processes

MDPI

Article

Enhancing Mean-Variance Mapping Optimization Using Opposite Gradient Method and Interior Point Method for Real Parameter Optimization Problems

Thirachit Saenphon, Suphakant Phimoltares * and Chidchanok Lursinsap

Advanced Virtual and Intelligent Computing Research Center (AVIC), Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University, Bangkok 10330, Thailand
* Correspondence: suphakant.p@chula.ac.th

Abstract: The aim of optimization methods is to identify the best results in the search area. In this research, we focused on a mixture of the interior point method, opposite gradient method, and mean-variance mapping optimization, named IPOG-MVMO, where the solutions can be obtained from the gradient field of the cost function on the constraint manifold. The process was divided into three main phases. In the first phase, the interior point method was applied for local searching. Secondly, the opposite gradient method was used to generate a population of candidate solutions. The last phase involved updating the population according to the mean and variance of the solutions. In the experiments on real parameter optimization problems, three types of functions, which were unimodal, multimodal, and continuous composition functions, were considered and used to compare our proposed method with other meta-heuristics techniques. The results showed that our proposed algorithms outperformed other algorithms in terms of finding the optimal solution.

Keywords: initial population; interior point method; mean-variance mapping optimization; meta-heuristics techniques; opposite gradient method

Citation: Saenphon, T.; Phimoltares, S.; Lursinsap, C. Enhancing Mean-Variance Mapping Optimization Using Opposite Gradient Method and Interior Point Method for Real Parameter Optimization Problems. *Processes* **2023**, *11*, 465. https://doi.org/10.3390/pr11020465

Academic Editors: Jun Wei Lim and Worapon Kiatkittipong

Received: 14 December 2022
Revised: 12 January 2023
Accepted: 15 January 2023
Published: 3 February 2023

1. Introduction

Optimization methods are employed to solve real parameter optimization problems in order to obtain a vector $\boldsymbol{x} = (x_1 \ldots x_D)$ that yields the optimal value of the function $f(\boldsymbol{x})$, where D is the number of dimensions of the vector. To solve optimization problems, a global solution is searched without related knowledge or the physical structure of the cost function. There are real parameter optimization problems, such as engineering optimization problems, as well as those related to scientific applications, energy savings for tissue paper mills involving energy efficiency scheduling [1], the optimal scheduling of vehicle-to-grid energy and ancillary services [2], query optimization mechanisms in cloud computing [3], energy management [4], and tools for analytics in mechanical engineering [5].

Previously, a wide variety of evolutionary algorithms have been proposed to obtain solutions from real-world continuous optimization problems. For instance, the covariance matrix adaptation evolution strategy (CMA-ES) algorithm [6] is based on correlated mutations. The covariance matrix C has been adopted to enable the proper distribution of mutations, resulting in an increased likelihood of the successful replication of the search process. However, there are several black box properties that may promptly lead to premature CMA-ES convergence, with many uncontrollable variations and uncertainties. Therefore, to increase the possibility of finding the global optimum using the restart strategy, many techniques have been proposed based on CMA-ES, such as the restart CMA evolution strategy with increasing population sizes (IPOP-CMA-ES) [7], the bi-population CMA-ES strategy (BIPOP-CMA-ES) [8], and the new bi-population CMA-ES strategy (NBIPOPaCMA-ES) [9]. Usually, a new generation is randomly sampled based on the current covariance matrix in the desired search space. Consequently, the starting point is still randomized. Therefore,

the time taken to find a solution increases as the amount of data increases. Moreover, the size of the search area and the surface of the cost function have also not been considered.

Another technique that solves the optimization problem is mean-variance mapping optimization (MVMO). MVMO [10] involves an evolutionary algorithm with two interesting issues. Firstly, the MVMO process determines the range of search areas for variables within an interval [0, 1]. This process ensures that new values computed later for the optimization population are always generated within this range before the fitness value is calculated. Secondly, the statistical features used in MVMO are useful for changing the search direction. Using the mean and variance of the different solutions and a mapping function for the mutation operation yields the optimal fitness value. Additionally, for each time that the algorithm produces better fitness values, the solutions with the *n*-best fitness values are updated and stored in the archive for use in finding the best solution, until the specified number of iterations has been reached. MVMO is applied in chemical process applications. For example, an adaptive PID controller based on MVMO has been proposed to enhance the performance of a chemical process with variable time delay [11]. Swarm-based mean-variance mapping optimization (MVMO-S) is an extension of MVMO, which is combined with swarm intelligence. The search procedure applied to this technique begins with the particles. They are represented in the form of the consistent function of archiving and mapping to the original MVMO [12]. Another extension of MVMO is the swarm variant of hybrid mean-variance mapping optimization (MVMO-SH) [13–15], which is an evolutionary algorithm. The algorithm takes advantage of the statistical features (mean and variance) of the dynamic search function, using mapping functions for mutation and modification according to the mean and variance of the *n*-best solutions that are recorded in the solution archive. The process of generating new offspring from techniques based on MVMO also uses randomness, regardless of the number of populations in the actual search surface of the cost function.

In addition to the algorithms mentioned above, there are still many algorithms that use natural behavior concepts to solve optimization problems, such as the seagull and gray wolf optimization algorithms. The seagull optimization algorithm (SOA) is a new type of bioinspired optimization algorithm that is based on the characteristics of a seagull. The SOA is combined with another algorithm to solve energy problems, such as short-term wind speed forecasting problems [16]. The gray wolf optimization algorithm (GWO) [17] relies on the level of leadership and the hunting mechanism of the gray wolf population for the search process of the optimization algorithm. The algorithms, based on different animal behavior patterns, are also subject to their effectiveness, which cannot be significantly improved when modified. Valenta and Langer proposed 2D P colonies to model the gray wolf optimization algorithm [18], where the performance was compared with that of the original GWO algorithm. From the computer simulation, the 2D P colonies had good performance for optimization problems.

Seagull, gray wolf optimization, and other evolutionary algorithms have the characteristics of self-management, adaptation, and self-learning. However, the new population determination process requires random initialization, resulting in relatively low computing efficiency. Considering the advantages and disadvantages of these algorithms, facing the same problem, the efficiency can be improved by modifying the original algorithms with other initialization concepts to avoid the randomization process.

Furthermore, a philosophy-inspired algorithm, namely, yin–yang pair optimization (YYOP) [19], uses the concept of balancing between two opposite entities for exploration and exploitation. YYPO is a low-complexity method that maintains good performance.

An important step in an optimization algorithm is the population generation step. Random initialization is used to generate new candidate solutions and eliminate the solutions that yield low scores, so that the cost value can either be maximized or minimized. However, the geometric structure of the cost function is not assessed at any step of the solution when generating offspring or for the search process. We proposed the application of the opposite gradient search [20] technique to search for the solution on the surface of

the cost function. The technique, namely the fast opposite gradient search (FOGS) method, refers to a surface search algorithm applied with the opposite gradient method (OGM) [21]. FOGS is different from other search methods. It is not based mainly on meta-heuristics; instead, it seeks the manifold to obtain the locations where the cost function has zero gradients and minimum values.

This article introduces methods to increase the problem-solving efficiency with a better initial population for real parameter optimization problems. The proposed method presents a combination of the interior point method, the opposite gradient for generating new offspring, and the mean-variance mapping optimization algorithm (IPOG-MVMO). The mapping optimization algorithm is applied for the mutation operation to generate a modification depending on the n-best solution's mean and variance, to obtain a candidate solution on a complex surface of the function. Therefore, this proposed research contribution involves generating new offspring and obtaining the solution from the high-dimensional space of the cost function.

The manuscript is structured as follows. Section 2 describes the concepts of the proposed technique. In Section 3, we present an experimental simulation and analyze the results. Section 4 presents the discussion. Finally, Section 5 provides the conclusions and scope for future research.

2. Proposed Concept

For the proposed method, there were three main phases. Firstly, IPM was applied for local searching. The results from the first phase were introduced as points used for the second phase to create a new population, which depended on the manifold of the cost function using OGM. Finally, the last phase aimed to obtain the best solution using MVMO.

Karmarkar [22] presented the interior point method (IPM) to solve a linear programming problem in polynomial time complexity. The number of iterations taken by IPM was not greater than 100, regardless of the size of the problem. Therefore, IPM is suitable for local searching for large problems. The repetitive path pattern of searching for IPM always walks within the domain of possible answers. Since each iteration of IPM is complicated, especially in cases where the coefficient matrix is dense, IPM spends more time on each iteration.

One important concept that has been used to find the best solution or enhance the IPM is the opposite gradient method (OGM). The OGM is used to create a new set of points from the point resulting from the IPM. MVMO then uses the new points from the OGM phase to mutate to achieve a better solution, instead of randomly generating points as the original MVMO. This proposed technique, based on a combination of IPM, OGM, and MVMO, is called IPOG-MVMO and yields a set of solutions throughout the surface of the cost function. The process of IPOG-MVMO is presented in Algorithm 1.

Algorithm 1 Proposed IPOG-MVMO algorithm

Initialize parameters.
Generate an initial population using IPM from randomized points.
Improve the population using OGM.
Update the archive using MVMO.
Return the best point in the archive.

2.1. Generating Initial Population Using Local Minima Obtained from the IPM

The proposed algorithm focused on the interior point method called the barrier method [23,24], because of the proof of convergence and its complexity. The IPM was used to find the local minima that are located near an initial population. A function $F(\boldsymbol{x})$ was defined and was differentiable in the neighborhood of a point $\boldsymbol{x}$. The process involved formulating a rule to test a differentiable function for the local minima that satisfied the restrictions. The procedure is described in Algorithm 2, where the output of this algorithm is assigned as the initial population for the next phase.

Algorithm 2 Finding the local minima using IPM

Q: an archive storing a population
D: number of dimensions in search space
Randomize N points in the feasible domain to generate an initial population.
Set Q as an empty set.
Set parameters $t := t_0 > 0,\ \mu > 1,\ \epsilon > 0$.
FOR $\{1 \leq j \leq N\}$
Set a random point $\boldsymbol{x}_j$ as a starting point.
WHILE ($\frac{D}{t} > \epsilon$) do
    Compute $\boldsymbol{x}_j^*$ by minimizing $tF(\boldsymbol{x}) + \varnothing(\boldsymbol{x})$, subject to $A\boldsymbol{x}_j = b$.
    Set $\boldsymbol{x}_j := \boldsymbol{x}_j^*$.
    Update new t as μt.
ENDWHILE
Insert $\boldsymbol{x}_j$ into Q.
ENDFOR
RETURN Q as an archive storing the initial population for the next phase.

According to Algorithm 2, there are three constants, $t_0 > 0$, $\mu > 1$, $\epsilon > 0$, to be defined to initiate the algorithm. In each iteration, $\boldsymbol{x}_j^*$ is computed by minimizing $tF(\boldsymbol{x}) + \varnothing(\boldsymbol{x})$ subject to constraint $A\boldsymbol{x}_j = b$ and $\varnothing(\boldsymbol{x}) = -\sum_{i=1}^{M} \log(b_i - a_i\boldsymbol{x})$. The solution is updated until $t \geq D/\epsilon$. All local minima in a set Q are used as the initial population for the next step.

2.2. Opposite Gradient Method (OGM)

This section briefly presents a concept for generating new offspring. For a vector in a D-dimensional space of the cost function, new offspring are generated at the position of the manifold by OGM, where its first derivative is zero. One of these locations is expected to be the best solution for the cost function. The proposed technique proceeded along the same line with the following observation. The first derivative of $F(\boldsymbol{x})$ is denoted by $F'(\boldsymbol{x})$ and the gradient vector field is represented as the matrix of the first derivative form. Let $\boldsymbol{x}^{(\sigma)}$ and $\boldsymbol{x}^{(\gamma)}$ be two vectors on the manifold of the cost function. The locations with zero values of the first derivatives on dimension i could be in the range between $\boldsymbol{x}^{(\sigma)}$ and $\boldsymbol{x}^{(\gamma)}$ such that $F_i\left(\boldsymbol{x}^{(\sigma)}\right) F_i\left(\boldsymbol{x}^{(\gamma)}\right) < 0$. If $\left|F_i\left(\boldsymbol{x}^{(\sigma)}\right)\right| > \left|F_i\left(\boldsymbol{x}^{(\gamma)}\right)\right|$, and the location with zero gradient is closer to $\boldsymbol{x}^{(\sigma)}$ than $\boldsymbol{x}^{(\gamma)}$. On the other hand, if $\left|F_i\left(\boldsymbol{x}^{(\sigma)}\right)\right| < \left|F_i\left(\boldsymbol{x}^{(\gamma)}\right)\right|$, the location with zero gradient is closer to $\boldsymbol{x}^{(\gamma)}$ than $\boldsymbol{x}^{(\sigma)}$. This concept can also be applied to any other dimensions.

