Firefly Algorithm With Self-Adaptation Mutation Mechanism for Solving Large-Scale Economic Dispatch With Valve-Point Effects and Multiple Fuel Options. *IEEE Access* **2018**, *6*, 45907–45922. [CrossRef]
20. Yu, J.T.; Kim, C.H.; Wadood, A.; Khurshaid, T.; Rhee, S.B. Jaya algorithm with self-adaptive multi-population and Lévy flights for solving economic load dispatch problems. *IEEE Access* **2019**, *7*, 21372–21384. [CrossRef]
21. Guesmi, T.; Farah, A.; Marouani, I.; Alshammari, B.; Abdallah, H.H. Chaotic sine–cosine algorithm for chance-constrained economic emission dispatch problem including wind energy. *IET Renew. Power Gener.* **2020**, *14*, 1808–1821. [CrossRef]
22. Hassan, M.H.; Kamel, S.; Abualigah, L.; Eid, A. Development and application of slime mould algorithm for optimal economic emission dispatch. *Expert Syst. Appl.* **2021**, *182*, 115205. [CrossRef]
23. Hassan, M.H.; Kamel, S.; Eid, A.; Nasrat, L.; Jurado, F.; Elnaggar, M.F. A developed eagle-strategy supply-demand optimizer for solving economic load dispatch problems. *Ain Shams Eng. J.* **2023**, *14*, 102083. [CrossRef]
24. Garmroudi, S.; Kayakutlu, G.; Kayalica, M.O.; Çolak, U. Improved Pelican optimization algorithm for solving load dispatch problems. *Energy* **2024**, *289*, 129811. [CrossRef]
25. Shaban, A.E.; Ismaeel, A.A.K.; Farhan, A.; Said, M.; El-Rifaie, A.M. Growth Optimizer Algorithm for Economic Load Dispatch Problem: Analysis and Evaluation. *Processes* **2024**, *12*, 2593. [CrossRef]
26. Kumar, M.; Dhillon, J.S. A conglomerated ion-motion and crisscross search optimizer for electric power load dispatch. *Appl. Soft Comput.* **2019**, *83*, 105641. [CrossRef]
27. Dong, R.; Sun, L.; Ma, L.; Heidari, A.A.; Zhou, X.; Chen, H. Boosting Kernel Search Optimizer with Slime Mould Foraging Behavior for Combined Economic Emission Dispatch Problems. *J. Bionic Eng.* **2023**, *20*, 2863–2895. [CrossRef]
28. Tai, T.C.; Lee, C.C.; Kuo, C.C. A hybrid grey wolf optimization algorithm using robust learning mechanism for large scale economic load dispatch with vale-point effect. *Appl. Sci. Sci.* **2023**, *13*, 2727. [CrossRef]
29. Kaur, G.; Dhillon, J.S. Economic power generation scheduling exploiting hill-climbed Sine–Cosine algorithm. *Appl. Soft Comput.* **2021**, *111*, 107690. [CrossRef]
30. Fahim, K.E.; De Silva, L.C.; Andiappan, V.; Shezan, S.A.; Yassin, H. A Novel Hybrid Algorithm for Solving Economic Load Dispatch in Power Systems. *Int. J. Energy Res.* **2024**, *2024*, 8420107. [CrossRef]
31. Xiong, G.; Shi, D. Orthogonal learning competitive swarm optimizer for economicdispatch problems. *Appl. Soft Comput.* **2018**, *66*, 134–148. [CrossRef]
32. Hassan, M.H.; Kamel, S.; Selim, A.; Shaheen, A.; Yu, J.; El-Sehiemy, R. Efficient economic operation based on load dispatch of power systems using a leader white shark optimization algorithm. *Neural Comput. Appl.* **2024**, *36*, 10613–10635. [CrossRef]
33. Singh, T. Chaotic slime mould algorithm for economic load dispatch problems. *Appl. Intell.* **2022**, *52*, 15325–15344. [CrossRef]