In particular, for each pair of two vectors obtained from set Q, the output of IPM in Algorithm 2, new next-generation vectors are generated from a varying jumping distance δ related to two vectors that have different signs of gradient on each dimension, as shown in Equation (1).

$$\delta = \frac{\left|F_i\left(\boldsymbol{x}^{(\gamma)}\right)\right|}{\left|F_i\left(\boldsymbol{x}^{(\sigma)}\right)\right| + \left|F_i\left(\boldsymbol{x}^{(\gamma)}\right)\right|} \|\boldsymbol{x}^{(\gamma)} - \boldsymbol{x}^{(\sigma)}\| w \tag{1}$$

Suppose that $\boldsymbol{x}^{(\sigma)}$ and $\boldsymbol{x}^{(\gamma)}$ are selected from the output IPM with the assumption that $F_i\left(\boldsymbol{x}^{(\sigma)}\right) F_i\left(\boldsymbol{x}^{(\gamma)}\right) < 0$. Then, we have two new vectors, $\boldsymbol{x}_{new}^{(\sigma)}$, $\boldsymbol{x}_{new}^{(\gamma)}$, whose coefficients on dimension i are calculated from this jumping distance, as shown in Equations (2) and (3), while the coefficients on the other dimensions are still unchanged.

$$x_{new,i}^{(\sigma)} = x_i^{(\sigma)} + \delta \tag{2}$$

$$x_{new,i}^{(\gamma)} = x_i^{(\gamma)} - \delta \tag{3}$$

In Equation (1), a weight value, w, is calculated at each iteration t according to Algorithm 3 to guarantee that the new vectors are generated within the pre-specified range of the search area. In essence, the weight value is in the interval between 0 and 1 and involves the jumping distance, δ, such that the new vectors discovered outside the search space are discarded immediately.

Algorithm 3 Generating candidate vectors $\boldsymbol{x}_{new}^{(\sigma)}$, $\boldsymbol{x}_{new}^{(\gamma)}$ within the pre-specified search space

w: weight value
NP: number of iterations
Q: an archive storing the current population
Set $\mathbf{X}_{cand} = \varnothing$.
w = 0.05w(NP/t)/* Calculating an updated weight value */
Compute a jumping distance δ using Equation (1).
Create two new candidate vectors $\boldsymbol{x}_{new}^{(\sigma)}$, $\boldsymbol{x}_{new}^{(\gamma)}$ using Equations (2) and (3).
IF {$\boldsymbol{x}_{new}^{(\sigma)}$ is in the range}
    Insert $\boldsymbol{x}_{new}^{(\sigma)}$ into $\mathbf{X}_{cand}$.
ELSE
    Discard $\boldsymbol{x}_{new}^{(\sigma)}$.
ENDIF
IF {$\boldsymbol{x}_{new}^{(\gamma)}$ is in the range}
    Insert $\boldsymbol{x}_{new}^{(\gamma)}$ into $\mathbf{X}_{cand}$.
ELSE
    Discard $\boldsymbol{x}_{new}^{(\gamma)}$.
ENDIF
RETURN $\mathbf{X}_{cand}$

Two new vectors can be generated and located approximately between the parent vectors. Accordingly, the next generation of vectors is calculated to decrease the search area. The cost function values of these new vectors must also be within an acceptable range. If some new vectors providing better cost value are attained, these new vectors will replace some vectors in the previous generation that are stored in the archive. Subsequently, for each dimension i, all the vectors in the archive are sorted ascendingly according to their cost values and divided into two different gradient groups, which are G_i^+, a group of vectors with a positive gradient value along dimension i, and G_i^-, a group of vectors with a negative gradient value along dimension i. For each group, a vector that contains the smallest gradient is retrieved such that these vectors are used as a pair of vectors to generate the next candidate vectors in the next generation. These steps are described in Algorithm 4.

The offspring from Algorithm 4 are filled in the archive Q to store the n-best offspring. The offspring index in the archive can be arranged according to the fitness suitability sequence and can be used as a guide for conducting the search. The size of the archive Q is not changed throughout the process. Note that the archive is updated only when the new vector obtains a solution better than the existing solution in the archive.

Algorithm 4 Generating a population using the opposite gradient method (OGM)

Q: an archive storing the initial population obtained from **Algorithm 2**
NP: number of iterations
D: number of dimensions
Set count = 1
WHILE {count $\leq$ NP}
FOR $\{1 \leq i \leq D\}$
  Set $w = 1$
  Partition the population in Q into G_i^+ and G_i^-.
  Select $\boldsymbol{x}^{(\sigma)}$, a vector with smallest gradient from G_i^+.
  Select $\boldsymbol{x}^{(\gamma)}$, a vector with smallest gradient from G_i^-.
  Compute vectors $\boldsymbol{x}_{new}^{(\sigma)}$, $\boldsymbol{x}_{new}^{(\gamma)}$ from $\boldsymbol{x}^{(\sigma)}$ and $\boldsymbol{x}^{(\gamma)}$ using **Algorithm 3**.
  IF $\boldsymbol{x}_{new}^{(\sigma)} \in \mathbf{X}_{cand}$ and $F\left(\boldsymbol{x}_{new}^{(\sigma)}\right)$ is less than the highest cost value from vectors in Q
    Discard the vector with the highest cost value.
    IF $F_i\left(\boldsymbol{x}_{new}^{(\sigma)}\right) > 0$
      Insert $\boldsymbol{x}_{new}^{(\sigma)}$ in G_i^+.
    ELSE
      Insert $\boldsymbol{x}_{new}^{(\sigma)}$ in G_i^-.
    ENDIF
  ENDIF
  IF $\boldsymbol{x}_{new}^{(\gamma)} \in \mathbf{X}_{cand}$ and $F\left(\boldsymbol{x}_{new}^{(\gamma)}\right)$ is less than the highest cost value from vectors in Q
    Discard the vector with the highest cost value.
    IF $F_i\left(\boldsymbol{x}_{new}^{(\gamma)}\right) < 0$
      Insert $\boldsymbol{x}_{new}^{(\gamma)}$ in G_i^-.
    ELSE
      Insert $\boldsymbol{x}_{new}^{(\gamma)}$ in G_i^+.
    ENDIF
  ENDIF
ENDFOR
$Q' = \left(\cup_{i=1}^{D} G_i^+\right) \cup \left(\cup_{i=1}^{D} G_i^-\right)$
Select n best points with respect to the cost values to store in Q.
count++
ENDWHILE
RETURN Q as an archive storing the initial population for the next phase

2.3. Combining IPM, OGM, and MVMO

The combination of IPM and OGM with MVMO is presented in this section. MVMO carries out global searching and focuses on the best solution. MVMO continuously updates an archive using compact memory with a fixed storage space. The n-best offspring are stored in the archive and serve as a guide toward the search direction. A solution stored in the archive is replaced by new offspring with a lower cost function value. Once the proposed combination is conducted, the vector with the lowest cost function value represents the optimal solution of the given function.

The search procedure applied to MVMO [25] is initiated with a particular set of points with D dimensions, obtained from Algorithm 4 in this study. MVMO has a key feature in that a mapping function is used for modifying the offspring depending on the specific mean-variance of the solutions collected in an archive. The mean $\overline{x}_i$ and variance v_i of dimension i are calculated once the update of the archive for each dimension is made with Equations (4) and (5), where N is the population size stored in the archive.

$$\overline{x}_i = \frac{1}{N} \sum_{j=1}^{N} x_i(j) \tag{4}$$

$$v_i = \frac{1}{N}\sum_{j=1}^{N}(x_i(j) - \overline{x}_i)^2 \tag{5}$$

The size of the search space depends on the mapping function. Moreover, the shape of the function with respect to $\overline{x}_i$ can be controlled by two shape variables $s_{i,1}$ and $s_{i,2}$ as follows:

$$h(\overline{x}_i, s_{i,1}, s_{i,2}, x) = \overline{x}_i\left(1 - e^{-xs_{i,1}}\right) + (1 - \overline{x}_i)e^{-(1-x)s_{i,2}} \tag{6}$$

The new coefficient of each selected dimension x_i of x is calculated using h values in terms of $h_x, h_1,\ h_0$ as follows:

$$x_i = h_x + (1 - h_1 + h_0)\ x_i^r - h_0 \tag{7}$$

where x_i^r is a number obtained randomly with uniform distribution in the range of [0, 1]. It can be guaranteed that the output of Equation (7) is within the range of [–100, 100]. According to Equation (6), $h_x, h_1,\ h_0$ are calculated using $x = x_i^r$, $x = 1$, and $x = 0$, respectively.

MVMO is capable of searching the global optimum with the best mean values of the solution. Two shape variables $s_{i,1}$ and $s_{i,2}$ are originated from a variable s_i, calculated from a scaling factor f_s and variance v_i to change the shape of the function as follows:

$$s_i = -f_s \ln(v_i) \tag{8}$$

where v_i is initially set to 1 and then recalculated once the update of the archive for each dimension is made using (5). The scaling factor f_s can be used to improve the accuracy when its value is greater than 1. On the other hand, the search is coarsely conducted when the value is less than 1. The factor is initiated for the search procedure with a small value. Subsequently, the size is increased for every iteration to increase the efficiency, as denoted in Equation (9):

$$f_s = f_s^*(1 + rand) \tag{9}$$

where $rand$ is randomized within [0, 1], and f_s^* can be calculated as in Equation (10):

$$f_s^* = f_{s_i}^* + \left(\frac{j}{j_f}\right)^2 \left(f_{s_f}^* - f_{s_i}^*\right) \tag{10}$$

where j and j_f are the current iteration number and the last iteration number, respectively. In this study, the values of $f_{s_i}^*$ and $f_{s_f}^*$ were set to 1 and 25, respectively.

As aforementioned, the shape of the mapping function can be determined, which depends upon the shape variables $s_{i,1}$ and $s_{i,2}$ of x_i. To improve the efficiency of the search process, the proper shape variables can be calculated using Algorithm 5.

The parameter d_i is initially set to 1 and then continuously updated at every iteration using the increment factor Δd calculated from Equation (11).

$$\Delta d = (1 + \Delta d_0 + 2{\cdot}\Delta d_0{\cdot}(rand - 0.5)) \tag{11}$$

This factor leads to the expansion or shrinkage of the value of parameter d_i , resulting in oscillation around the current s_i. The optimal interval of Δd_0 is within [0.01, 0.4].

In essence, the original MVMO encounters two problems, causing a long-duration searching process and difficulty in finding the best solution [10,26]. The first problem is zero variance when solutions stored in the archive are located at the same position. The second problem is that the value of the variance is sometimes outside of the specified range. This research attempted to overcome the limitations of MVMO and improve the flexibility of global searching over the classical MVMO by generating the population with the opposite gradient method before proceeding to MVMO. Algorithm 6 presents the determination of the n-best solutions using MVMO within fewer generations when using the population obtained from OGM.

Algorithm 5 Calculating the shape variables $s_{i,1}$ and $s_{i,2}$ for dimension i at iteration j

Calculate the variance v_i using Equation (5).
Calculate the scaling factor f_s using Equations (9) and (10).
Calculate the variable s_i using Equation (8).
Set $s_{i,1} = s_i$ and $s_{i,2} = s_i$.
IF $s_i > 0$ then
    IF $s_i > d_i$
        Update d_i by multiplying with Δd.
    ELSE
        Update d_i by dividing with Δd.
    ENDIF
    IF $d_i > s_i$
        Set $\alpha = d_i$ and $\beta = s_i$.
    ELSE
        Set $\alpha = s_i$ and $\beta = d_i$.
    ENDIF
    IF $rand < 0.5$
        Set $s_{i,1} = \alpha$ and $s_{i,2} = \beta$.
    ELSE
        Set $s_{i,1} = \beta$ and $s_{i,2} = \alpha$.
    ENDIF
ENDIF
RETURN $s_{i,1}$, $s_{i,2}$, d_i

Algorithm 6 Generating n-best offspring using MVMO

Q: an archive storing the initial population obtained from **Algorithm 4**
NP: number of iterations
D: number of dimensions
Set the parameters $d_i, \Delta d_0, f^*_{s_i}, f^*_{s_f}$.
WHILE {count $\leq$ NP}
    FOR {$1 \leq i \leq D$}
        Calculate $\overline{x}_i$ using Equation (4).
        Apply **Algorithm 5** to obtain two shape variables $s_{i,1}$ and $s_{i,2}$.
        Calculate h_x, h_1, h_0 using Equation (6).
        Generate new x_i using Equation (7).
    ENDFOR
    Calculate the cost value of $\boldsymbol{x}_{new}$.
    IF $F(\boldsymbol{x}_{new})$ is less than the highest cost value from vectors in Q.
        Discard the vector with the highest cost value.
        Insert $\boldsymbol{x}_{new}$ in Q.
    ENDIF
ENDWHILE

3. Experiments

The IPOG-MVMO algorithm and the other algorithms, which were NBIPOPaCMA [9], PVADE [27], and MVMO, were applied to 28 benchmark functions on real-parameter optimization [28] using a computer with a Core i7 2.10 GHz CPU and 6 GB RAM for comparative purposes. All 28 benchmark functions have been described by Liang et al. [29,30]. The functions were composed of three types: unimodal (F1–F5), multimodal (F6–F20), and composition functions (F21–F28). All algorithms began running from the same set of initial points. The global optimal point of each test function was at the origin of the vector space. The error value ε can be calculated by Equation (12):

$$\varepsilon = \left| f_j(\mathbf{0}) - f_j(\boldsymbol{x}^*) \right| \tag{12}$$

where $f_j(\mathbf{0})$ is the cost value at the exact optimal point of the j-th benchmark function and $f_j(\boldsymbol{x}^*)$ is the cost value of the optimal point obtained by a selected algorithm. The error value is rounded to zero if it is less than 0.00000001 or 1×10^{-8}.

3.1. IPOG-MVMO Parameter Set-Up

By using IPOG-MVMO, all functions have been tested with $N = 15D$ points, where D is the number of dimensions. The parameters selected for IPOG-MVMO are presented in Table 1.

Table 1. The parameters of the proposed algorithm experiment.

Parameters	Values
The number of dimensions (D) in each problem	$\{10,\ 30,\ 50\}$
Search range	$[-100, 100]^D$
$f^*_{s_i}$	1
$f^*_{s_f}$	25
d_i	1
Δd_0	0.05
N	$15D$
Size of the archive	15
NP	100,000
Number of experimental trials	50

The parameters used for IPOG-MVMO are presented in Table 1. The number of dimensions (D) in each problem, and the search range, were defined according to the evaluation criteria of CEC2013 benchmark problems. The number of points (N) varied depending on the number of dimensions, to improve the flexibility of the algorithms to deal with more difficult problems. The number of iterations (NP) and the number of experimental trials were defined to achieve reliable results. Moreover, other parameters were successfully tested in the original MVMO process.

3.2. Experimental Results

The best value, worst value, mean, and standard deviation of the error between the cost value of the solution found by an algorithm and the exact cost value were recorded over 50 trials. With these four metrics, the best value was used for the comparison and was the focus of this study. The results for 10, 30, and 50 dimensions are summarized in Tables 2–8, respectively. To clearly illustrate the comparison, only the mean of the error was selected and plotted, as is shown in Figures 1–3.

Table 2. The comparative results of the errors obtained from unimodal functions F1–F5.

F1												
Algorithms	D = 10				D = 30				D = 50			
	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.
NBIPOPaCMA	0	0	0	0	0	1.91×10^{-8}	0	4.30×10^{-8}	0	0	0	0
PVADE	0	0	0	0	0	0	0	0	0	0	0	0
MVMO	0	0	0	0	0	0	0	0	0	0	0	0
IPOG-MVMO	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
F2												
Algorithms	D = 10				D = 30				D = 50			
	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.
NBIPOPaCMA	0	0	0	0	0	3.45×10^{-8}	1.61×10^{-8}	3.78×10^{-9}	0	0	0	0
PVADE	0	6.57×10^{2}	1.39×10^{1}	1.12×10^{2}	1.35×10^{3}	4.23×10^{4}	9.03×10^{3}	7.40×10^{3}	3.55×10^{4}	4.15×10^{5}	2.07×10^{5}	9.44×10^{4}
MVMO	0	0	0	0	5.59×10^{-4}	9.21×10^{-4}	7.65×10^{-4}	9.49×10^{-5}	5.59×10^{-4}	9.21×10^{-4}	7.65×10^{-4}	9.49×10^{-5}
IPOG-MVMO	**0**	**0**	**0**	**0**	**0**	$\mathbf{5.27 \times 10^{-7}}$	**0**	**0**	**0**	**0**	**0**	**0**
F3												
Algorithms	D = 10				D = 30				D = 50			
	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.
NBIPOPaCMA	0	0	0	0	0	7.42×10^{-8}	2.21×10^{-8}	1.68×10^{-9}	1.17×10^{-5}	5.01×10^{0}	1.41×10^{0}	2.19×10^{0}
PVADE	0	4.85×10^{-3}	9.7×10^{-5}	7.15×10^{-4}	0	1.55×10^{3}	4.01×10^{1}	2.31×10^{3}	6.14×10^{5}	2.63×10^{7}	1.89×10^{7}	1.16×10^{7}
MVMO	0	0	0	0	1.00×10^{3}	1.11×10^{7}	7.95×10^{5}	1.81×10^{6}	2.88×10^{-6}	1.29×10^{6}	6.08×10^{4}	2.72×10^{5}
IPOG-MVMO	**0**	**0**	**0**	**0**	**0**	$\mathbf{1.17 \times 10^{-6}}$	$\mathbf{3.65 \times 10^{-7}}$	$\mathbf{4.62 \times 10^{-7}}$	1.75×10^{-5}	6.24×10^{-4}	2.11×10^{-4}	2.23×10^{-4}
F4												
Algorithms	D = 10				D = 30				D = 50			
	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.
NBIPOPaCMA	0	0	0	0	7.75×10^{-8}	1.71×10^{-8}	1.39×10^{-8}	4.81×10^{-8}	7.75×10^{-8}	1.71×10^{-8}	1.39×10^{-8}	4.81×10^{-8}
PVADE	0	2.12×10^{-1}	4.21×10^{-3}	3.13×10^{-2}	5.53×10^{-7}	1.33×10^{-3}	1.93×10^{-4}	3.31×10^{-4}	5.53×10^{-7}	1.33×10^{-3}	1.93×10^{-4}	3.31×10^{-4}
MVMO	0	1.18×10^{-8}	0	0	5.37×10^{-6}	3.91×10^{-3}	4.62×10^{-4}	7.56×10^{-4}	5.37×10^{-6}	3.91×10^{-3}	4.62×10^{-4}	7.56×10^{-4}
IPOG-MVMO	**0**	**0**	**0**	**0**	**0**	$\mathbf{1.42 \times 10^{-6}}$	$\mathbf{3.65 \times 10^{-7}}$	$\mathbf{5.64 \times 10^{-7}}$	**0**	$\mathbf{1.42 \times 10^{-6}}$	$\mathbf{3.65 \times 10^{-7}}$	$\mathbf{5.64 \times 10^{-7}}$
F5												
Algorithms	D = 10				D = 30				D = 50			
	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.
NBIPOPaCMA	0	1.15×10^{-8}	0	0	0	3.34×10^{-8}	1.39×10^{-8}	6.18×10^{-8}	0	0	0	0
PVADE	0	8.13×10^{-5}	1.26×10^{-5}	1.11×10^{-4}	0	3.39×10^{-8}	1.02×10^{-8}	1.14×10^{-8}	1.45×10^{-4}	1.6×10^{-4}	1.02×10^{-3}	1.11×10^{-3}
MVMO	0	4.19×10^{-8}	0	0	1.22×10^{-8}	3.01×10^{-8}	2.34×10^{-8}	2.12×10^{-8}	0	0	0	0
IPOG-MVMO	**0**	**0**	**0**	**0**	**0**	$\mathbf{2.79 \times 10^{-8}}$	$\mathbf{1.02 \times 10^{-8}}$	$\mathbf{1.46 \times 10^{-8}}$	**0**	**0**	**0**	**0**

Table 3. The comparative results of the errors obtained from multimodal functions F6–F9.

F6												
Algorithms	D = 10				D = 30				D = 50			
	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.
NBIPOPaCMA	0	0	0	0	0	2.13×10^{-8}	0	3.96×10^{-9}	0	0	0	0
PVADE	0	7.35×10^{0}	7.81×10^{0}	3.89×10^{0}	0	2.67×10^{1}	4.96×10^{-1}	3.73×10^{0}	2.57×10^{1}	1.48×10^{2}	8.52×10^{1}	3.15×10^{1}
MVMO	0	0	0	0	0	3.91×10^{1}	1.04×10^{1}	9.71×10^{0}	0	4.34×10^{1}	3.05×10^{1}	2.01×10^{1}
IPOG-MVMO	**0**	**0**	**0**	**0**	**0**	$\mathbf{2.81 \times 10^{-8}}$	**0**	$\mathbf{8.11 \times 10^{-9}}$	**0**	$\mathbf{5.76 \times 10^{0}}$	$\mathbf{1.21 \times 10^{0}}$	$\mathbf{2.26 \times 10^{0}}$
F7												
Algorithms	D = 10				D = 30				D = 50			
	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.
NBIPOPaCMA	0	1.62×10^{1}	1.40×10^{2}	4.80×10^{2}	5.67×10^{-3}	5.69×10^{1}	7.01×10^{0}	1.71×10^{1}	3.05×10^{-3}	1.83×10^{1}	3.94×10^{0}	8.05×10^{0}
PVADE	3.11×10^{-5}	1.10×10^{1}	3.02×10^{-1}	1.53×10^{0}	4.21×10^{-1}	2.64×10^{1}	4.60×10^{0}	5.75×10^{0}	3.92×10^{4}	4.41×10^{1}	1.89×10^{1}	9.26×10^{0}
MVMO	6.33×10^{-3}	6.34×10^{-3}	6.13×10^{-3}	0	2.48×10^{0}	1.92×10^{1}	8.17×10^{0}	3.62×10^{0}	1.52×10^{1}	4.91×10^{1}	3.04×10^{1}	7.87×10^{0}
IPOG-MVMO	**0**	**0**	**0**	**0**	$\mathbf{4.22 \times 10^{-5}}$	$\mathbf{4.62 \times 10^{-4}}$	$\mathbf{2.11 \times 10^{-4}}$	$\mathbf{2.22 \times 10^{-4}}$	$\mathbf{1.32 \times 10^{-5}}$	$\mathbf{4.81 \times 10^{-4}}$	$\mathbf{2.21 \times 10^{-4}}$	$\mathbf{2.15 \times 10^{-4}}$
F8												
Algorithms	D = 10				D = 30				D = 50			
	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.
NBIPOPaCMA	3.49×10^{-8}	1.68×10^{-8}	1.04×10^{-8}	8.57×10^{-9}	2.08×10^{1}	2.19×10^{1}	2.13×10^{1}	2.85×10^{1}	7.77×10^{-1}	2.11×10^{1}	5.87×10^{0}	9.20×10^{0}
PVADE	1.92×10^{1}	2.01×10^{1}	2.01×10^{1}	6.11×10^{-2}	2.04×10^{1}	2.14×10^{1}	2.61×10^{1}	4.08×10^{1}	2.10×10^{1}	2.12×10^{1}	2.11×10^{1}	3.49×10^{-2}
MVMO	1.94×10^{1}	2.11×10^{1}	1.98×10^{1}	6.54×10^{-2}	2.08×10^{1}	2.10×10^{1}	2.19×10^{1}	5.41×10^{2}	2.10 x10^{1}	2.12×10^{1}	2.11×10^{1}	3.94×10^{-2}
IPOG-MVMO	**0**	$\mathbf{6.12 \times 10^{-8}}$	$\mathbf{1.54 \times 10^{-7}}$	$\mathbf{1.02 \times 10^{-7}}$	2.08×10^{1}	2.10×10^{1}	2.09×10^{1}	2.37×10^{2}	8.35×10^{2}	2.16×10^{1}	1.96×10^{1}	4.34×10^{0}
F9												
Algorithms	D = 10				D = 30				D = 50			
	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.
NBIPOPaCMA	1.92×10^{-8}	2.47×10^{1}	3.82×10^{0}	9.56×10^{-9}	2.02 x10^{0}	1.78×10^{1}	4.30×10^{0}	4.81×10^{0}	0	7.41×10^{1}	1.50×10^{1}	3.30×10^{0}
PVADE	8.11×10^{-3}	3.36×10^{0}	1.34×10^{0}	9.67×10^{-1}	2.27×10^{1}	3.09×10^{1}	2.74×10^{1}	1.77×10^{0}	1.99×10^{1}	3.34×10^{1}	2.60×10^{1}	3.05×10^{0}
MVMO	4.92×10^{-1}	2.45×10^{0}	8.26×10^{-1}	6.66×10^{-1}	7.11×10^{0}	1.92×10^{1}	1.32×10^{1}	2.65×10^{0}	2.48×10^{1}	4.17×10^{1}	3.33×10^{1}	4.39×10^{0}
IPOG-MVMO	**0**	$\mathbf{9.45 \times 10^{-8}}$	$\mathbf{6.95 \times 10^{-8}}$	$\mathbf{4.45 \times 10^{-8}}$	$\mathbf{7.54 \times 10^{-7}}$	$\mathbf{1.19 \times 10^{-6}}$	$\mathbf{9.57 \times 10^{-7}}$	$\mathbf{2.19 \times 10^{-7}}$	4.82×10^{-2}	6.73×10^{0}	1.64×10^{0}	2.57×10^{00}