34. Xiong, C.; Xu, L.; Ma, L.; Hu, P.; Ye, Z.; Sun, J. Research on large-scale clean energy optimal scheduling method based on multi-source data-driven. *Front. Energy Res.* **2024**, *11*, 1230818. [CrossRef]
35. Vedik, B.; Naveen, P.; Shiva, C.K. A novel disruption based symbiotic organisms search to solve economic dispatch. *Evol. Intell.* **2022**, *15*, 255–290. [CrossRef]
36. Ghasemi, M.; Davoudkhani, I.F.; Akbari, E.; Rahimnejad, A.; Ghavidel, S.; Li, L. A novel and effective optimization algorithm for global optimization and its engineering applications: Turbulent flow of water-based optimization (TFWO). *Eng. Appl. Artif. Intell.* **2020**, *92*, 103666. [CrossRef]

37. Satapathy, S.; Naik, A. Social Group Optimization (SGO): A New Population Evolutionary Optimization Technique. *Complex Intell. Syst.* **2016**, *2*, 173–203. [CrossRef]
38. Naik, A.; Satapathy, S. A Comparative Study of Social Group Optimization with a Few Recent Optimization Algorithms. *Complex Intell. Syst.* **2021**, *7*, 249–295. [CrossRef]
39. Reddy, A.K.V.K.; Narayana, K.V.L. A combinatorial social group whale optimization algorithm for numerical and engineering optimization problems. *Appl. Soft Comput. J.* **2021**, *99*, 106903.
40. Naik, A.; Satapathy, S.C.; Abraham, A. Modified social group optimization—A meta-heuristic algorithm to solve short-term hydrothermal scheduling. *Appl. Soft Comput.* **2020**, *95*, 106524. [CrossRef]
41. Jena, J.J.; Satapathy, S.C. A new adaptive tuned social group optimization (SGO) algorithm with sigmoid-adaptive inertia weight for solving engineering design problems. *Multimed Tools Appl.* **2024**, *83*, 3021–3055. [CrossRef]
42. Secui, D.C.; Bendea, G.; Secui, M.L.; Hora, C.; Bendea, C. The Chaotic Social Group Optimization for the Economic Dispatch Problem. *Int. J. Intell. Eng. Syst.* **2021**, *14*, 666–677.
43. Naik, A. Chaotic Social Group Optimization for Structural Engineering Design Problems. *J. Bionic Eng.* **2023**, *20*, 1852–1877. [CrossRef]
44. Deep, K.; Thakur, M. A new crossover operator for real coded genetic algorithms. *Appl. Math. Comput.* **2007**, *188*, 895–911. [CrossRef]
45. Singh, A.; Deep, K. Real coded genetic algorithm operator embedded in Gravitational Search algorithm for continuous optimization. *Int. J. Intell. Syst. Appl.* **2015**, *12*, 1–22. [CrossRef]
46. Yu, H.; Qiao, S.; Heidari, A.A.; Saleh, A.A.; Bi, C.; Mafarja, M.; Cai, Z.; Chan, H. Laplace crossover and random replacement strategy boosted Harris Hawk optimization: Performance optimization and analysis. *J. Comput. Des. Eng.* **2022**, *6*, 66. [CrossRef]
47. Abed-alguni, B.H.; Paul, D.; Hammad, R. Improved Salp swarm algorithm for solving single-objective continuous optimization problems. *Appl. Intell.* **2022**, *52*, 17217–17236. [CrossRef]
48. Yu, S.; Heidari, A.A.; He, C.; Cai, Z.; Althobaiti, M.M.; Mansour, R.F.; Liang, G.; Chen, H. Parameter estimation of static solar photovoltaic models using Laplacian Nelder-Mead hunger games search. Solar. *Energy* **2022**, *242*, 79–104. [CrossRef]
49. Garg, V.; Deep, K.; Alnowibet, K.A.; Mohamed, A.W.; Shokouhifar, M.; Werner, F. LX-BBSCA: Laplacian biogeography-based sine cosine algorithm for structural engineering design optimization. *AIMS Math.* **2023**, *8*, 30610–30638. [CrossRef]
50. Sinha, N.; Chakrabarti, R.; Chattopadhyay, P.K. Evolutionary programming techniques for economic load dispatch. *IEEE Trans. Evol. Comput.* **2003**, *7*, 83–94. [CrossRef]
51. Gaing, Z.-L. Closure to discussion of "particle swarm optimization to solving the economic dispatch considering the generator constraints". *IEEE Trans. Power Syst.* **2004**, *19*, 2122–2123. [CrossRef]
52. Basu, M. Economic environmental dispatch using multi-objective differential evolution. *Appl. Soft Comput.* **2011**, *11*, 2845–2853. [CrossRef]
53. Elsayed, W.T.; Hegazy, Y.G.; Bendary, F.M.; El-Bages, F.M. Modified social spider algorithm for solving the economic dispatch problem. *Eng. Sci. Technol.* **2016**, *19*, 1672–1681. [CrossRef]
54. Zaoui, S.; Belmadani, A. Solution of combined economic and emission dispatch problems of power systems without penalty. *Appl. Artif. Intell.* **2022**, *36*, e1976092. [CrossRef]
55. Chen, C.; Zou, D.; Li, C. Improved Jaya algorithm for economic dispatch considering valve-point effect and multi-fuel options. *IEEE Access* **2020**, *8*, 84981–84995. [CrossRef]
56. Vaisakh, K.; Reddy, A.S. MSFLA/GHS/SFLA-GHS/SDE algorithms for economic dispatch problem considering multiple fuels and valve point loadings. *Appl. Soft Comput. J.* **2013**, *13*, 4281–4291. [CrossRef]
57. Pandit, M.; Srivastava, L.; Sharma, M.; Dubey, H.M.; Panigrahi, B.K. Large scale multi-zone optimal power dispatch using hybrid hierarchical evolution technique. *J. Eng.* **2014**, *13*, 71–80. [CrossRef]
58. Nguyen, T.T.; Quynh, N.V.; Dai, L.V. Improved firefly algorithm: A novel method for optimal operation of thermal generating units. *Complexity* **2018**, *2018*, 7267593. [CrossRef]
59. Nguyen, T.T.; Vo, N.D.; Dinhhoang, B.; Phamhuu, L. Modified cuckoo search algorithm for solving nonconvex economic load dispatch problems. *Adv. Electr. Electron. Eng.* **2016**, *14*, 236–246. [CrossRef]
60. Hassan, M.H.; Kamel, S.; Jurado, F.; Ebeed, M.; Elnaggar, M.F. Economic load dispatch solution of large-scale power systems using an enhanced beluga whale optimizer. *Alex. Eng. J.* **2023**, *72*, 573–591. [CrossRef]
61. Kamboj, V.K.; Kumari, C.L.; Bath, S.K.; Prashar, D.; Rashid, M.; Alshamrani, S.S.; AlGhamdi, A.S. A cost-effective solution for non-convex economic load dispatch problems in power systems using slime mould algorithm. *Sustainability* **2022**, *14*, 2586. [CrossRef]
62. Alghamdi, A.S. Greedy sine-cosine non-hierarchical grey wolf optimizer for solving non-convex economic load dispatch problems. *Energies* **2022**, *15*, 3904. [CrossRef]

63. Xu, S.; Xiong, G.; Mohamed, A.W.; Bouchekara, H.R.E.H. Forgetting velocity based improved comprehensive learning particle swarm optimization for non-convex economic dispatch problems with valve-point effects and multi-fuel options. *Energy* **2022**, *256*, 124511. [CrossRef]
64. Yu, J.-T.; Kim, C.-H.; Rhee, S.-B. Clustering cuckoo search optimization for economic load dispatch problem. *Neural Comput. Appl.* **2020**, *32*, 16951–16969. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.