Table 4. The comparative results of the errors obtained from multimodal functions F10–F13.

F10												
Algorithms	D = 10				D = 30				D = 50			
	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.
NBIPOPaCMA	1.02×10^{-8}	6.71×10^{-8}	3.24×10^{-8}	1.84×10^{-9}	1.01×10^{-8}	3.55×10^{-8}	1.94×10^{-8}	8.32×10^{-8}	0	4.69×10^{-2}	9.37×10^{-3}	2.09×10^{-2}
PVADE	0	1.79×10^{-1}	5.01×10^{-2}	3.79×10^{-2}	1.73×10^{-2}	1.89×10^{-1}	7.64×10^{-2}	3.53×10^{-2}	4.77×10^{-2}	1.15×10^{0}	5.99×10^{-1}	3.42×10^{-1}
MVMO	9.95×10^{-3}	3.58×10^{-2}	1.58×10^{-2}	2.11×10^{-2}	7.40×10^{-3}	7.35×10^{-2}	2.78×10^{-2}	1.64×10^{-2}	0	0	0	0
IPOG-MVMO	**0**	$\mathbf{3.87 \times 10^{-8}}$	$\mathbf{2.89 \times 10^{-8}}$	$\mathbf{7.34 \times 10^{-9}}$	**0**	$\mathbf{4.01 \times 10^{-8}}$	$\mathbf{1.42 \times 10^{-8}}$	$\mathbf{1.78 \times 10^{-8}}$	**0**	$\mathbf{6.66 \times 10^{1}}$	$\mathbf{2.78 \times 10^{-3}}$	$\mathbf{3.91 \times 10^{-2}}$
F11												
Algorithms	D = 10				D = 30				D = 50			
	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.
NBIPOPaCMA	0	6.72×10^{-8}	3.27×10^{-8}	2.11×10^{-9}	9.95×10^{-1}	3.98×10^{0}	2.70×10^{0}	1.25×10^{0}	7.37×10^{-2}	1.89×10^{1}	3.94×10^{0}	8.36×10^{0}
PVADE	0	1.21×10^{1}	4.04×10^{0}	2.33×10^{0}	1.03×10^{0}	7.34×10^{0}	3.13×10^{0}	1.49×10^{0}	6.76×10^{1}	2.34×10^{2}	1.68×10^{2}	4.08×10^{1}
MVMO	0	6.02×10^{0}	2.32×10^{0}	1.29×10^{-2}	1.01×10^{0}	8.05×10^{0}	4.34×10^{0}	1.90×10^{0}	2.59×10^{1}	7.57×10^{1}	4.55×10^{1}	1.22×10^{1}
IPOG-MVMO	**0**	$\mathbf{1.13 \times 10^{-8}}$	$\mathbf{4.87 \times 10^{-8}}$	$\mathbf{3.13 \times 10^{-9}}$	**0**	$\mathbf{1.03 \times 10^{-7}}$	$\mathbf{2.13 \times 10^{-8}}$	$\mathbf{4.03 \times 10^{-8}}$	$\mathbf{1.10 \times 10^{-3}}$	$\mathbf{4.90 \times 10^{-2}}$	$\mathbf{1.21 \times 10^{-2}}$	$\mathbf{1.81 \times 10^{-2}}$
F12												
Algorithms	D = 10				D = 30				D = 50			
	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.
NBIPOPaCMA	0	1.03×10^{0}	3.43×10^{-1}	1.89×10^{-9}	6.70×10^{-8}	3.98×10^{0}	2.27×10^{0}	1.70×10^{0}	7.37×10^{-2}	1.02×10^{0}	3.30×10^{-1}	3.96×10^{-1}
PVADE	9.87×10^{-1}	1.59×10^{1}	5.86×10^{0}	3.68×10^{0}	1.59×10^{0}	3.24×10^{1}	2.13×10^{1}	3.79×10^{0}	2.20×10^{2}	3.12×10^{2}	2.56×10^{2}	2.01×10^{1}
MVMO	1.89×10^{0}	1.55×10^{1}	5.89×10^{0}	1.32×10^{0}	1.59×10^{1}	5.97×10^{1}	3.36×10^{1}	1.14×10^{1}	3.98×10^{1}	1.34×10^{2}	7.77×10^{1}	2.33×10^{1}
IPOG-MVMO	**0**	$\mathbf{3.49 \times 10^{-5}}$	$\mathbf{4.78 \times 10^{-8}}$	$\mathbf{7.56 \times 10^{-9}}$	**0**	$\mathbf{2.72 \times 10^{-8}}$	$\mathbf{8.87 \times 10^{-8}}$	$\mathbf{1.21 \times 10^{-8}}$	$\mathbf{7.74 \times 10^{-5}}$	$\mathbf{2.45 \times 10^{-3}}$	$\mathbf{9.99 \times 10^{4}}$	$\mathbf{1.05 \times 10^{-3}}$
F13												
Algorithms	D = 10				D = 30				D = 50			
	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.
NBIPOPaCMA	0	4.54×10^{0}	7.57×10^{-1}	1.45×10^{0}	0	3.07×10^{0}	2.77×10^{0}	1.36×10^{0}	6.46×10^{-3}	1.48×10^{1}	3.10×10^{0}	6.54×10^{0}
PVADE	0	2.08×10^{1}	8.35×10^{0}	4.34×10^{0}	7.60×10^{1}	1.56×10^{2}	1.21×10^{2}	1.86×10^{1}	8.10×10^{1}	1.62×10^{2}	1.18×10^{2}	1.69×10^{1}
MVMO	1.87×10^{0}	1.82×10^{1}	8.84×10^{0}	9.01×10^{0}	3.06×10^{1}	9.95×10^{1}	5.90×10^{1}	1.67×10^{1}	7.20×10^{1}	2.05×10^{2}	1.33×10^{2}	3.43×10^{1}
IPOG-MVMO	**0**	$\mathbf{3.58 \times 10^{-5}}$	$\mathbf{1.08 \times 10^{-5}}$	$\mathbf{1.01 \times 10^{-5}}$	**0**	$\mathbf{1.79 \times 10^{-6}}$	$\mathbf{3.23 \times 10^{-7}}$	$\mathbf{6.07 \times 10^{-7}}$	$\mathbf{6.43 \times 10^{-4}}$	$\mathbf{6.30 \times 10^{0}}$	$\mathbf{1.54 \times 10^{0}}$	$\mathbf{2.59 \times 10^{0}}$

Table 5. The comparative results of the errors obtained from multimodal functions F14–F17.

F14

Algorithms	D = 10				D = 30				D = 50			
	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.
NBIPOPaCMA	2.13×10^{1}	7.68×10^{2}	3.44×10^{2}	2.11×10^{2}	7.12×10^{2}	1.35×10^{3}	8.70×10^{2}	3.40×10^{2}	3.16×10^{1}	4.59×10^{1}	3.96×10^{1}	5.94×10^{0}
PVADE	4.01×10^{1}	5.13×10^{2}	1.67×10^{2}	1.07×10^{2}	2.04×10^{3}	4.01×10^{3}	3.11×10^{3}	4.70×10^{2}	2.14×10^{3}	3.90×10^{3}	3.07×10^{3}	3.80×10^{2}
MVMO	3.21×10^{0}	2.08×10^{1}	8.01×10^{0}	7.59×10^{0}	9.40×10^{1}	1.66×10^{3}	8.56×10^{2}	4.12×10^{2}	2.19×10^{3}	4.84×10^{3}	3.89×10^{3}	6.02×10^{2}
IPOG-MVMO	$\mathbf{2.03 \times 10^{-5}}$	$\mathbf{2.67 \times 10^{-5}}$	$\mathbf{2.39 \times 10^{-5}}$	$\mathbf{4.78 \times 10^{-6}}$	$\mathbf{2.37 \times 10^{1}}$	$\mathbf{6.35 \times 10^{2}}$	$\mathbf{3.09 \times 10^{2}}$	$\mathbf{3.11 \times 10^{2}}$	$\mathbf{9.23 \times 10^{-5}}$	$\mathbf{6.10 \times 10^{1}}$	$\mathbf{5.05 \times 10^{1}}$	$\mathbf{1.70 \times 10^{0}}$

F15

Algorithms	D = 10				D = 30				D = 50			
	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.
NBIPOPaCMA	6.87×10^{0}	2.38×10^{2}	1.17×10^{2}	8.89×10^{1}	5.53×10^{2}	1.67×10^{3}	7.65×10^{2}	2.85×10^{2}	1.27×10^{-1}	9.68×10^{1}	6.51×10^{1}	3.94×10^{1}
PVADE	3.71×10^{2}	1.02×10^{3}	7.84×10^{2}	1.68×10^{2}	2.13×10^{3}	4.94×10^{3}	3.32×10^{3}	4.08×10^{2}	4.54×10^{3}	6.17×10^{3}	5.36×10^{3}	3.54×10^{2}
MVMO	1.89×10^{2}	6.57×10^{2}	5.58×10^{2}	9.01×10^{1}	1.70×10^{3}	4.45×10^{3}	3.03×10^{3}	5.41×10^{2}	5.75×10^{3}	8.65×10^{3}	6.64×10^{3}	5.92×10^{2}
IPOG-MVMO	$\mathbf{2.58 \times 10^{-5}}$	$\mathbf{7.28 \times 10^{-2}}$	$\mathbf{3.21 \times 10^{-2}}$	$\mathbf{3.13 \times 10^{-2}}$	$\mathbf{2.19 \times 10^{1}}$	$\mathbf{1.22 \times 10^{3}}$	$\mathbf{2.28 \times 10^{2}}$	$\mathbf{2.37 \times 10^{2}}$	$\mathbf{3.69 \times 10^{-3}}$	$\mathbf{6.67 \times 10^{1}}$	$\mathbf{2.03 \times 10^{1}}$	$\mathbf{2.76 \times 10^{1}}$

F16

Algorithms	D = 10				D = 30				D = 50			
	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.
NBIPOPaCMA	0	1.97×10^{1}	5.68×10^{0}	7.23×10^{0}	5.68×10^{-2}	1.23×10^{0}	9.14×10^{-1}	1.86×10^{-1}	1.18×10^{-3}	3.51×10^{0}	1.20×10^{0}	1.65×10^{0}
PVADE	4.75×10^{-1}	1.26×10^{0}	8.93×10^{-1}	1.88×10^{-1}	1.42×10^{0}	3.13×10^{0}	2.32×10^{0}	3.01×10^{-1}	2.67×10^{0}	3.84×10^{0}	3.38×10^{0}	2.86×10^{-1}
MVMO	3.52×10^{-1}	6.49×10^{-1}	5.39×10^{-1}	1.62×10^{-1}	5.37×10^{-1}	1.62×10^{0}	1.08×10^{0}	2.88×10^{-1}	6.60×10^{-1}	1.70×10^{0}	1.14×10^{0}	2.37×10^{-1}
IPOG-MVMO	**0**	$\mathbf{7.64 \times 10^{-1}}$	$\mathbf{1.63 \times 10^{-1}}$	$\mathbf{3.18 \times 10^{-1}}$	8.08×10^{-2}	1.34×10^{0}	5.40×10^{0}	1.09×10^{1}	3.12×10^{-1}	3.25×10^{0}	2.11×10^{0}	2.24×10^{-1}

F17

Algorithms	D = 10				D = 30				D = 50			
	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.
NBIPOPaCMA	0	9.71×10^{1}	1.89×10^{1}	2.64×10^{1}	3.41×10^{1}	4.26×10^{1}	3.53×10^{1}	1.87×10^{0}	2.10×10^{0}	5.55×10^{1}	1.30×10^{1}	2.37×10^{1}
PVADE	1.15×10^{1}	1.75×10^{1}	1.39×10^{1}	1.39×10^{1}	7.44×10^{1}	1.17×10^{2}	9.52×10^{1}	1.10×10^{1}	1.44×10^{2}	3.17×10^{2}	2.37×10^{2}	1.10×10^{1}
MVMO	1.03×10^{1}	1.18×10^{1}	1.11×10^{1}	7.65×10^{-1}	4.38×10^{1}	8.14×10^{1}	6.72×10^{1}	8.50×10^{0}	8.89×10^{1}	1.43×10^{2}	1.11×10^{2}	1.34×10^{1}
IPOG-MVMO	**0**	$\mathbf{3.07 \times 10^{1}}$	$\mathbf{9.85 \times 10^{0}}$	$\mathbf{8.85 \times 10^{0}}$	$\mathbf{3.25 \times 10^{1}}$	$\mathbf{6.60 \times 10^{1}}$	$\mathbf{4.31 \times 10^{1}}$	$\mathbf{4.89 \times 10^{0}}$	4.50×10^{0}	6.66×10^{1}	3.48×10^{1}	3.30×10^{1}

Table 6. The comparative results of the errors obtained from composition functions F18–F20.

F18												
Algorithms	D = 10				D = 30				D = 50			
	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.
NBIPOPaCMA	0	5.74×10^{1}	1.58×10^{1}	1.51×10^{1}	3.86×10^{1}	1.72×10^{2}	6.38×10^{1}	4.45×10^{1}	8.60×10^{-2}	9.86×10^{1}	2.54×10^{1}	4.13×10^{1}
PVADE	1.13×10^{1}	3.26×10^{1}	2.40×10^{1}	3.90×10^{0}	1.40×10^{2}	1.89×10^{2}	1.66×10^{2}	1.12×10^{1}	3.43×10^{2}	4.22×10^{2}	3.65×10^{2}	1.12×10^{1}
MVMO	1.51×10^{1}	2.01×10^{1}	1.72×10^{1}	2.58×10^{0}	4.58×10^{1}	7.88×10^{1}	5.94×10^{1}	7.59×10^{0}	8.06×10^{1}	1.63×10^{2}	1.07×10^{2}	1.73×10^{1}
IPOG-MVMO	**0**	$\mathbf{1.01 \times 10^{-1}}$	$\mathbf{3.27 \times 10^{-2}}$	$\mathbf{3.71 \times 10^{-2}}$	$\mathbf{1.88 \times 10^{1}}$	$\mathbf{3.38 \times 10^{1}}$	$\mathbf{2.94 \times 10^{1}}$	$\mathbf{9.87 \times 10^{0}}$	$\mathbf{1.20 \times 10^{-2}}$	$\mathbf{4.18 \times 10^{0}}$	$\mathbf{1.34 \times 10^{0}}$	$\mathbf{1.92 \times 10^{0}}$
F19												
Algorithms	D = 10				D = 30				D = 50			
	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.
NBIPOPaCMA	0	7.38×10^{-1}	4.34×10^{-1}	2.01×10^{-1}	1.14×10^{0}	2.54×10^{0}	2.40×10^{0}	4.41×10^{-1}	1.60×10^{-1}	2.85×10^{0}	8.65×10^{-1}	1.16×10^{0}
PVADE	3.27×10^{-1}	9.69×10^{-1}	5.98×10^{-1}	1.25×10^{-1}	3.10×10^{0}	1.06×10^{1}	6.49×10^{0}	1.74×10^{0}	1.06×10^{1}	3.10×10^{1}	2.12×10^{1}	4.74×10^{0}
MVMO	3.52×10^{-1}	6.07×10^{-1}	5.19×10^{-1}	1.44×10^{-1}	1.10×10^{0}	3.13×10^{0}	1.93×10^{0}	4.09×10^{-1}	2.79×10^{0}	7.55×10^{0}	4.82×10^{0}	1.24×10^{0}
IPOG-MVMO	**0**	$\mathbf{1.90 \times 10^{-1}}$	$\mathbf{2.59 \times 10^{-2}}$	$\mathbf{6.02 \times 10^{-2}}$	$\mathbf{3.90 \times 10^{-1}}$	$\mathbf{2.88 \times 10^{0}}$	$\mathbf{1.06 \times 10^{0}}$	$\mathbf{4.73 \times 10^{-1}}$	$\mathbf{1.52 \times 10^{-1}}$	$\mathbf{3.28 \times 10^{0}}$	$\mathbf{1.53 \times 10^{0}}$	$\mathbf{1.52 \times 10^{0}}$
F20												
Algorithms	D = 10				D = 30				D = 50			
	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.
NBIPOPaCMA	2.11×10^{0}	4.85×10^{0}	3.26×10^{0}	9.87×10^{-1}	1.01×10^{1}	1.56×10^{1}	1.29×10^{1}	5.98×10^{-1}	7.85×10^{-1}	2.53×10^{1}	7.01×10^{0}	1.05×10^{0}
PVADE	9.96×10^{-1}	3.52×10^{0}	2.12×10^{0}	5.61×10^{-1}	1.04×10^{1}	1.50×10^{1}	1.34×10^{1}	1.89×10^{-1}	1.95×10^{1}	2.55×10^{1}	2.36×10^{1}	1.90×10^{0}
MVMO	1.83×10^{0}	2.78×10^{0}	2.34×10^{0}	4.76×10^{-1}	8.81×10^{0}	1.19×10^{1}	1.04×10^{1}	5.85×10^{-1}	1.87×10^{1}	2.23×10^{1}	2.01×10^{1}	6.84×10^{-1}
IPOG-MVMO	1.12×10^{0}	2.74×10^{0}	1.95×10^{0}	4.01×10^{-1}	$\mathbf{6.84 \times 10^{0}}$	$\mathbf{1.28 \times 10^{1}}$	$\mathbf{1.01 \times 10^{1}}$	$\mathbf{6.23 \times 10^{-1}}$	2.10×10^{0}	1.78×10^{1}	2.30×10^{0}	4.57×10^{-1}

Table 7. The comparative results of the errors obtained from composition functions F21–F24.

F21												
Algorithms	$D = 10$				$D = 30$				$D = 50$			
	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.
NBIPOPaCMA	3.64×10^{0}	1.42×10^{1}	1.05×10^{1}	4.47×10^{0}	2.00×10^{2}	2.00×10^{2}	2.00×10^{2}	0	1.00×10^{2}	2.00×10^{2}	1.91×10^{2}	1.43×10^{1}
PVADE	2.53×10^{2}	3.81×10^{2}	2.90×10^{2}	4.06×10^{1}	2.00×10^{2}	4.43×10^{2}	3.17×10^{2}	6.22×10^{1}	8.36×10^{2}	1.22×10^{3}	9.56×10^{2}	1.44×10^{2}
MVMO	3.35×10^{-5}	3.90×10^{2}	2.26×10^{2}	9.73×10^{1}	2.00×10^{2}	4.43×10^{2}	2.95×10^{2}	1.07×10^{2}	1.00×10^{2}	1.13×10^{3}	2.45×10^{2}	1.91×10^{2}
IPOG-MVMO	$\mathbf{2.68 \times 10^{0}}$	$\mathbf{1.50 \times 10^{1}}$	$\mathbf{1.22 \times 10^{1}}$	$\mathbf{4.11 \times 10^{0}}$	$\mathbf{2.00 \times 10^{2}}$	$\mathbf{4.45 \times 10^{2}}$	$\mathbf{2.57 \times 10^{2}}$	$\mathbf{1.15 \times 10^{1}}$	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{5.24 \times 10^{2}}$	$\mathbf{2.29 \times 10^{2}}$	$\mathbf{3.48 \times 10^{1}}$
F22												
Algorithms	$D = 10$				$D = 30$				$D = 50$			
	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.
NBIPOPaCMA	3.31×10^{0}	1.00×10^{1}	8.81×10^{0}	3.05×10^{0}	3.54×10^{2}	1.08×10^{3}	8.03×10^{2}	3.12×10^{2}	1.89×10^{2}	3.86×10^{3}	1.45×10^{3}	6.01×10^{2}
PVADE	2.34×10^{1}	5.07×10^{2}	2.57×10^{2}	1.11×10^{2}	1.72×10^{3}	3.38×10^{3}	2.49×10^{3}	3.86×10^{2}	6.11×10^{3}	1.01×10^{4}	7.72×10^{3}	8.44×10^{2}
MVMO	5.02×10^{0}	1.00×10^{2}	3.27×10^{1}	1.91×10^{1}	2.35×10^{2}	2.00×10^{3}	8.19×10^{2}	4.28×10^{2}	1.17×10^{3}	4.94×10^{3}	2.76×10^{3}	8.38×10^{2}
IPOG-MVMO	$\mathbf{3.00 \times 10^{0}}$	$\mathbf{1.76 \times 10^{1}}$	$\mathbf{6.20 \times 10^{0}}$	$\mathbf{2.48 \times 10^{2}}$	$\mathbf{2.00 \times 10^{2}}$	$\mathbf{2.83 \times 10^{3}}$	$\mathbf{6.26 \times 10^{2}}$	$\mathbf{2.55 \times 10^{2}}$	3.92×10^{2}	4.68×10^{3}	2.35×10^{3}	6.94×10^{2}
F23												
Algorithms	$D = 10$				$D = 30$				$D = 50$			
	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.
NBIPOPaCMA	3.59×10^{0}	4.14×10^{0}	3.96×10^{0}	2.01×10^{1}	3.52×10^{2}	3.59×10^{3}	1.29×10^{3}	1.35×10^{3}	1.37×10^{3}	1.31×10^{4}	4.29×10^{3}	5.00×10^{3}
PVADE	1.18×10^{1}	1.12×10^{2}	9.14×10^{1}	1.78×10^{1}	4.74×10^{3}	7.25×10^{3}	5.81×10^{3}	5.04×10^{2}	7.91×10^{3}	1.54×10^{4}	1.17×10^{4}	1.48×10^{3}
MVMO	5.63×10^{1}	8.71×10^{2}	4.95×10^{2}	2.03×10^{2}	1.69×10^{3}	4.45×10^{3}	3.09×10^{3}	5.28×10^{2}	6.17×10^{3}	1.12×10^{4}	8.64×10^{3}	1.32×10^{3}
IPOG-MVMO	$\mathbf{3.54 \times 10^{0}}$	$\mathbf{3.14 \times 10^{0}}$	$\mathbf{3.89 \times 10^{0}}$	$\mathbf{3.01 \times 10^{-1}}$	4.89×10^{2}	4.28×10^{3}	2.79×10^{3}	9.41×10^{2}	$\mathbf{4.97 \times 10^{2}}$	$\mathbf{7.29 \times 10^{3}}$	$\mathbf{5.79 \times 10^{3}}$	$\mathbf{9.41 \times 10^{2}}$
F24												
Algorithms	$D = 10$				$D = 30$				$D = 50$			
	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.
NBIPOPaCMA	3.59×10^{0}	3.70×10^{0}	3.74×10^{0}	5.01×10^{-2}	1.39×10^{2}	3.01×10^{2}	2.42×10^{2}	8.00×10^{1}	2.35×10^{2}	3.86×10^{2}	3.27×10^{2}	7.60×10^{1}
PVADE	1.01×10^{2}	2.11×10^{2}	1.94×10^{2}	1.30×10^{1}	2.01×10^{2}	2.61×10^{2}	2.03×10^{2}	1.39×10^{0}	2.40×10^{2}	3.28×10^{2}	2.78×10^{2}	1.83×10^{1}
MVMO	1.01×10^{2}	2.07×10^{2}	1.83×10^{2}	3.74×10^{1}	2.04×10^{2}	2.20×10^{2}	2.11×10^{2}	3.77×10^{0}	2.28×10^{2}	2.61×10^{2}	2.45×10^{2}	8.05×10^{0}
IPOG-MVMO	$\mathbf{9.85 \times 10^{-1}}$	$\mathbf{4.42 \times 10^{0}}$	$\mathbf{2.07 \times 10^{0}}$	$\mathbf{1.44 \times 10^{-2}}$	$\mathbf{1.05 \times 10^{2}}$	$\mathbf{4.62 \times 10^{2}}$	$\mathbf{2.17 \times 10^{2}}$	$\mathbf{1.46 \times 10^{0}}$	$\mathbf{1.05 \times 10^{2}}$	$\mathbf{4.62 \times 10^{2}}$	$\mathbf{2.17 \times 10^{2}}$	$\mathbf{1.46 \times 10^{1}}$

Table 8. The comparative results of the errors obtained from composition functions F25–F28.

F25												
Algorithms	$D = 10$				$D = 30$				$D = 50$			
	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.
NBIPOPaCMA	7.31×10^{0}	7.49×10^{0}	7.44×10^{0}	5.01×10^{-2}	2.05×10^{2}	3.01×10^{2}	2.66×10^{2}	4.26×10^{1}	2.34×10^{2}	3.86×10^{2}	3.23×10^{2}	7.86×10^{1}
PVADE	1.90×10^{2}	2.13×10^{2}	2.04×10^{2}	3.77×10^{0}	2.00×10^{2}	2.56×10^{2}	2.30×10^{2}	2.04×10^{1}	3.17×10^{2}	3.92×10^{2}	3.54×10^{2}	1.72×10^{1}
MVMO	1.01×10^{2}	2.01×10^{2}	1.94×10^{2}	2.25×10^{1}	2.00×10^{2}	2.71×10^{2}	2.51×10^{2}	9.39×10^{0}	3.00×10^{2}	3.50×10^{2}	3.26×10^{2}	1.13×10^{1}
IPOG-MVMO	$\mathbf{7.09 \times 10^{0}}$	$\mathbf{7.57 \times 10^{0}}$	$\mathbf{7.37 \times 10^{0}}$	$\mathbf{4.88 \times 10^{-2}}$	2.09×10^{2}	2.75×10^{2}	2.55×10^{2}	2.84×10^{1}	2.36×10^{2}	2.77×10^{2}	2.57×10^{2}	8.88×10^{1}
F26												
Algorithms	$D = 10$				$D = 30$				$D = 50$			
	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.
NBIPOPaCMA	4.04×10^{0}	4.29×10^{0}	4.14×10^{0}	1.41×10^{-1}	1.55×10^{2}	3.26×10^{2}	2.42×10^{2}	7.92×10^{1}	1.95×10^{2}	4.84×10^{2}	3.43×10^{2}	1.42×10^{2}
PVADE	1.04×10^{2}	2.00×10^{2}	1.84×10^{2}	3.33×10^{1}	2.00×10^{2}	3.11×10^{2}	2.18×10^{2}	4.01×10^{1}	2.00×10^{2}	3.96×10^{2}	3.47×10^{2}	6.01×10^{1}
MVMO	1.02×10^{2}	1.15×10^{2}	1.08×10^{2}	3.52×10^{0}	2.00×10^{2}	2.00×10^{2}	2.00×10^{2}	$4.47 \times 10\text{-}^{3}$	2.00×10^{2}	2.00×10^{2}	2.00×10^{2}	3.21×10^{-3}
IPOG-MVMO	$\mathbf{3.94 \times 10^{0}}$	$\mathbf{4.46 \times 10^{0}}$	$\mathbf{4.12 \times 10^{0}}$	$\mathbf{1.29 \times 10^{-1}}$	1.89×10^{2}	3.11×10^{2}	2.12×10^{2}	2.29×10^{1}	$\mathbf{1.94 \times 10^{2}}$	$\mathbf{2.96 \times 10^{2}}$	$\mathbf{2.12 \times 10^{2}}$	$\mathbf{2.29 \times 10^{1}}$
F27												
Algorithms	$D = 10$				$D = 30$				$D = 50$			
	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.
NBIPOPaCMA	1.13×10^{1}	1.26×10^{1}	1.22×10^{1}	5.70×10^{-1}	4.00×10^{2}	5.85×10^{2}	4.75×10^{2}	6.75×10^{1}	4.00×10^{2}	2.15×10^{3}	1.48×10^{3}	9.01×10^{2}
PVADE	2.95×10^{2}	5.35×10^{2}	3.64×10^{2}	8.98×10^{1}	3.06×10^{2}	3.61×10^{2}	3.26×10^{2}	1.14×10^{1}	8.27×10^{2}	1.36×10^{3}	1.11×10^{3}	1.85×10^{2}
MVMO	2.48×10^{2}	2.88×10^{2}	2.60×10^{2}	4.03×10^{0}	3.43×10^{2}	6.89×10^{2}	4.74×10^{2}	9.19×10^{1}	9.09×10^{2}	1.31×10^{3}	1.08×10^{3}	1.10×10^{2}
IPOG-MVMO	$\mathbf{5.75 \times 10^{0}}$	$\mathbf{1.11 \times 10^{1}}$	$\mathbf{7.34 \times 10^{0}}$	$\mathbf{2.45 \times 10^{-1}}$	3.95×10^{2}	5.17×10^{2}	4.34×10^{2}	2.45×10^{1}	5.95×10^{2}	1.17×10^{3}	7.34×10^{2}	2.45×10^{2}
F28												
Algorithms	$D = 10$				$D = 30$				$D = 50$			
	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.	Best	Worst	Mean	Std.
NBIPOPaCMA	3.60×10^{0}	1.11×10^{1}	9.63×10^{0}	3.31×10^{0}	3.00×10^{2}	3.00×10^{2}	3.00×10^{2}	0	4.00×10^{2}	4.00×10^{2}	4.00×10^{2}	0
PVADE	8.45×10^{1}	3.01×10^{2}	2.38×10^{2}	4.52×10^{1}	3.00×10^{2}	3.00×10^{2}	3.00×10^{2}	2.23×10^{-5}	4.00×10^{2}	3.54×10^{5}	4.64×10^{2}	4.39×10^{2}
MVMO	9.87×10^{1}	2.90×10^{2}	1.88×10^{2}	1.00 x10^{2}	3.00×10^{2}	3.00×10^{2}	3.00 x10^{2}	5.36×10^{-4}	4.00×10^{2}	4.00×10^{2}	4.00×10^{2}	1.29×10^{-2}
IPOG-MVMO	$\mathbf{3.00 \times 10^{0}}$	$\mathbf{8.72 \times 10^{0}}$	$\mathbf{4.49 \times 10^{0}}$	$\mathbf{2.37 \times 10^{-1}}$	$\mathbf{3.00 \times 10^{2}}$	$\mathbf{3.00 \times 10^{2}}$	$\mathbf{3.00 \times 10^{2}}$	$\mathbf{2.37 \times 10^{-5}}$	$\mathbf{4.00 \times 10^{2}}$	$\mathbf{4.00 \times 10^{2}}$	$\mathbf{4.00 \times 10^{2}}$	$\mathbf{1.13 \times 10^{-2}}$

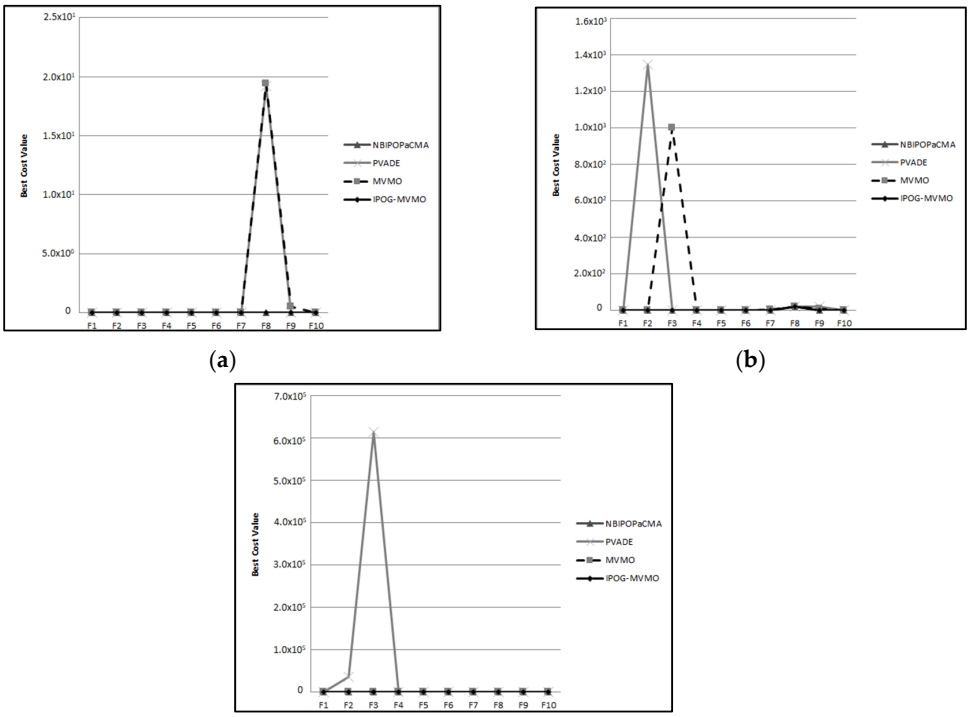

Figure 1. Comparison of the best cost value for F1–F10 at (**a**) 10 dimensions, (**b**) 30 dimensions, and (**c**) 50 dimensions.

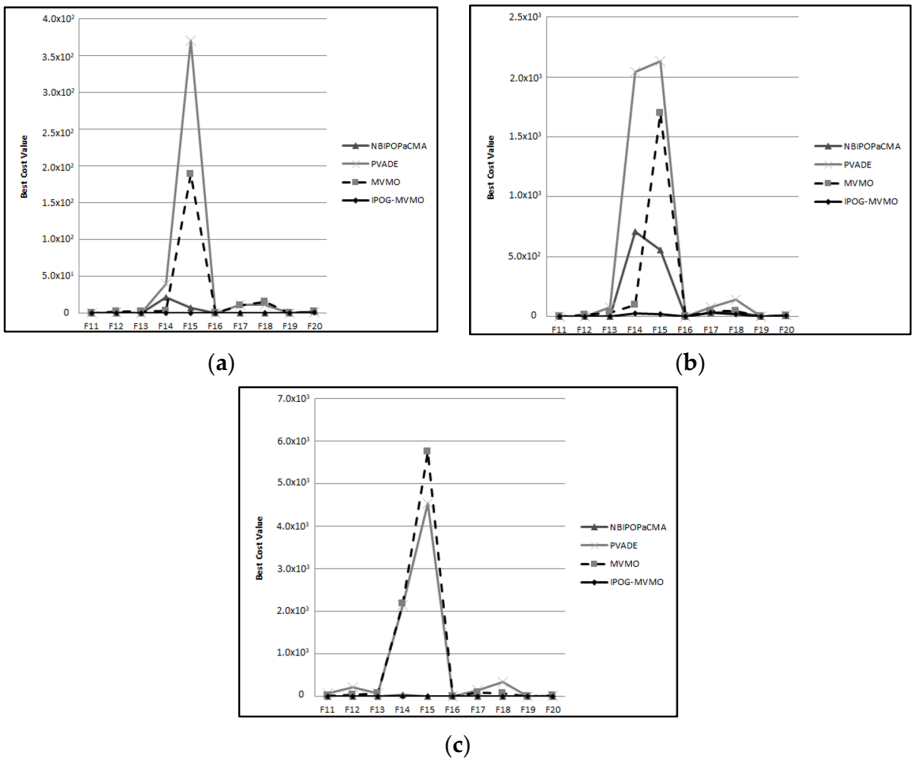

Figure 2. Comparison of the best cost value for F11–F20 at (**a**) 10 dimensions, (**b**) 30 dimensions, and (**c**) 50 dimensions.

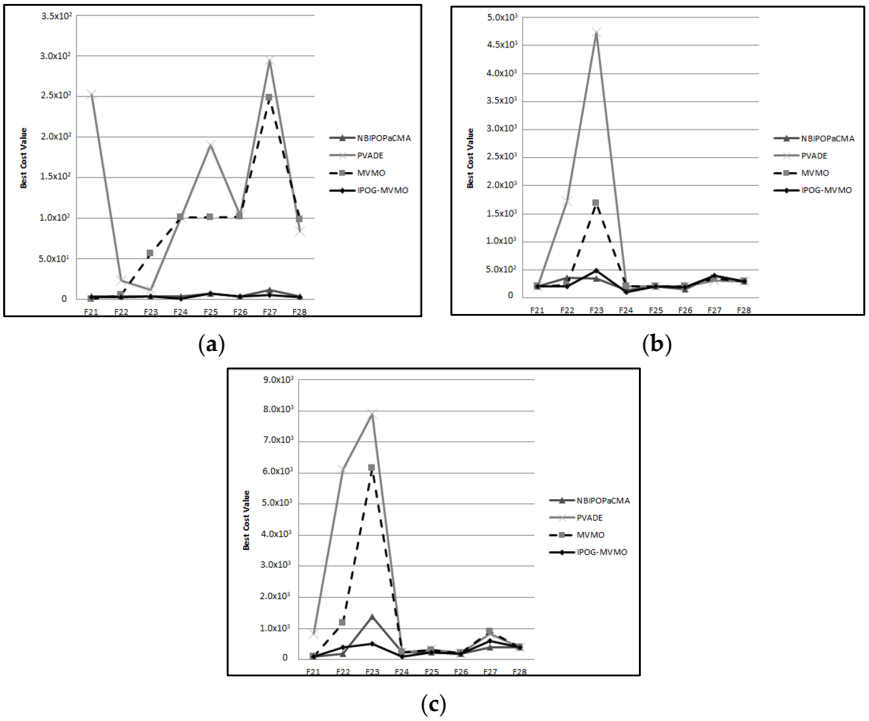

Figure 3. Comparison of the best cost value for F21–F28 at (**a**) 10 dimensions, (**b**) 30 dimensions, and (**c**) 50 dimensions.

To solve unimodal functions F1–F5, the proposed algorithm was able to search the solution without error in every case except F3 (D = 50), where NBIPOPaCMA yielded the better solution, as is shown in Table 2. Moreover, when considering the mean of the error, IPOG-MVMO yielded a zero mean in 10 out of 15 cases. In other words, in these 10 cases, IPOG-MVMO guaranteed the best solution for every trial in every case. It seems that IPOG-MVMO is a suitable technique for solving unimodal problems.

The multimodal functions F6–F20 were analyzed in three groups based on their complexity. To solve multimodal functions in the D = 10 group with the lowest complexity, IPOG-MVMO attained results with no error, except for F14, F15, and F20. Nevertheless, these results were still satisfactory when compared with other algorithms. For the D = 30 group, IPOG-MVMO performed worse than the other algorithms only in two cases, which were F8 and F16. Moreover, IPOG-MVMO also attained error-free results in F6 and F10–F13. For the D = 50 group with the highest complexity, IPOG-MVMO yielded lower performance than the other algorithms in only five cases, which were F8, F9, F16, F17, and F20.

To solve the composition functions F21–F28, the proposed algorithm also outperformed the other algorithms in all cases of D = 10. The algorithm was able to attain better results than the other algorithms in F22 and F24 for D = 30, and in F23, F24, and F26 for D = 50, respectively. Note that this type of function could not provide zero errors in all cases due to its complexity.

The superior results for each problem are highlighted in bold where the best value of IPOG-MVMO was obtained and was better than the other methods; although sometimes, more than one method provided the best value. Clearly, among the 28 × 3 cases, there were 68 cases in which IPOG-MVMO yielded the best result. Likewise, when considering the mean value, IPOG-MVMO yielded the best result in 59 out of the 28 × 3 cases. The results indicated that applying IPM and OGM to MVMO provided a solution that was closer to the optimal solution.

Some examples for the performance comparison were selected, as presented in Figure 4. We also present the average error with respect to the number of iterations (NP) in Figure 4. The average error was computed over 50 trials. The results showed that applying IPM and

OGM to MVMO can accelerate the search process to achieve a solution that was closer to the optimal solution within a smaller number of iterations.

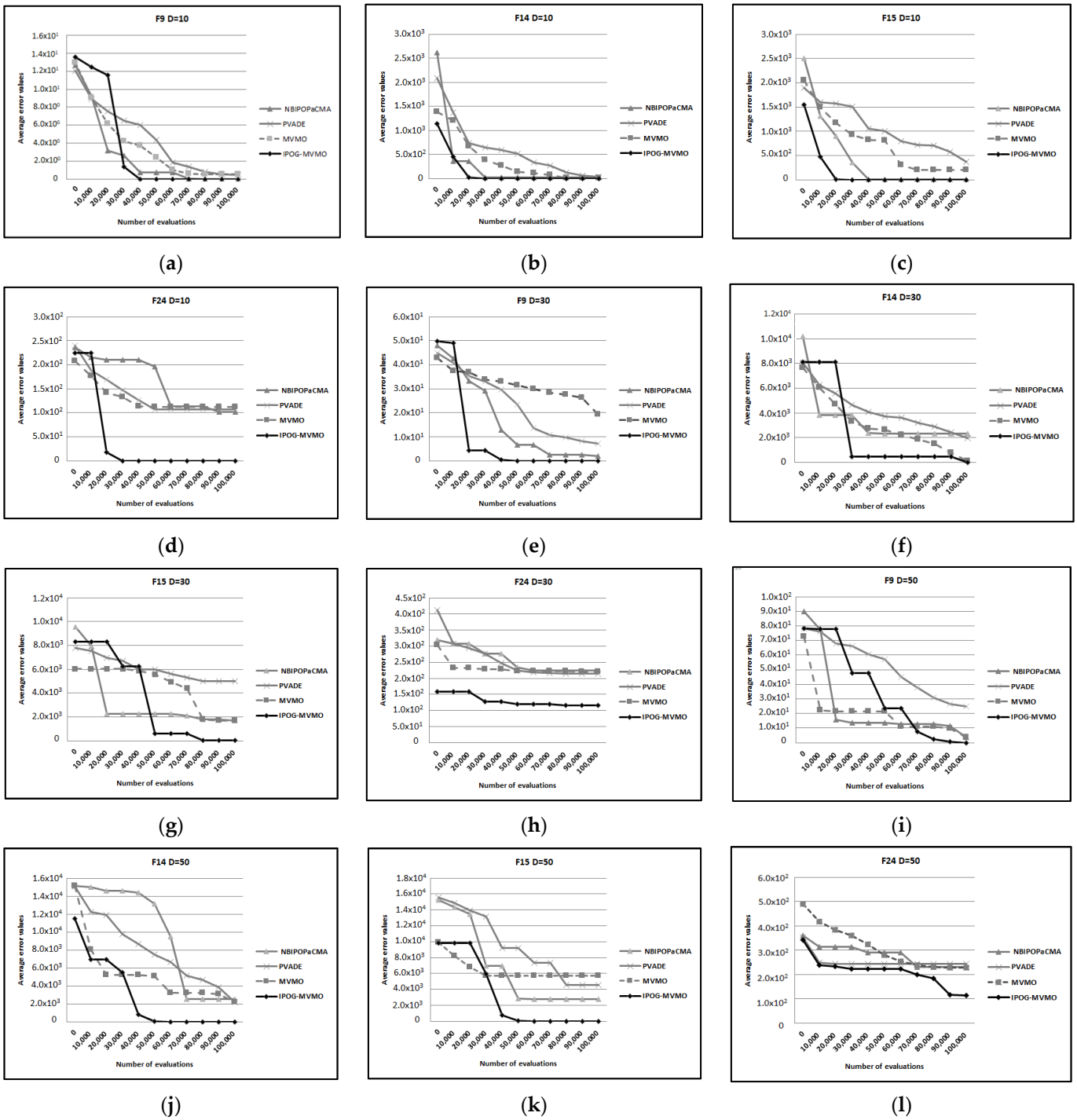

Figure 4. Examples of the performance comparison among IPOG-MVMO, NBIPOPaCMA, PVADE, and MVMO. F9, F14, F15, and F24 at 10, 30, and 50 dimensions were selected. The vertical axis represents the averaged error, while the horizontal axis represents the number of iterations. All were averaged over 50 trials. (**a**–**d**) F9, F14, F15, and F24 with D = 10; (**e**–**h**) F9, F14, F15, and F24 with D = 30; and (**i**–**l**) F9, F14, F15, and F24 with D = 50, respectively.

4. Discussion

4.1. Optimal Solutions

Twenty-eight different functions of three types were taken to verify our proposed method. The experimental results in terms of error are shown in Tables 2–8. When considering the best solution out of all the methods, it was found that IPOG-MVMO achieved the best result in 68 cases out of 84 cases, or 80.95%. Next, three types of functions were further analyzed separately. For the unimodal functions F1–F5, the proposed method offered the best result in 14 out of 15 cases, or 93.33%. This is quite acceptable when applying our method to this type of function. In the case of the multimodal functions F6–F20, there were 45 cases in total. Among these cases, IPOG-MVMO yielded the best result in 37 cases, or 82.22%. This was slightly lower than for the unimodal function due to the complexity of the problem. The remaining functions were the composition functions, F21–F28. The proposed method yielded the best result in 17 out of 24 cases, or 70.83%, which was lower than for unimodal and multimodal functions. Subsequently, the results can be discussed from a different perspective, where we mainly focused on the number of dimensions. There were 28 cases for each dimension and our method yielded the best result in 27, 22, and 19 cases, or 96.43%, 78.57%, and 67.86%, respectively.

In essence, zero error indicates whether the algorithm can reach the exact solution within the pre-defined number of iterations. IPOG-MVMO provided zero error in 33 out of 84 cases. Table 9 presents the likelihood that IPOG-MVMO and four comparative methods could reach the exact solution in terms of function type and the number of dimensions. YYPO, a method inspired by the philosophy of creating a balance between two concepts, was also conducted and is included in this table. For unimodal functions, there was only one case at $D = 50$ in which the error was non-zero, whereas the percentage dropped when the number of dimensions was higher in multimodal functions. Moreover, NBIPOPaCMA presented a higher likelihood of obtaining zero error at $D = 50$; however, IPOG-MVMO still provided the best results at $D = 10$ and $D = 30$. For the composition function, the best solution over 50 trials did not reach the exact solution for any algorithm. Since the function type and the number of dimensions corresponded to the problem difficulty and complexity, it can be inferred that although our IPOG-MVMO method outperformed the other comparative methods, the performance gradually decreased as the problem difficulty and complexity rose. However, using different parameter settings might result in greater performance for solving black box problems.

Table 9. Likelihood of obtaining zero error in 50 trials.

Algorithms	Unimodal Functions			Multimodal Functions			Composition Functions		
	D	*D*	*D*	*D*	*D*	*D*	*D*	*D*	*D*
NBIPOPaCMA	100	80	60	33.33	13.33	20	0	0	0
PVADE	100	60	20	26.67	6.67	0	0	0	0
MVMO	100	20	40	13.33	6.67	13.33	0	0	0
YYPO	40	40	20	6.67	6.67	6.67	0	0	0
IPOG-MVMO	100	100	80	80	33.33	13.33	0	0	0

When considering the mean value of error in Tables 2–8, it was found that IPOG-MVMO yielded the best result in 59 out of 84 cases, or 70.24%. Typically, the zero mean rarely occurs because the method must reach the exact solution in every trial. Our proposed method achieved zero mean in 13 cases, although most were unimodal functions. According to the analysis of the function type, IPOG-MVMO yielded the best result in 12 cases, or 80%, for the unimodal functions, 34 cases, or 75.56%, for the multimodal functions, and 13 cases, or 54.17%, for the composition functions. It can be seen that the type of function also affected the performance in terms of the mean value. In the case of the number of dimensions, IPOG-MVMO yielded the best result in 25 cases, or 89.29%, 18 cases, or 64.29%, and 16 cases, or 57.14%, in 10, 30, and 50 dimensions, respectively. The number of dimensions was another factor that impacted our algorithm.

Finally, the worst value of error presented a solution to be expected over a specific number of trials. IPOG-MVMO achieved the best result in 48 cases, or 57.14%, which was higher than the other methods. Again, the method still worked well in 11 cases, or 73.33%, of unimodal functions, while it yielded the best result in 29 cases, or 64.44%, and eight cases, or 33.33%, in multimodal and composition functions, respectively. In terms of the number of dimensions, our method achieved the best result in 20 out of 28 cases in 10 dimensions. For the cases with 30 dimensions, our method yielded the best result in 12 cases, or 42.86%, and the percentage became higher when the number of dimensions was 50, with a 57.14% rate of obtaining the best result. More parameter settings and experiments are required to identify the relationship between the number of dimensions and the optimal solutions in the worst-case scenario.

4.2. Stability

The standard deviation of error can be used to measure the stability of the proposed method compared with the other methods. Zero standard deviation means that the method can provide the same solution over a limited number of trials. Under the experiments, IPOG-MVMO obtained the best result in 46 out of 84 cases, or 54.76%, and there were 12 cases of zero standard deviation, 10 of which were uniform functions. Based on the function type, the unimodal function was the type with the highest stability, and the best result was obtained in 11 out of 15 cases, whereas the multimodal functions yielded the best result in 26 out of 45 cases. Unfortunately, the composition functions method yielded the best result only in nine out of 24 cases, which was greater than the other methods, but most of the cases were in 10 dimensions. For the number of dimensions, the method yielded the best result in 20 cases (71.43%), 11 cases (39.29%), and 15 cases (53.57%), in 10, 30, and 50 dimensions, respectively. Without taking the composition functions into consideration, it seems that the number of dimensions had a minor impact on the stability of the method, because, as previously mentioned, the composition method worked well only in 10 dimensions.

5. Conclusions

In this study, IPOG-MVMO, a combination of the interior point method (IPM), the opposite gradient method (OGM), and mean-variance mapping optimization (MVMO), was proposed to identify a solution for the continuous real-world optimization problem. IPM and OGM were combined with the original MVMO in the area of an IPM local search and then a new population was created using the opposite gradient concept. This procedure ensured that the newly created population was located somewhere close to the minimum value so that a better solution could be obtained using MVMO. For experiments, creating a new population close to the minimum value to obtain an accurate and fast solution by using the local search strength of IPM and OGM's solution convergence was a key step in this study. When these two approaches were combined with traditional MVMO, the best cost was achieved, which was better than using a single technique, i.e., MVMO. It was important to generate a new population near the minimum in the optimization problem, which can be applied to optimization problems in many areas, such as energy saving for tissue paper mills through energy efficiency scheduling, and the optimal scheduling of energy and ancillary services. Future research will focus on the hybridization of IPOG-MVMO and other techniques, and improving the algorithms to ensure compatibility with combinatorial real-world problems.

Author Contributions: Conceptualization, T.S., S.P. and C.L.; methodology, T.S., S.P. and C.L.; software, T.S.; validation, T.S. and S.P.; formal analysis, T.S. and S.P.; investigation, S.P.; resources, T.S. and C.L.; data curation, T.S.; writing—original draft preparation, T.S.; writing—review and editing, S.P.; visualization, T.S.; supervision, S.P. and C.L.; project administration, T.S. and S.P.; funding acquisition, T.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Officer of the Higher Education Commission, Ministry of Education, Thailand, through Silpakorn University under Grant Number 5/2555.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available in [31].

Acknowledgments: The authors would like to gratefully thank the Faculty of Information and Communication Technology, Silpakorn University, and the Officer of the Higher Education Commission, Ministry of Education, Thailand, for supporting this work through a grant funded under the Higher Education Research Promotion and National Research University Project of Thailand.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; the collection, analyses, or interpretation of data; the writing of the manuscript; or the decision to publish the results.

References

1. Zeng, Z.; Chen, X.; Wang, K. Energy saving for tissue paper mills by energy-efficiency scheduling under time-of-use electricity tariffs. *Processes* **2021**, *9*, 274. [CrossRef]
2. Sortomme, E.; El-Sharkawi, M.A. Optimal scheduling of vehicle-to-grid energy and ancillary services. *IEEE Trans. Smart Grid* **2012**, *3*, 351–359. [CrossRef]
3. Azhir, E.; Jafari Navimipour, N.; Hosseinzadeh, M.; Sharifi, A.; Darwesh, A. Deterministic and non-deterministic query optimization techniques in the cloud computing. *Concurr. Comput. Pract. Exp.* **2019**, *31*, e5240. [CrossRef]
4. Foroozandeh, Z.; Ramos, S.; Soares, J.; Vale, Z. Energy management in smart building by a multi-objective optimization model and pascoletti-serafini scalarization approach. *Processes* **2021**, *9*, 257. [CrossRef]
5. Belfiore, N.P.; Rudas, I.J. Applications of computational intelligence to mechanical engineering. In Proceedings of the IEEE 15th International Symposium on Computational Intelligence and Informatics (CINTI), Budapest, Hungary, 9–21 November 2014.
6. Hansen, N.; Sibylle, D.M.; Koumoutsakos, P. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evol. Comput.* **2003**, *11*, 1–18. [CrossRef] [PubMed]
7. Auger, A.; Hansen, N. A restart CMA evolution strategy with increasing population size. In Proceedings of the IEEE Congress on Evolutionary Computation (CEC), Edinburgh, Scotland, UK, 2–5 September 2005.
8. Hansen, N. Benchmarking a BI-population CMA-ES on the BBOB-2009 function tested workshop. In Proceedings of the GECCO Genetic and Evolutionary Computation Conference, Montreal, QC, Canada, 8–12 July 2009.
9. Loshchilov, I. CMA-ES with restarts for solving CEC 2013 benchmark problems. In Proceedings of the IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico, 20–23 June 2013.
10. Erlich, I.; Venayagamoorthy, G.K.; Nakawiro, W. A mean-variance optimization algorithm. In Proceedings of the 2010 IEEE World Congress on Computational Intelligence, Barcelona, Spain, 18–23 July 2010.
11. Salazar, E.; Herrera, M.; Camacho, O. An Application of MVMO Based Adaptive PID Controller for Process with Variable Delay. In *Systems and Information Sciences*; Botto-Tobar, M., Zamora, W., Larrea Plúa, J., Bazurto Roldan, J., Santamaría Philco, A., Eds.; Springer: Cham, Switzerland, 2021; Volume 1273, pp. 1–6. [CrossRef]
12. Rueda, J.L.; Erlich, I. Short-term transmission expansion planning by using swarm mean-variance mapping optimization. In Proceedings of the 17th International Conference on Intelligent System Applications to Power Systems, Tokyo, Japan, 1–4 July 2013.
13. Rueda, J.L.; Erlich, I. Hybrid mean-variance mapping optimization for solving the IEEE-CEC 2013 competition problems. In Proceedings of the IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico, 20–23 June 2013.
14. Khoa, T.H.; Vasant, P.M.; Singh, M.S.B.; Dieu, V.N. Swarm based mean-variance mapping optimization for solving economic dispatch with cubic fuel cost function. *Lect. Notes Comput. Sci* **2015**, *9012*, 3–12. [CrossRef]
15. Mori, H.; Ikegami, H. An efficient MVMO-SH method for optimal capacitor allocation in electric power distribution systems. In *Advances in Swarm Intelligence*; Tan, Y., Takagi, H., Shi, Y., Niu, B., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2017; Volume 10386, pp. 1–10. [CrossRef]
16. Chen, X.; Li, Y.; Zhang, Y.; Ye, X.; Xiong, X.; Zhang, F. A novel hybrid model based on an improved seagull optimization algorithm for short-term wind speed forecasting. *Processes* **2021**, *9*, 387. [CrossRef]
17. Sun, X.; Fan, Z.; Ji, Y.; Wang, S.; Yan, S.; Wu, S.; Fu, Q.; Ghazali, K.H. Robust multi-user detection based on hybrid gray wolf optimization. *Concurr. Comput. Pract. Exp.* **2021**, *33*, e5273. [CrossRef]
18. Valenta, D.; Langer, M. On numerical 2D P colonies modelling the gray wolf optimization algorithm. *Processes* **2021**, *9*, 330. [CrossRef]
19. Punnathanam, V.; Kotecha, P. Yin-yang-pair optimization: A novel lightweight optimization algorithm. *Eng. Appl. Artif. Intell.* **2016**, *54*, 62–79. [CrossRef]
20. Saenphon, T.; Lursinsap, C. Fast evolutionary solution finding for optimization using opposite gradient movement. In Proceedings of the International Conference on Natural Computation (ICNC), Shanghai, China, 26–28 July 2011.

21. Saenphon, T.; Phimoltares, S.; Lursinsap, C. Combining new fast opposite gradient search with ant colony optimization for solving travelling salesman problem. *Eng. Appl. Artif. Intell* **2014**, *35*, 324–334. [CrossRef]
22. Narendra, K.K. A new polynomial-time algorithm for linear programming. *Combinatorica* **1984**, *4*, 373–395. [CrossRef]
23. Ohmori, S.; Yoshimoto, K. A Primal-Dual Interior-Point Method for Facility Layout Problem with Relative-Positioning Constraints. *Algorithms* **2021**, *14*, 60. [CrossRef]
24. Boyd, S.; Vandenberghe, L. *Convex Optimization*; Cambridge University Press: Cambridge, UK, 2004; pp. 561–620. [CrossRef]
25. Rueda, J.L.; Erlich, I. Evaluation of the mean-variance mapping optimization for solving multimodal problems. In Proceedings of the IEEE Symposium Series on Computational Intelligence, Singapore, 16–19 April 2013.
26. Cepeda, J.C.; Rueda, J.L.; Erlich, I.; Korai, A.W.; Gonzalez-Longatt, F.M. Mean-variance mapping optimization algorithm for power system applications in DIgSILENT power factory. In *PowerFactory Applications for Power System Analysis*; Springer: Cham, Switzerland, 2014; pp. 267–296. [CrossRef]
27. Coelho, L.D.; Ayala, H.V.H.; Freire, R.Z. Population's variance-based adaptive differential evolution for real parameter optimization. In Proceedings of the IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico, 20–23 June 2013.
28. Tianjun, L.; Stutzle, T. Benchmark results for a simple hybrid algorithm on the CEC 2013 benchmark set for real-parameter optimization. In Proceedings of the IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico, 20–23 June 2013.
29. Liang, J.; Qin, A.K.; Suganthan, P.N.; Baskar, S. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Trans. Evol. Comput.* **2006**, *10*, 281–295. [CrossRef]
30. Liang, J.; Qu, B.Y.; Suganthan, P.N.; Hernández-Díaz, A.G. *Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session and Competition on Real-Parameter Optimization*; Computational Intelligence Laboratory, Zhengzhou University: Zhengzhou, China; Nanyang Technological University: Singapore, 2013.
31. Liang, J.J.; Qu, B.Y.; Suganthan, P.N. *Zhengzhou China and Technical Report*; Nanyang Technological University: Singapore, 2013.